Diego Calvanese
Boris Konev

28^{th} International Workshop on Description Logics

Athens, Greece
7–10 June 2015

Preface

The International Workshop on Description Logics is the main annual event of the Description Logic research community. It is the forum at which those interested in description logics, from both academia and industry, meet to discuss ideas, share information, and compare experiences. The workshop explicitly welcomes submissions from researchers that are new to the area and provides quality feedback via peer-reviewing, while at the same time being of an "inclusive" nature with a very high acceptance rate. There are only informal (electronic) proceedings and inclusion of a paper there is not supposed to preclude its publication at conferences. Further information can be found on the DL Web pages at http://dl.kr.org/.

This volume of informal proceedings contains the papers presented at the 28th International Workshop on Description Logics (DL 2015) held on June 6-9, 2015 in Athens (Greece). This year there were 68 submissions, divided among full papers presenting original research, and extended abstracts (of at most 3 pages). Submissions have been judged solely based upon their content and quality, and the type of submission had no bearing on the decision between long oral, short oral and poster presentation. Each submission was reviewed by 3 program committee members or additional reviewers recruited by the PC. In the spirit of inclusiveness, the committee decided to accept 60 papers, among them 37 as long and short oral presentations and 23 as poster presentations. We thank all program committee members and additional reviewers for their invaluable effort.

The program also included 3 invited talks, which were given by Carsten Lutz, Axel Polleres, and Maarten de Rijke. The abstracts of these talks are included in this volume.

We also gratefully acknowledge the Artificial Intelligence Journal and the Foundation for Principles of Knowledge Representation and Reasoning (KR Inc.) for financial support and EasyChair for providing a convenient and efficient platform for preparing the program. Last but not least we thank all authors and participants of DL 2015.

June 2015
Athens

Diego Calvanese
Boris Konev

Table of Contents

Invited Talks

Oral Presentations

Poster Presentations

Conference Organisation

General Chairs

Giorgos Stamou	National Technical University of Athens, Greece
Giorgos Stoilos	National Technical University of Athens, Greece

Program Chairs

Diego Calvanese	Free University of Bozen-Bolzano, Italy
Boris Konev	University of Liverpool, UK

Program Committee

Alessandro Artale	Free University of Bozen-Bolzano, Italy
Franz Baader	TU Dresden, Germany
Meghyn Bienvenu	CNRS & Université Paris-Sud, France
Alex Borgida	Rutgers University, USA
Stefan Borgwardt	TU Dresden, Germany
Elena Botoeva	Free University of Bozen-Bolzano, Italy
Bernardo Cuenca Grau	University of Oxford, UK
Giuseppe De Giacomo	Sapienza Università di Roma, Italy
Chiara Ghidini	FBK-irst, Italy
Silvio Ghilardi	Università degli Studi di Milano, Italy
Laura Giordano	Università del Piemonte Orientale, Italy
Víctor Gutiérrez Basulto	University of Bremen, Germany
Pascal Hitzler	Wright State University, USA
Ian Horrocks	University of Oxford, UK
Mark Kaminski	University of Oxford, UK
Yevgeny Kazakov	The University of Ulm, Germany
C. Maria Keet	University of Cape Town, South Africa
Pavel Klinov	University of Ulm, Germany
Ilianna Kollia	National Technical University of Athens, Greece
Roman Kontchakov	Birkbeck, University of London, UK
Oliver Kutz	Free University of Bozen-Bolzano, Italy
Maurizio Lenzerini	University of Rome "La Sapienza", Italy
Michel Ludwig	TU Dresden, Germany
Carsten Lutz	Universität Bremen, Germany
Thomas Meyer	Centre for Artificial Intelligence Research, UKZN and CSIR Meraka, South Africa
Barbara Morawska	TU Dresden, Germany

Ralf Möller	Universität zu Lübeck, Germany
Yavor Nenov	University of Oxford, UK
Magdalena Ortiz	Vienna University of Technology, Austria
Jeff Z. Pan	University of Aberdeen, UK
Peter Patel-Schneider	Nuance Communications
Riccardo Rosati	Sapienza Università di Roma, Italy
Sebastian Rudolph	Technische Universität Dresden, Germany
Vladislav Ryzhikov	Free University of Bozen-Bolzano, Italy
Uli Sattler	University of Manchester, UK
Viorica Sofronie-Stokkermans	University of Koblenz and MPII Saarbrücken, Germany
David Toman	University of Waterloo, Canada
Dmitry Tsarkov	The University of Manchester, UK
Anni-Yasmin Turhan	University of Oxford, UK
Grant Weddell	University of Waterloo, Canada
Frank Wolter	University of Liverpool, UK
Guohui Xiao	Free University of Bozen-Bolzano, Italy

Additional Reviewers

Armas Romero, Ana
Bate, Andrew
Bibel, Wolfgang
Bozzato, Loris
Britz, Arina
Carral, David
Casini, Giovanni
De Masellis, Riccardo
Ecke, Andreas
Feier, Cristina
Fiorentini, Camillo
Ibanez-Garcia, Yazmin Angelica
Jung, Jean Christoph
Klarman, Szymon
Krisnadhi, Adila A.
Lippmann, Marcel
Neuenstadt, Christian
Oezcep, Oezguer Luetfue
Ozaki, Ana
Rens, Gavin
Schiller, Marvin
Sebastiani, Roberto
Sengupta, Kunal
Simkus, Mantas
Stepanova, Daria
Thost, Veronika
Wang, Cong
Zarrieß, Benjamin
Zheleznyakov, Dmitriy

Query Rewriting Beyond DL-Lite

Carsten Lutz

Fachbereich Informatik, Universität Bremen, Germany

1 Abstract of Invited Talk

Query rewriting has become a very prominent tool for efficiently implementing ontology-mediated querying in practice. The technique was originally introduced in the context of DL-Lite [4], but is now increasingly being used also for more expressive DLs. While rewritings are not guaranteed to exist beyond DL-Lite, the simple structure of ontologies that emerge from practical applications gives hope that non-existence of rewritings is a rare case.

The aim of the talk is to survey FO- and Datalog-rewriting of ontology-mediated queries in description logics beyond DL-Lite. It is structured into three parts. The first part is concerned with FO-rewritings in Horn-DLs such as $\mathcal{EL}$, $\mathcal{ELI}$, and Horn-$\mathcal{SHI}$, the second part considers FO-rewritings in non-Horn-DLs such as $\mathcal{ALC}$ and $\mathcal{ALCI}$, and the third part is about Datalog-rewritings in non-Horn DLs. In all three parts, I will try to emphasize useful characterizations of FO-rewritability, practically efficient algorithms for constructing rewritings, and relevant computational complexity results.

The presentation is based on joint work with Meghyn Bienvenu, Balder ten Cate, Peter Hansen, İnanc Seylan, and Frank Wolter. The subsequent section provides some supplementary material that is featured in the talk, but has not yet been published elsewhere. It establishes a link between the first and the second part of the talk.

2 Supplementary Material

In [3, 7], we have proposed an approach to deciding the FO-rewritability of OMQs for the case where the ontology/TBox is formulated in a Horn DL such as $\mathcal{EL}$, $\mathcal{ELI}$, and Horn-$\mathcal{ALCI}$. The approach has led to efficient (yet complete) practical implementations, and it relies on a characterization of FO-rewritability in terms of tree-shaped ABoxes. Intuitively, the characterization relies on a property of TBoxes that is called 'unraveling tolerance' and which typically is enjoyed by Horn DLs, but not by DLs that include forms of disjunction. In contrast, the only known complete approach to deciding FO-rewritability of OMQs in which the TBox is formulated in full (non-Horn) $\mathcal{ALC}$ and $\mathcal{ALCI}$ is via the CSP connection in [9, 2]. Since $\mathcal{ALC}$- and $\mathcal{ALCI}$-TBoxes are typically not unraveling tolerant, it might seems that these two world are largely unrelated. In the following, though, we point out a characterization of FO-rewritability in full $\mathcal{ALCI}$ that establishes an interesting connection to tree-shaped ABoxes and thus to the Horn case. We

consider *Boolean atomic queries (BAQs)*, that is, queries of the form $\exists x\, A(x)$ with A a concept name.

An *ontology-mediated query (OMQ)* is a triple $Q = (\mathcal{T}, \Sigma, q)$ with $\mathcal{T}$ a TBox, Σ an ABox signature (set of concept and role names), and q a query. An *OBDA language* is a set of OMQs. We use $(\mathcal{ALCI}, \text{BAQ})$ to denote the OBDA language that consists of all OMQs $(\mathcal{T}, \Sigma, q)$ with $\mathcal{T}$ an $\mathcal{ALCI}$-TBox and q a BAQ, and likewise for other combinations of a DL and a query language. An OMQ $Q = (\mathcal{T}, \Sigma, q)$ is *FO-rewritable* if there is an FO-sentence φ such that for every Σ-ABox $\mathcal{A}$ that is consistent w.r.t. $\mathcal{T}$, we have $\mathcal{A} \models Q$ iff $\mathcal{A} \models \varphi$.

As usual in OBDA, an *ABox* is a finite set of assertions of the form $A(a)$ or $r(a, b)$ with A a concept name and r a role name. We write $r^-(a, b) \in \mathcal{A}$ to mean $r(b, a) \in \mathcal{A}$ and use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individuals used in $\mathcal{A}$. An ABox $\mathcal{A}$ is *tree-shaped* if the undirected graph $(\mathsf{Ind}(\mathcal{A}), \{\{a, b\} \mid r(a, b) \in \mathcal{A}\})$ is a tree and whenever $r(a, b) \in \mathcal{A}$, then (i) $s(a, b) \in \mathcal{A}$ implies $r = s$ and (ii) $\mathcal{A}$ contains no assertion of the form $s(b, a)$. Tree-shapedness of conjunctive queries (CQs) is defined accordingly. Note that, in both cases, our trees allow upwards- and downwards-directed edges, but no multi-edges.

We now introduce unravelings of ABoxes and the notion of unraveling tolerance [9]. Let $\mathcal{A}$ be an ABox and $a \in \mathsf{Ind}(\mathcal{A})$. The *unraveling* $\mathcal{A}^u_a$ *of* $\mathcal{A}$ *at* a is the following (possibly infinite) ABox:

- $\mathsf{Ind}(\mathcal{A}^u_a)$ is the set of sequences $b_0 r_0 b_1 \cdots r_{n-1} b_n$, $n \geq 0$, such that $b_0 = a$, $b_0, \ldots, b_n \in \mathsf{Ind}(\mathcal{A})$ and $r_0, \ldots, r_{n-1}$ are (potentially inverse) roles;
- for each $C(b) \in \mathcal{A}$ and $\alpha = b_0 \cdots b_n \in \mathsf{Ind}(\mathcal{A}^u_a)$ with $b_n = b$: $C(\alpha) \in \mathcal{A}^u_a$;
- for each $\alpha = b_0 r_0 \cdots r_{n-1} b_n \in \mathsf{Ind}(\mathcal{A}^u_a)$ with $n > 0$: $r_{n-1}(b_0 \cdots b_{n-1}, \alpha) \in \mathcal{A}^u_a$.

For all $\alpha = b_0 \cdots b_n \in \mathsf{Ind}(\mathcal{A}^u_a)$, we write $\mathsf{tail}(\alpha)$ to denote b_n. Note that $\mathcal{A}^u_a$ is tree-shaped. An OMQ $Q = (\mathcal{T}, \Sigma, q)$ is *unraveling tolerant* if for every Σ-ABox $\mathcal{A}$, $\mathcal{A} \models Q$ implies $\mathcal{A}^u_a \models Q$ for some $a \in \mathsf{Ind}(\mathcal{A})$. Note that this is essentially the same notion of unraveling tolerance as introduced in [9].

It can be shown as in [9] that in OBDA languages where the TBoxes are formulated in Horn DLs such as $\mathcal{EL}$, $\mathcal{ELI}$, and Horn-$\mathcal{ALC}$ and where queries are BAQs or *atomic queries* (*AQs*, queries of the form $A(x)$ with A a concept name), all OMQs are unraveling tolerant. This underlies the following characterization from [3].

Theorem 1 ([3]). *A BAQ $Q = (\mathcal{T}, \Sigma, q)$ from (Horn-$\mathcal{ALCI}$, AQ) is FO-rewritable iff there exists a $k \geq 0$ such that for all tree-shaped Σ-ABoxes $\mathcal{A}$ which are consistent with $\mathcal{T}$, $\mathcal{A} \models Q$ implies $\mathcal{A}|_k \models Q$ where $\mathcal{A}_k$ is $\mathcal{A}$ with all nodes on level exceeding k removed.*

Using a pumping argument, it can be shown that if there is any bound k as Theorem 1, then we can choose $k = 2^{2|\mathcal{T}|}$. Based on this, worst-case optimal (ExpTime) decision procedures for FO-rewritability in (Horn-$\mathcal{ALCI}$, AQ) can be devised using automata methods. Efficiently computing rewritings in practice requires further algorithm engineering [7].

We will now establish a characterization of FO-rewritability in the non-Horn OBDA language $(\mathcal{ALCI}, \text{BAQ})$. It is shown in [2] that for every OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ALCI}, \text{BAQ})$, there is a CSP template (a finite relational structure) T_Q over signature Σ such that for all Σ-ABoxes $\mathcal{A}$, we have $\mathcal{A}, \mathcal{T} \models q$ iff $\mathcal{A} \not\to T_Q$, that is, iff there is no homomorphism from $\mathcal{A}$ to T_Q (in the standard sense of labeled directed graphs). We say that a CSP template T is *FO-definable* if there is an FO-sentence φ such that for all finite Σ-structures S, we have $S \to T$ iff $S \models \varphi$. The complement of T is *definable in monadic Datalog* if there is a monadic Datalog program Π such that for all finite Σ-structures S, we have $S \not\to T$ iff $S \models \Pi$. Note that a CSP template is FO-definable iff its complement is (just take the negation of the defining sentence), but this is not true for monadic Datalog definability.

It is easy to see that an OMQ Q is FO-rewritable if and only if the complement of T_Q is FO-definable, and likewise for rewritability into monadic Datalog. In [2], this observation is used together with results on the FO-definability of CSPs [8] to show the following.

Theorem 2 ([2]). *FO-rewritability in $(\mathcal{ALCI}, BAQ)$ and $(\mathcal{ALCI}, AQ)$ is decidable and* NExpTime*-complete.*

This approach is also capable of producing actual rewritings, but unfortunately it is best-case exponential. This calls for a better understanding of FO-rewritability in $(\mathcal{ALCI}, \text{BAQ})$ and related languages, as a basis for more practical (yet complete) approaches.

As a preliminary, we show that unraveling tolerance is equivalent to rewritability into monadic Datalog. This actually follows straightforwardly from known results about CSPs.

Theorem 3. *An OMQ from $(\mathcal{ALCI}, BAQ)$ is unraveling tolerant iff it is rewritable into monadic Datalog.*

Proof. A CSP template T over signature Σ has *tree duality* iff there is a set $\mathcal{O}$ of tree-shaped Σ-structures (called *obstructions* and where tree-shapedness is defined as for ABoxes and CQs above) such that for all finite Σ-structures S, we have $T \leftarrow S$ iff $S \not\leftarrow O$ for all $O \in \mathcal{O}$. It was shown in [5] that T has tree duality iff the complement of T is definable in monadic Datalog. It thus remains to show that an OMQ from $(\mathcal{ALCI}, \text{BAQ})$ is unraveling tolerant iff T_Q has tree duality.

"if". Assume that $Q = (\mathcal{T}, \Sigma, q)$ is unraveling tolerant. Let $\mathcal{O}$ be the set of all tree-shaped Σ-ABoxes $\mathcal{A}$ with $\mathcal{A} \models Q$. Then $\mathcal{O}$ witnesses tree duality: if $T_Q \leftarrow \mathcal{A}$ for some Σ-ABox $\mathcal{A}$, then $\mathcal{A} \not\models Q$; since $\mathcal{B} \models Q$ and $\mathcal{B} \to \mathcal{A}$ implies $\mathcal{A} \models Q$ [2], we thus have $\mathcal{A} \not\leftarrow \mathcal{B}$ for all $\mathcal{B} \in \mathcal{O}$ as required. Conversely, assume that $\mathcal{A}$ is a Σ-ABox with $\mathcal{A} \not\leftarrow \mathcal{B}$ for all $\mathcal{B} \in \mathcal{O}$. Clearly, $\mathcal{A}^u_a \to \mathcal{A}$ for all $a \in \mathsf{Ind}(\mathcal{A})$. Thus, no such $\mathcal{A}^u_a$ is in $\mathcal{O}$, implying that $\mathcal{A}^u_a \not\models Q$. Since Q is unraveling tolerant, $\mathcal{A} \not\models Q$ which implies $T_Q \leftarrow \mathcal{A}$ as required.

"only if". Assume that T_Q has tree duality with set of obstructions $\mathcal{O}$. Let $\mathcal{A}$ be a Σ-ABox with $\mathcal{A} \models Q$. Then $T_Q \not\leftarrow \mathcal{A}$ and thus $\mathcal{A} \leftarrow \mathcal{B}$ for some $\mathcal{B} \in \mathcal{O}$. Since

$\mathcal{B}$ is tree-shaped, $\mathcal{A} \leftarrow \mathcal{B}$ implies $\mathcal{A}_a^u \leftarrow \mathcal{B}$ for some $a \in \mathsf{Ind}(\mathcal{A})$. Consequently $T_Q \not\leftarrow \mathcal{A}_a^u$ which yields $\mathcal{A}_a^u \models Q$ as required. □

We now establish the announced characterization.

Theorem 4. *Let $Q = (\mathcal{T}, q, \Sigma)$ be an OMQ from $(\mathcal{ALCI}, BAQ)$. Then Q is FO-rewritable iff*

1. *Q is FO-rewritable on tree-shaped ABoxes and*
2. *Q is unraveling tolerant.*

Proof. "if". Assume that Q is unraveling tolerant and FO-rewritable on tree-shaped ABoxes. By Theorem 3, the complement of the template T_Q is definable by a monadic Datalog program Π_Q. Let Π'_Q be obtained from Π_Q by identifying the variables in rule bodies in all possible ways and then retaining only those rules whose bodies are a tree-shaped CQ. It can be verified that Π'_Q is a rewriting of Q: $\mathcal{A} \models Q$ implies $\mathcal{A}_a^u \models Q$ for some $a \in \mathsf{Ind}(\mathcal{A})$ (since Q is unraveling tolerant) implies $\mathcal{A}_a^u \models \Pi_Q$ (since Π_Q is a rewriting of Q) implies $\mathcal{A}_a^u \models \Pi'_Q$ (since $\mathcal{A}_a^u$ is tree-shaped) implies $\mathcal{A} \models \Pi'_Q$ (since $\mathcal{A}_A^u \to \mathcal{A}$). Conversely, $\mathcal{A} \models \Pi'_Q$ implies $\mathcal{A} \models \Pi_Q$ (by construction of Π'_Q) implies $\mathcal{A} \models Q$. It is easy to further modify Π'_Q so that in addition to being tree shaped, every role body contains at most one EDB atom.

We now use the existence of Π'_Q to argue that Q has an FO-rewriting φ on tree-shaped ABoxes that takes the form of a union of tree-shaped CQs. Let ψ be an FO-rewriting of Q on tree-shaped ABoxes. By Gaifman's locality theorem, there is a number $d \geq 0$ such that for every Σ-ABox $\mathcal{A}$, we have $\mathcal{A} \models \psi$ iff $\mathcal{A}_d^* \models \psi$ where $\mathcal{A}_d^*$ is obtained by taking the disjoint union of all d-neighborhoods in $\mathcal{A}$; here, the *d-neighborhood* in $\mathcal{A}$ around $a \in \mathsf{Ind}(\mathcal{A})$ is the restriction of $\mathcal{A}$ to all individuals that can be reached from a on a role path in $\mathcal{A}$ of length at most d. Note that ψ is a rewriting of Q and every OMQ from $(\mathcal{ALCI}, \text{BAQ})$ satisfies the property that if a Σ-ABox $\mathcal{A}$ is the disjoint union of ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_k$, then $\mathcal{A} \models Q$ iff $\mathcal{A}_i \models Q$ for at least one $\mathcal{A}_i$. We can thus strengthen the above obervation as follows: for every Σ-ABox $\mathcal{A}$, we have $\mathcal{A} \models \psi$ iff there is some d-neighborhood $\mathcal{N}$ in $\mathcal{A}$ such that $\mathcal{N} \models \psi$. Since both ψ and Π'_Q are rewritings of the same query Q, the same applies to the monadic Datalog program Π'_Q instead of to ψ. Moreover, we can find an $\ell \geq 0$ such that for every Σ-ABox $\mathcal{A}$ with $\mathcal{A} \models \Pi'_Q$, there is an $\mathcal{A}' \subseteq \mathcal{A}$ with $\mathcal{A}' \models \Pi'_Q$ and in which every individual has degree at most ℓ—due to the special shape of Π'_Q, we can in fact simply choose for ℓ the number of IDB relations in Π'_Q. Combining these two observations, we get the following: for every tree-shaped Σ-ABox $\mathcal{A}$ with $\mathcal{A} \models Q$, there is a tree-shaped ABox $\mathcal{A}' \subseteq \mathcal{A}$ of depth at most d and degree at most ℓ such that $\mathcal{A}' \models Q$. We can thus choose as the desired rewriting φ the UCQ that consists of all tree-shaped ABoxes $\mathcal{A}$ (viewed as a CQ) that satisfy $\mathcal{A} \models Q$ and are of depth at most d and of degree at most ℓ.

It remains to note that, due to its syntactic shape, φ is an FO-rewriting not only on tree-shaped ABoxes, but also on unrestricted ones. First assume that $\mathcal{A}$ is a Σ-ABox with $\mathcal{A} \models Q$. Since Q is unraveling tolerant, there then is an

$a \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{A}_a^u \models Q$. Since φ is an FO-rewriting on tree-shaped ABoxes, we get $\mathcal{A}_a^u \models \varphi$. Since φ is a UCQ and $\mathcal{A}_a^u \to \mathcal{A}$, we obtain $\mathcal{A} \models \varphi$. Conversely, assume $\mathcal{A} \models \varphi$. Since φ is a union of tree-shaped CQs, this yields $\mathcal{A}_a^u \models \varphi$ for some $a \in \mathsf{Ind}(\mathcal{A})$, thus $\mathcal{A}_a^u \models Q$ and $\mathcal{A} \models Q$.

"only if". Assume that Q is FO-rewritable. Then it is clearly also FO-rewritable on tree-shaped ABoxes (the same rewriting works). It thus remains to show that Q is unraveling tolerant.

It is proved in [1] that a CSP template T over signature Σ is FO-rewritable iff it has *finite duality*, that is, iff there is a finite set of structures $\mathcal{O}$ such that for all finite Σ-structures S, we have $T \leftarrow S$ iff $S \not\leftarrow O$ for all $O \in \mathcal{O}$. It was shown in [10] that finite duality implies tree duality. In fact, as observed in [8], we can assume w.l.o.g. that the finitely many elements of $\mathcal{O}$ are finite and tree-shaped. One could call this *finite duality in terms of finite trees*.

Now back to our OMQ Q. Since Q is FO-rewritable, so is T_Q. By the above result on finite duality in terms of finite trees, there is thus a finite set Γ of tree-shaped ABoxes such that for all Σ-ABoxes $\mathcal{A}$, we have $\mathcal{A} \models Q$ iff $\mathcal{B} \to \mathcal{A}$ for some $\mathcal{B} \in \Gamma$. Consequently, the UCQ $\widehat{q} = \bigvee_{\mathcal{B} \in \Gamma} q_{\mathcal{B}}$ is an FO-rewriting of Q, where $q_{\mathcal{B}}$ is $\mathcal{B}$ viewed as a Boolean CQ in the obvious way. Note that $\widehat{q}$ is a disjunction of tree-shaped CQs. It is thus straightforward to show that for all Σ-ABoxes $\mathcal{A}$, we have $\mathcal{A} \models \widehat{q}$ iff $\mathcal{A}_a^u \models \widehat{q}$ for some $a \in \mathsf{Ind}(\mathcal{A})$. The unraveling tolerance of Q follows. □

The proof of Theorem 4 also yields the following corollary, which strengthens the observation from [2] that in $(\mathcal{ALCI}, \mathrm{BAQ})$, every FO-rewritable OMQ is UCQ-rewritable (essentially a consequence of Rossmann's homomorphism preservation theorem).

Corollary 1. *If an OMQ in $(\mathcal{ALCI}, BAQ)$ is FO-rewritable, then it is rewritable into a union of tree-shaped conjunctive queries.*

We remark that, even when switching to the OBDA language $(\mathcal{ALC}, \mathrm{BAQ})$, it is not possible to replace the undirected trees in Corollary 1 with directed trees.

We close with some discussion of Theorem 4. As future work, we plan to adapt the result from $(\mathcal{ALCI}, \mathrm{BAQ})$ to $(\mathcal{ALCI}, \mathrm{AQ})$ and to use them as a basis for developing practically feasible algorithms that construct FO-rewritings. Dealing with $(\mathcal{ALCI}, \mathrm{AQ})$ seems to require more liberal definitions of tree-shaped ABoxes and of unraveling tolerance which allow for back-edges to the root as in the tree-model property for DLs with nominals. To obtain a first impression of the effect of answer variables, the reader might want to consider the following OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ALCI}, \mathrm{BAQ})$:

$$\mathcal{T} = \{P \sqcap \exists r.P \sqsubseteq A, \neg P \sqcap \exists r.\neg P \sqsubseteq A\} \qquad \Sigma = \{r\} \qquad q = \exists x\, A(x)$$

and its variation Q' from $(\mathcal{ALCI}, \mathrm{BAQ})$ obtained by replacing q with the AQ $q' = A(x)$. Q is not unraveling tolerant as witnessed by the ABox $\mathcal{A} = \{r(a, a)\}$ which satisfies $\mathcal{A} \models Q$, but $\mathcal{A}_a^u \not\models Q$. The same is true for Q' if the notion of

unraveling tolerance is adapted in a naive way to non-Boolean OMQs. However, while Q is not FO-rewritable (by Theorem 4), it is not too hard to prove that the FO-formula $r(x, x)$ is an FO-rewriting of Q'.

Another interesting question concerns the complexity of deciding FO-rewritability in $(\mathcal{ALCI}, \text{BAQ})$ (and of course also $(\mathcal{ALCI}, \text{AQ})$) via Theorem 4. It is shown in [5] that unraveling tolerance is decidable (in 3-ExpTime) and using techniques from [2], it is possible to prove NExpTime-hardness. We speculate that the problem might actually be NExpTime-complete. Regarding FO-rewritability in $(\mathcal{ALCI}, \text{BAQ})$ on tree-shaped ABoxes, it seems likely that a 2-ExpTime lower bound can be established by combining reductions from [3] and [6]—thus FO-rewritability on tree-shaped ABoxes would be harder than on unrestricted ABoxes! However, if we already know that Q is unraveling tolerant, then FO-rewritability on trees is trivially in NExpTime, simply by Theorem [6].

Acknowledgements. I am grateful to Frank Wolter for, as always, very helpful and stimulating discussions.

References

1. Atserias, A.: On digraph coloring problems and treewidth duality. Eur. J. Comb. 29(4), 796–820 (2008)
2. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Trans. Database Syst. 39(4), 33:1–33:44 (2014)
3. Bienvenu, M., Lutz, C., Wolter, F.: First-order rewritability of atomic queries in Horn description logics. In: Proc. of IJCAI (2013)
4. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)
5. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. SIAM J. Comput. 28(1), 57–104 (1998)
6. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. of KR (2006)
7. Hansen, P., Lutz, C., İnanç Seylan, Wolter, F.: Efficient query rewriting in the description logic $\mathcal{EL}$ and beyond. In: Proc. of IJCAI (2015)
8. Larose, B., Loten, C., Tardif, C.: A characterisation of first-order constraint satisfaction problems. Logical Methods in Computer Science 3(4) (2007)
9. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: Proc. of KR (2012)
10. Nesetril, J., Tardif, C.: Duality theorems for finite structures (characterising gaps and good characterisations). J. Comb. Theory, Ser. B 80(1), 80–97 (2000)

Integrating Open Data: (How) Can Description Logics Help me?

Axel Polleres

Vienna University of Economics and Business, Vienna, Austria

At last year's DL workshop Alon Halevy told us about Web tables and how Google makes sense of tabular data on the Web together with Web knowledge graphs [2]. Somewhat surprinsing, a still more unconquered area for Web data extraction seems to be the realm of Open Data: rather than extracting structured data from the surface Web, another emerging source of data on the Web are lots of structured data sets being published openly on various Open Data Portals (e.g. http://www.publicdata.eu/, http://data.gov.gr/ http://data.gov.uk/, http://www.data.gov/, http://data.gv.at/, http://open.wien.at/, just to name a few). However, despite already offering structured data, these Open Data portals often offer only limited search functionality, and intergrating and using Open Data from these portals involves various challenges, such as data quality problems [3], heterogeneity within metadata descriptions, dynamics, or lack of semantic descriptions of the data. Driven by a practical use case, the Open City Data pipeline project [1], we will report on experiences and obstacles for collecting and integrating Open Data across various data sets. We wil discuss how both methods from knowledge representation and reasoning as well as from statistics and data mining can be used to tackle some issues we encountered.

References

1. Bischof, S., Martin, C., Polleres, A., Schneider, P.: Open City Data Pipeline: Collecting, Integrating, and Predicting Open City Data. In: 4th Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data (Know@LOD). Portoroz, Slovenia (May 2015), http://www.polleres.net/publications/bisc-etal-2015KnowLOD.pdf
2. Halevy, A.: Structured data on the web (or, a personal journey away from and back to ontologies). In: Informal Proceedings of the 27th International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1193 (2014)
3. Umbrich, J., Neumaier, S., Polleres, A.: Towards assessing the quality evolution of open data portals. In: ODQ2015: Open Data Quality: from Theory to Practice Workshop. Munich, Germany (Mar 2015), http://polleres.net/publications/umbr-etal-2015ODQ.pdf

Entity-oriented Search Engine Result Pages

Maarten de Rijke

University of Amsterdam, Amsterdam, The Netherlands
derijke@uva.nl

Modern search engine result pages often contain a mixture of results from structured and unstructured sources. Where such mixtures of structured and unstructured information are called for, the state-of-the-art is to organize complex search engine result pages around entities. Generating such a mixture of entity-oriented results in response to a traditional keyword query raises a number of interesting retrieval challenges. How do we link queries to entities? How do we identify different aspects of entities in cases where we are unsure about the user's intent? How do we associate an entity with a topic that a user appears to be interested in? And how do we explain the relation between entities that are being presented as being similar or related?

In this area, a wide variety of complementary and competing proposed solutions exist. This talk provides a snapshot of current approaches to entity-focused search engine result pages, illustrates key developments using example, and outlines open questions and research opportunities.

The talk is based in part on joint work with Lars Buitinck, David Graus, Xinyi Li, Edgar Meij, Daan Odijk, Ridho Reinanda, Isaac Sijaranamual, Manos Tsagkias, Christophe Van Gysel, Nikos Voskarides, and Wouter Weerkamp.

DL-Lite and Conjunctive Queries Extended by Optional Matching (Extended Abstract)*

Shqiponja Ahmetaj, Wolfgang Fischl, Reinhard Pichler, Mantas Šimkus, and Sebastian Skritek

Institute for Information Systems, TU Vienna

1 Introduction

Conjunctive Queries (CQs) constitute the core of most query languages and have been studied intensively in several areas. For querying incomplete data, CQs however suffer one major drawback: they require the complete query to be matched into the data to return answers. One extension of CQs that tries to overcome this problem are *well-designed pattern trees (wdPTs)* [9]. Developed in the context of the Semantic Web, wdPTs are equivalent to a well-behaved fragment of $\{\mathsf{AND}, \mathsf{OPT}\}$-queries of SPARQL [12], and allow a user to retrieve *partial answers* to a query.

Because of this feature, however, wdPTs are nonmonotone. This is problematic for query answering in the presence of implicit knowledge – expressed e.g. by an ontology specified in some Description Logic (DL) – since the usual certain answer semantics turns out to be unsatisfactory in this setting. We observe that the recently released recommendation of the SPARQL entailment regimes [6] provides a semantics exactly for this case. However, it is defined in a simpler and less expressive way than the certain answers semantics, and does not utilize the full potential of the implicit information.

The **goal of this work** is to introduce an intuitive certain answer semantics for the class of well-designed pattern trees under DL-Lite$_\mathcal{R}$ (which provides the theoretical underpinning of the OWL 2 QL entailment regime). After introducing wdPTs, we first discuss some of the problems with an adoption of a certain answer semantics for them and propose a suitable modified definition. We then briefly present results on the complexity of typical reasoning tasks.

Related Work to our findings includes the work our approaches are based upon [3–6]. There is a huge body of results on CQ answering under different DLs (cf. [4, 5, 11, 13]). For SPARQL recent work [8] presents a *stronger* semantics, where entire mappings are discarded, whose possible extensions to optional subqueries would imply inconsistencies in the knowledge base. Further related work includes [2, 7, 10] which is discussed in the long version of this paper.

* A longer version of this paper has been accepted for publication at WWW 2015 [1].

2 DL-Lite$_\mathcal{R}$ and Well-designed Pattern Trees

We assume the reader to be familiar with DL-Lite$_\mathcal{R}$ [4]. A DL-Lite$_\mathcal{R}$ *knowledge base* (KB) is a tuple $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{A}$ is an *ABox* and $\mathcal{T}$ is a *TBox*. The definition of an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is the usual one. In addition, we make the *standard name assumption (SNA)*, i.e. we assume that $\Delta^\mathcal{I}$ contains all individuals, and that $a^\mathcal{I} = a$ for each individual a.

A *well-designed pattern tree* $\mathcal{P}$ is a tuple $(T, \lambda, \boldsymbol{x})$ such that:

1. T is a rooted tree and λ maps each node t in T to a *conjunctive query (CQ)*. A CQ here is a set of *atoms*, where atoms are built as usual, i.e. from concept and role names together with individuals and variables.
2. For every variable y occurring in T, the set of nodes containing y is connected.
3. $\boldsymbol{x}$ is a tuple of variables from T, called the *free variables* of $\mathcal{P}$.

Intuitively, the parent-child relationships in the tree express *optional matching*. I.e., the result of the "parent-CQ" shall be extended by the "child-CQ" if possible — otherwise the child shall be ignored, and only the result of the parent is returned. Finally $\boldsymbol{x}$ are the "output" variables.

A *mapping* μ is any partial function whose domain $\mathsf{dom}(\mu)$ contains only variables. We say a mapping μ_1 is subsumbed by another mapping μ_2, denoted by $\mu_1 \sqsubseteq \mu_2$, if $\mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu_2)$ and $\mu_1(x) = \mu_2(x)$ for all $x \in \mathsf{dom}(\mu_1)$. Also, for a mapping μ and some property A, we shall say that μ is $\sqsubseteq$*-maximal w.r.t.* A if μ satisfies A, and there is no μ' such that $\mu \sqsubseteq \mu'$, $\mu' \not\sqsubseteq \mu$, and μ' satisfies A. For any mapping μ and a tuple of variables $\boldsymbol{x}$, we denote by $\mu_{\boldsymbol{x}}$ the restriction of μ to the variables in $\boldsymbol{x}$.

The notion of a mapping μ that is a *match* for a CQ q in an interpretation $\mathcal{I}$ is defined in the standard way. Assume a wdPT $\mathcal{P} = (T, \lambda, \boldsymbol{x})$ and an interpretation $\mathcal{I}$. For an initial segment T' of T, i.e. a connected subgraph containing the root of T, we define $q_{T'}$ to be the CQ $\bigcup_{t \in T'} \lambda(t)$. Then a *match* for $\mathcal{P}$ in $\mathcal{I}$ is any mapping μ such that μ is a match for $q_{T'}$ in $\mathcal{I}$ for some initial segment T' of T. Let M be the set of all $\sqsubseteq$-maximal matches from $\mathcal{P}$ to $\mathcal{I}$. Then the result of evaluating $\mathcal{P}$ over $\mathcal{I}$, projected to $\boldsymbol{x}$, is the set $[\![\mathcal{P}]\!]_\mathcal{I} = \{\mu_{\boldsymbol{x}} \mid \mu \in M\}$. Note that the order of child nodes in such tress does not affect the query answer (see [9, 12]).

In the following example, we illustrate wdPTs as well as the reason why the usual certain answer semantics (i.e., a tuple is a certain answer if it is present in every model) turns out to be unsatisfactory in our setting:

Example 1. Let $\mathcal{P}$ be the wdPT $(T, \lambda, \boldsymbol{x})$ where T consists of the root r with the single child t, $\lambda(r) = \{\mathsf{teaches}(x, y)\}$, $\lambda(t) = \{\mathsf{knows}(y, z)\}$, and $\boldsymbol{x} = \{x, z\}$. Consider the KB $\mathcal{K}$ consisting of an ABox $\mathcal{A} = \{\mathsf{Prof}(b)\}$, and a TBox $\mathcal{T} = \{(\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches})\}$. Let $\mathcal{I}$ be as follows: $\mathsf{Prof}^\mathcal{I} = \{(b)\}$. Clearly, $\mathcal{I} \models \mathcal{K}$. The query yields in $\mathcal{I}$ as only answer the mapping $\mu = \{x \to b\}$. Clearly, also the interpretation $\mathcal{I}'$, where $\mathsf{Prof}^{\mathcal{I}'} = \{(b)\}$, $\mathsf{teaches}^{\mathcal{I}'} = \{(b, c)\}$ and $\mathsf{knows}^{\mathcal{I}'} = \{(c, d)\}$ is a model of $\mathcal{K}$. But in $\mathcal{I}'$, μ is no longer an answer since μ can be extended to answer $\mu' = \{x \to b, z \to d\}$. Hence, there is no mapping which is an answer in every possible model of $\mathcal{K}$. □

Definition 1. *Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ be a KB and $\mathcal{P} = (T, \lambda, \boldsymbol{x})$ a wdPT. A mapping μ is a* certain answer *to $\mathcal{P}$ over $\mathcal{K}$ if it is a $\sqsubseteq$-maximal mapping s.t. (1) $\mu \sqsubseteq [\![\mathcal{P}]\!]_{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$, and (2) $\mathsf{vars}(q_{T'}) \cap \boldsymbol{x} = \mathsf{dom}(\mu)$ for some initial segment T' of T. We denote by $\mathsf{cert}(\mathcal{P}, \mathcal{K})$ the set of certain answers to $\mathcal{P}$ over $\mathcal{K}$.*

The reason for restricting the set of certain answers to $\sqsubseteq$-maximal mappings is that wdPTs in general may have "subsumed" answers, i.e. mappings s.t. also some proper extension is an answer. But then – with set semantics – we cannot recognize the reason why some subsumed answer is possibly not an answer in some possible world. Therefore, in our first step towards extending CQs by optional matching, we allow only "maximal" answers as certain answers.

Property (2) ensures that the domain of such an answer adheres to the tree structure of the wdPT. However, we can show that this can be enforced in a simple post-processing step. Likewise, also projection can be deferred to a post-processing step. The task is thus to compute a set $\mathsf{certp}(\mathcal{P}, \mathcal{K})$ of *certain pre-answers* (i.e., mappings that satisfy Definition 1 except property (2), ignoring projection), which can be done via the *canonical model*. For a given KB $\mathcal{K}$, we assume a canonical model of $\mathcal{K}$, denoted as $\mathsf{can}(\mathcal{K})$, to be defined as in [4].

Theorem 1. *Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ be a KB and $\mathcal{P}$ a wdPT. Then, $\mathsf{certp}(\mathcal{P}, \mathcal{K}) = \mathrm{MAX}([\![\mathcal{P}]\!]_{\mathsf{can}(\mathcal{K})}\downarrow)$, where $\mathrm{MAX}(M)$ is the set of $\sqsubseteq$-maximal mappings in M, $M\downarrow := \{\mu\downarrow \mid \mu \in M\}$ ($\mu\downarrow$ is the restriction of μ to those variables which are mapped to the individual names that occur in $\mathcal{A}$).*

To cope with the potential infinite canonical model, query rewriting algorithms have been developed in the literature. By introducing several adaptations and extensions of the rewriting-based CQ evaluation for DL-Lite from [4], we develop two different algorithms to answer wdPTs over DL-Lite$_{\mathcal{R}}$ KBs. [1] Based on these rewriting algorithms, we analyze the complexity of query answering and of several static query analysis tasks such as query containment and equivalence. We are able to show that the additional power of our new semantics comes without additional costs in terms of complexity.

For future work, we want to investigate further more expressive DLs under our certain answer semantics. The implementation of the rewriting algorithms and the development of suitable benchmarks, is a challenging task as well. Additionally, we will extend our work to allow TBox queries and other fragments of SPARQL.

Acknowledgements

This work was supported by the Vienna Science and Technology Fund (WWTF), project ICT12-15 and by the Austrian Science Fund (FWF): P25207-N23 and W1255-N23.

[1] Note that, in the full version we consider a fragment of well-designed SPARQL under OWL 2 QL entailment, which corresponds exactly to what we consider here.

References

1. S. Ahmetaj, W. Fischl, R. Pichler, M. Šimkus, and S. Skritek. Towards reconciling SPARQL and certain answers, 2014. Accepted for publication, WWW 2015.
2. M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In *Proc. of PODS 2014*, pages 14–26. ACM, 2014.
3. M. Arenas and J. Pérez. Querying semantic web data with SPARQL. In *Proc. of PODS 2011*, pages 305–316. ACM, 2011.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
5. T. Eiter, M. Ortiz, M. Šimkus, T. Tran, and G. Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI 2012*. AAAI Press, 2012.
6. B. Glimm and C. Ogbuji. SPARQL 1.1 Entailment Regimes. W3C Recommendation, W3C, Mar. 2013. `http://www.w3.org/TR/sparql11-entailment`.
7. R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of ISWC 2014*, pages 552–567. Springer, 2014.
8. E. V. Kostylev and B. C. Grau. On the semantics of SPARQL queries with optional matching under entailment regimes. In *Proc. of ISWC 2014*, pages 374–389. Springer, 2014.
9. A. Letelier, J. Pérez, R. Pichler, and S. Skritek. Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.*, 38(4):25, 2013.
10. L. Libkin. Incomplete data: what went wrong, and how to fix it. In *Proc. PODS 2014*, pages 1–13. ACM, 2014.
11. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.
12. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.
13. R. Rosati. On conjunctive query answering in EL. In *Proc. DL 2007*. CEUR-WS.org, 2007.

Dealing with Inconsistencies due to Class Disjointness in SPARQL Update

Albin Ahmeti[1,3], Diego Calvanese[2], Axel Polleres[3], and Vadim Savenkov[3]

[1] Vienna University of Technology, Favoritenstraße 9, 1040 Vienna, Austria
[2] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
[3] Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Abstract. The problem of updating ontologies has received increased attention in recent years. In the approaches proposed so far, either the update language is restricted to (sets of) atomic updates, or, where the full SPARQL Update language is allowed, the TBox language is restricted to RDFS where no inconsistencies can arise. In this paper we discuss directions to overcome these limitations. Starting from a DL-Lite fragment covering RDFS and concept/class disjointness axioms, we define two semantics for SPARQL Update: under cautious semantics, inconsistencies are resolved by rejecting *all* updates potentially introducing conflicts; under brave semantics, instead, conflicts are overridden in favor of new information where possible. The latter approach builds upon existing work on the evolution of DL-Lite knowledge bases, setting it in the context of generic SPARQL updates.

1 Introduction

This paper initiates the study of SPARQL updates in the context of potential inconsistencies: as a minimalistic ontology language allowing for inconsistencies, we consider RDFS$_\neg$, an extension of RDFS [8] with *class disjointness axioms* of the form $\{P$ `disjointWith` $Q\}$ from OWL [10], sometimes referred to as *negative inclusions* or NIs [4], as the corresponding description logic encoding of this statement is $P \sqsubseteq \neg Q$.

As a running example, we assume a triple store G with an RDFS$_\neg$ ontology (TBox) $\mathcal{T}$ encoding an educational domain, asserting a range restriction plus mutual disjointness of the concepts like professor and student (we use Turtle syntax [2], in which dw abbreviates OWL's `disjointWith` keyword, and dom and rng respectively stand for the `domain` and `range` keywords of RDFS).

$\mathcal{T} = \{$`:studentOf` dom `:Student. :studentOf` rng `:Professor. :Professor` dw `:Student.` $\}$

Consider the following SPARQL update [6] request u in the context of the TBox $\mathcal{T}$:

```
INSERT {?X :studentOf ?Y} WHERE  {?X :attendsClassOf ?Y}
```

Consider an ABox with data on student tutors that happen to attend each other's classes: $\mathcal{A}_1$ = {`:jimmy :attendsClassOf :ann. :ann :attendsClassOf :jimmy`}. Here, u would create two assertions `:jimmy :studentOf :ann` and `:ann :studentOf :jimmy`. Due to the range and domain constraints in $\mathcal{T}$, these assertions result in clashes both for Jimmy and for Ann. Note that all inconsistencies

Table 1. *DL-Lite*$_{\text{RDFS}_\neg}$ assertions vs. RDF(S), where A, A' denote concept (or, class) names, P, P' denote role (or, property) names, Γ is the set of IRI constants (excl. the OWL/RDF(S) vocabulary) and $x, y \in \Gamma$. For RDF(S), we use abbreviations (rsc, sp, dom, rng, a) as introduced in [11].

TBox	RDFS$_\neg$	TBox	RDFS$_\neg$	TBox	RDFS$_\neg$	ABox	RDFS$_\neg$
1. $A' \sqsubseteq A$	A' sc A.	3. $\exists P \sqsubseteq A$	P dom A.	5. $A' \sqsubseteq \neg A$	A' dw A.	6. $A(x)$	x a A.
2. $P' \sqsubseteq P$	P' sp P.	4. $\exists P^- \sqsubseteq A$	P rng A.			7. $P(x, y)$	x P y.

are in the new data, and thus we say that u is *intrinsically inconsistent* for the particular ABox $\mathcal{A}_1$. Our solution for such updates will be to discard problematic assignments but keep those that cause no clashes.

Now, let $\mathcal{A}_2$ be the ABox {`:jimmy :attendsClassOf :ann. :jimmy a :Professor`}. It is clear that after the update u, the ABox will become inconsistent, due to the property assertion `:jimmy :studentOf :ann`, implying that Jimmy is both a professor and a student which contradicts the TBox disjointness axiom. In contrast to the previous case, the clash now is between the prior knowledge and the new data. We propose *two update semantics*, extending our previous work [1] for dealing with such inconsistencies and provide *rewriting algorithms* for implementing them using the basic constructs of the SPARQL language (e.g., making use of the UNION, MINUS, FILTER, and OPTIONAL operators).

In the remainder of the paper, after some short preliminaries (Sec. 2) we discuss checking of intrinsic inconsistencies in Sec. 3, and then in Sec. 4 we present two semantics for dealing with general inconsistencies in the context of materialized triple stores. An overview of related work and concluding remarks can be found in Sec. 5.

2 Preliminaries

We introduce basic notions about RDF graphs, RDFS$_\neg$ ontologies, and SPARQL queries. In this paper we use RDF and DL notation interchangeably, treating RDF graphs that do not use non-standard RDFS$_\neg$ vocabulary [12] as sets of TBox and ABox assertions.

Definition 1 (RDFS$_\neg$ ABox, TBox, triple store). *We call a set $\mathcal{T}$ of inclusion assertions of the forms 1–5 in Table 1 an* (RDFS$_\neg$) TBox*, a set $\mathcal{A}$ of assertions of the forms 6–7 in Table 1 an* (RDF) ABox*, and the union $G = \mathcal{T} \cup \mathcal{A}$ an* (RDFS$_\neg$) triple store*.*

Definition 2 (Interpretation, satisfaction, model, consistency). *An* interpretation $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *consists of a non-empty set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, which maps*
- *each atomic concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$,*

$$\frac{?C \text{ sc } ?D. \quad ?S \text{ a } ?C.}{?S \text{ a } ?D.} \qquad \frac{?P \text{ dom } ?C. \quad ?S\ ?P\ ?O.}{?S \text{ a } ?C.}$$

$$\frac{?P \text{ sp } ?Q. \quad ?S\ ?P\ ?O.}{?S\ ?Q\ ?O.} \qquad \frac{?P \text{ rng } ?C. \quad ?S\ ?P\ ?O.}{?O \text{ a } ?C.} \qquad \frac{?S \text{ a } ?C,?D. \quad ?C \text{ dW } ?D.}{\bot}$$

Fig. 1. Minimal RDFS rules from [11] plus class disjointness "clash" rule from OWL2 RL [10].

– *each negation of atomic concept A to $(\neg A^{\mathcal{I}}) = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$,*
– *each atomic role P to a binary relation $P^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$, and*
– *each element of Γ to an element of $\Delta^{\mathcal{I}}$.*

For expressions $\exists P$ and $\exists P^-$, the interpretation function is defined as $(\exists P)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in P^{\mathcal{I}}\}$ resp. $(\exists P^-)^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in P^{\mathcal{I}}\}$. An interpretation $\mathcal{I}$ satisfies *an inclusion assertion $E_1 \sqsubseteq E_2$ (of one of the forms 1–5 in Table 1), if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$. Analogously, $\mathcal{I}$* satisfies *ABox assertions of the form $A(x)$, if $x^{\mathcal{I}} \in A^{\mathcal{I}}$, and of the form $P(x,y)$, if $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in P^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is called a* model *of a triple store G (resp., a TBox $\mathcal{T}$, an ABox $\mathcal{A}$), denoted $\mathcal{I} \models G$ (resp., $\mathcal{I} \models \mathcal{T}$, $\mathcal{I} \models \mathcal{A}$), if $\mathcal{I}$ satisfies all assertions in G (resp., $\mathcal{T}$, $\mathcal{A}$). Finally, G is called consistent, if it does not entail both $C(x)$ and $\neg C(x)$ for any concept C and constant $x \in \Gamma$, where entailment is defined as usual.*

As in [1], we treat only restricted SPARQL queries corresponding to (unions of) conjunctive queries without non-distinguished variables over DL ontologies:

Definition 3 (BGP, CQ, UCQ, query answer). *A* conjunctive query *(CQ) q, or* basic graph pattern (BGP)*, is a set of atoms of the form 6–7 from Table 1, where now $x, y \in \Gamma \cup \mathcal{V}$.*[4] *A* union of conjunctive queries *(UCQ) Q, or* UNION *pattern, is a set of CQs. We denote with $\mathcal{V}(q)$ (or $\mathcal{V}(Q)$) the set of variables from $\mathcal{V}$ occurring in q (resp., Q). An* answer *(under RDFS$_\neg$ Entailment) to a CQ q over a triple store G is a substitution θ of the variables in $\mathcal{V}(q)$ with constants in Γ such that every model of G satisfies all facts in $q\theta$. We denote the set of all such answers with $ans_{\text{rdfs}}(q, G)$ (or simply $ans(q, G)$). The set of answers to a UCQ Q is $\bigcup_{q \in Q} ans(q, G)$.*

We also recall from [1], that query answering in the presence of ontologies can be done either by rule-based pre-materialization of the ABox or by query rewriting. Let $rewrite(q, \mathcal{T})$ be the UCQ resulting from applying $PerfectRef$ [3] (or, equivalently, the stripped-down version from [1, Alg.1]) to a query q and let $G = \mathcal{T} \cup \mathcal{A}$ be a triple store. Furthermore, let $mat(G)$ be the triple store obtained from exhaustive application of the inference rules in Fig. 1 on a consistent triple store G, and—analogously—let $chase(q, \mathcal{T})$ refer to "materialization" w.r.t. $\mathcal{T}$ applied to a CQ q. The next result transfers from [1] to consistent RDFS$_\neg$ stores.

Proposition 1. *Let $G = \mathcal{T} \cup \mathcal{A}$ be a consistent triple store, and q a CQ. Then, $ans(q, G) = ans(rewrite(q, \mathcal{T}), \mathcal{A}) = ans(q, mat(G))$.*

We have used this previously to define the semantics of SPARQL update operations.

Definition 4 (SPARQL update operation, simple update of a triple store). *Let P_d and P_i be BGPs, and P_w a BGP or* UNION *pattern. Then an* update operation *$u(P_d, P_i, P_w)$ has the form*

DELETE P_d **INSERT** P_i **WHERE** P_w

Let $G = \mathcal{T} \cup \mathcal{A}$ be a triple store. Then the *simple update* of G w.r.t. $u(P_d, P_i, P_w)$ is defined as $G_{u(P_d,P_i,P_w)} = (G \setminus \mathcal{A}_d) \cup \mathcal{A}_i$, where $\mathcal{A}_d = \bigcup_{\theta \in ans(P_w, G)} gr(P_d\theta)$, $\mathcal{A}_i = \bigcup_{\theta \in ans(P_w, G)} gr(P_i\theta)$, and $gr(P)$ denotes the set of ground triples in pattern P.

[4] $\mathcal{V}$ is a countably infinite set of variables (written as '?'-prefixed alphanumeric strings).

For the sake of readability of the algorithms, we assume that all update operations $u(P_d, P_i, P_w)$ in this paper *contain no constants* in the BGPs P_d and P_i, and that all property assertions $(?X \ \mathsf{p} \ ?Y)$ in P_d have distinct variables $?X$ and $?Y$. Enhancing our results to updates with constants and variable equalities in P_d and P_i is not difficult, but requires distinguishing special cases: e.g., instead of replacing the variable y in a pattern Q by z, the expression Q FILTER$(y = z)$ can be used in the case when y is a constant; instead of $Q(y)$ MINUS P for a variable y, Q FILTER NOT EXISTS P should be used for ground Q.

We call a triple store or (ABox) *materialized* if in each state it always guarantees $G \backslash \mathcal{T} = mat(G) \backslash mat(\mathcal{T})$. In the present paper, we will always focus on "materialization preserving" semantics for SPARQL update operations, which we dubbed $\mathbf{Sem}_2^{mat}$ in [1] and which preserves a materialized triple store. We recall the intuition behind $\mathbf{Sem}_2^{mat}$, given an update $u = (P_d, P_i, P_w)$: *(i)* delete the instantiations of P_d *plus all their causes*; *(ii)* insert the instantiations of P_i *plus all their effects*.

Definition 5 ($\mathbf{Sem}_2^{mat}$ [1]). *Let $u(P_d, P_i, P_w)$ be an update operation. Then*

$$G^{\mathbf{Sem}_2^{mat}}_{u(P_d,P_i,P_w)} = G_{u(P_d^{\text{caus}}, P_i^{\text{eff}}, \{P_w\}\{P_d^{fvars}\})}$$

Here, given a pattern P, $P^{\text{caus}} = flatten(rewrite(P, \mathcal{T}))$; $P^{\text{eff}} = chase(P, \mathcal{T})$ and $P^{fvars} = \{?v \ \mathsf{a} \ \texttt{rdfs:Resource} \mid ?v \in \mathcal{V}(P^{caus}) \setminus \mathcal{V}(P)\}$, where $flatten(\cdot)$ computes the set of all triples occurring in the UCQ $rewrite(P, \mathcal{T})$, which in our case allows us to obtain all possible "causes" of each atom in P_d, and "$?v$ a rdfs:Resource" is a shortcut for a pattern that binds $?v$ to any $x \in \Gamma$ occurring in G.

We refer to [1] for further details, but stress that as such, $\mathbf{Sem}_2^{mat}$ is not able to detect or deal with inconsistencies arising from P_i and G. In what follows, we will discuss how this can be remedied.

3 Checking Consistency of a SPARQL Update

Within previous work on the evolution of DL-Lite knowledge bases [4], updates given in the form of pairs of ABoxes $\mathcal{A}_d, \mathcal{A}_i$ have been studied. However, whereas such update might be viewed to fit straightforwardly to the corresponding $\mathcal{A}_d, \mathcal{A}_i$ in Def. 4, in [4] it is assumed that $\mathcal{A}_i$ is consistent with the TBox, and thus one only needs to consider how to deal with inconsistencies between the update and the old state of the knowledge base. This a priori assumption may be insufficient for SPARQL updates though, where concrete values for inserted triples are obtained from variable bindings in the WHERE clause, and depending on the bindings, the update can be either consistent or not. This is demonstrated by the update u from Sec. 1 which, when applied to the ABox $\mathcal{A}_4$, results in an inconsistent set $\mathcal{A}_i$ of insertions . We call this *intrinsic inconsistency* of an update *relative to a triple store $G = \mathcal{T} \cup \mathcal{A}$.*

Definition 6. *Let G be a triple store. The update u is said to be* intrinsically consistent w.r.t. *G if the set of new assertions $\mathcal{A}_i$ from Def. 4 generated by applying u to G, taken in isolation from the ABox of G, does not contradict the TBox of G. Otherwise, the update is said to be intrinsically inconsistent w.r.t. G.*

Algorithm 1: constructing a SPARQL ASK query to check intrinsic inconsistency (for the definition of P_i^{eff}, cf. Def. 5)

Input: RDFS$_\neg$ TBox $\mathcal{T}$, SPARQL update $u(P_d, P_i, P_w)$
Output: A SPARQL ASK query returning $True$ if u is intrinsically inconsistent

```
if ⊥ ∈ P_i^eff then
    return ASK {}    //u contains clashes in itself, i.e., is inconsistent for any triple store
else
    W := { FILTER(False)};    //neutral element w.r.t. union
    foreach pair of triple patterns (?X a P), (?Y a R) in P_i^eff do
        if P ⊑ ¬R ∈ T then
            W := W UNION {{P_w θ1[?X ↦ ?Z]} . {P_w θ2[?Y ↦ ?Z]}} for a fresh ?Z
    return ASK WHERE {W}
```

Intrinsic inconsistency of the update differs crucially from the inconsistency w.r.t. the old state of the knowledge base, illustrated by the ABox $\mathcal{A}_2$ from Sec. 1. This latter case can be addressed by adopting an update policy that prefers newer assertions in case of conflicts, as studied in the context of DL-Lite KB evolutions [4], which we will discuss in Sec. 4 below. Intrinsic inconsistencies however are harder to deal with, since there is no cue which assertion should be discarded in order to avoid the inconsistency. Our proposal here is thus to discard *all* mutually inconsistent pairs of insertions.

We first present an algorithm for checking intrinsic inconsistency by means of SPARQL ASK queries and then a safe rewriting algorithm. This rewriting is based on an observation that clashing triples can be introduced by a combination of two bindings of variables in the WHERE clause, as the example in the Sec. 1 (the ABox $\mathcal{A}_1$) illustrates. To handle such cases, two copies of the WHERE clause P_w are created by the rewriting in Algorithms 1 and 2, for each pair of disjoint concepts according to the TBox of the triple store. These algorithms use notation described in Rem. 1 below.

Remark 1. Our rewriting algorithms rely on producing fresh copies of the WHERE clause. Assume θ, θ_1, θ_2, ... to be substitutions replacing each variable in a given formula with a distinct fresh one. For a substitution σ, we also define $\theta[\sigma]$ resp. $\theta_i[\sigma]$ to be an extension of σ, renaming each variable at positions not affected by σ with a distinct fresh one. For instance, let F be a triple $(?Z$ `:studentOf` $?Y)$. Now, $F\theta$ makes a variable disjoint copy of F: $?Z_1$ `:studentOf` $?Y_1$ for fresh $?Z_1, ?Y_1$. $F[?Z \mapsto ?X]$ is just a substitution of $?Z$ by $?X$ in F. Finally, $F\theta[?Z \mapsto ?X]$ results in $?X$ `:studentOf` $?Y_2$ for fresh $?Y_2$. We assume that all occurrences of $F\theta[\sigma]$ stand for syntactically the same query, but that $F\theta[\sigma_1]$ and $F\theta[\sigma_2]$, for distinct σ_1 and σ_2, can only have variables in $range(\sigma_1) \cap range(\sigma_2)$ in common. That is, the choice of fresh variables is defined by the parameterizing substitution σ. ■

Now, the possibility of unifying two variables $?X$ and $?Y$ in P_w on a given triple store can be tested with the query $\{P_w\theta_1[?X \mapsto ?Z]\}\{P_w\theta_2[?Y \mapsto ?Z]\}$ where θ_1 and θ_2 are variable renamings as in Rem. 1 and $?Z$ is a fresh variable.

In order to check the intrinsic consistency of an update, this condition should be evaluated for every pair of variables of P_w, the unification of which leads to a clash. A SPARQL ASK query based on this idea is produced by Alg. 1. Note that it suffices to

Algorithm 2: Safe rewriting $safe(u)$

Input: RDFS$_\neg$ TBox $\mathcal{T}$, SPARQL update $u(P_d, P_i, P_w)$
Output: SPARQL update $safe(u)$

1 **if** $\bot \in P_i^{\text{eff}}$ **then**
2 | **return** $u(P_d, P_i, \mathsf{FILTER}(\mathit{False}))$
3 $W := \{\mathsf{FILTER}(\mathit{False})\}$; *//neutral element w.r.t. union*
4 **foreach** *pair of triple patterns* $(?X\ \mathsf{a}\ P),\ (?Y\ \mathsf{a}\ R)$ *in* P_i^{eff} **do**
5 | **if** $P \sqsubseteq \neg R \in \mathcal{T}$ **then**
6 | | *//cf. Rem. 1 for notation* $\theta[\ldots]$
7 | | $W := W\ \mathsf{UNION}\ \{P_w\theta_1[?X \mapsto ?Y]\}\ \mathsf{UNION}\ \{P_w\theta_2[?Y \mapsto ?X]\}\}$
8 **return** $u(P_d, P_i, P_w\ \mathsf{MINUS}\ \{W\})$

check only triples of the form $\{?X\ \mathsf{a}\ ?C\}$ at line 5 of Alg. 1, since disjointness conditions can only be formulated for concepts, according to the syntax in Table 1. Furthermore, since we are taking the facts in P_i^{eff} extended by all facts implied by $\mathcal{T}$, at line 6 of Alg. 1 it suffices to check the disjointness conditions explicitly mentioned in $\mathcal{T}$ and not all those which are implied by $\mathcal{T}$.

Example 1. Consider the update u from Sec. 1, in which the INSERT clause P_i can create clashing triples. To identify potential clashes, Alg. 1 first applies the inference rule for the range constraint, and computes $P_i^{\text{eff}} = \{?X$ `a :Student` $. ?Y$ `a :Professor`$\}$. Now both variables $?X, ?Y$ occur in the triples of type (6) from Table 1 with clashing concept names. The following ASK query is produced by Alg. 1.

```
ASK WHERE { ?X :attendsClassOf ?Y . ?Y :attendsClassOf ?X1 }
```

(In this and subsequent examples we omit the trivial $\mathsf{FILTER}(\mathit{False})$ union branch used in rewritings to initialize variables with disjunctive conditions, such as W in Alg. 1) ■

Suppose that an insert is not intrinsically consistent for a given triple store. One solution would be to discard it completely, should the above ASK query return True. Another option which we consider here is to only discard those variable bindings from the WHERE clause, which make the INSERT clause P_i inconsistent. This is the task of the *safe rewriting* $safe(\cdot)$ in Alg. 2, removing all variable bindings that participate in a clash between different triples of P_i. Let P_w be a WHERE clause, in which the variables $?X$ and $?Y$ should not be unified to avoid clashes. With θ_1, θ_2 being "fresh" variable renamings as in Rem. 1, Alg. 2 uses the union of $P_w\theta_1[?X \mapsto ?Y]$ and $P_w\theta_2[?Y \mapsto ?X]$ to eliminate unsafe bindings that send $?X$ and $?Y$ to a same value.

Example 2. Alg. 2 extends the WHERE clause of the update u from Sec. 1 as follows:

```
INSERT{?X :studentOf ?Y} WHERE{?X :attendsClassOf ?Y
 MINUS{{?X1 :attendsClassOf ?X} UNION {?Y :attendsClassOf ?Y2}}}
```

Note that the safe rewriting can make the update void. For instance, $safe(u)$ has no effect on the ABox $\mathcal{A}_1$ from Sec. 1, since there is no cue, which of `:jimmy :attendsClassOf :ann`, `:ann :attendsClassOf :jimmy` needs to be dismissed to avoid the clash. However, if we extend this ABox with assertions both satisfying the WHERE clause of u and not causing undesirable variable unifications, $safe(u)$

would make insertions based on such bindings. For instance, adding the fact `:bob :attendsClassOf :alice` to $\mathcal{A}_1$ would assert `:bob :studentOf :alice` as a result of $safe(u)$. ■

A rationale for using MINUS rather than FILTER NOT EXISTS in Alg. 2 (and also in a rewriting in forthcoming Sec. 4) can be illustrated by an update in which variables in the INSERT and DELETE clauses are bound in different branches of a UNION:

```
DELETE {?V a :Professor} INSERT {?X :studentOf ?Y}
WHERE {{?X :attendsClassOf ?Y} UNION {?V :attendsClassOf ?W}}
```

A safe rewriting of this update (abbreviating `:attendsClassOf` as `:aCo`) is

```
DELETE {?V a :Professor} INSERT {?X :studentOf ?Y}
WHERE { {{?X :aCo ?Y} UNION {?V :aCo ?W}}
          MINUS{ {{?X1 :aCo ?X} UNION {?V1 :aCo ?W1}}
           UNION {{?Y :aCo ?Y2} UNION {?V2 :aCo ?W2}} } }
```

It can be verified that with FILTER NOT EXISTS in place of MINUS this update makes no insertions on all triple stores: the branches `{?V1 :aCo ?W1}` and `{?V2 :aCo ?W2}` are satisfied whenever `{?X :aCo ?Y}` is, making FILTER NOT EXISTS evaluate to *False* whenever `{?X :aCo ?Y}` holds.

We conclude this section by formalizing the intuition of update safety. For a triple store G and an update $u = (P_d, P_i, P_w)$, let $[\![P_w]\!]^u_G$ denote the set of variable bindings computed by the query "SELECT $?X_1, \ldots, ?X_k$ WHERE P_w" over G, where $?X_1, \ldots, ?X_k$ are the variables occurring in P_i or in P_d.

Theorem 1. *Let $\mathcal{T}$ be a TBox, let u be a SPARQL update (P_i, P_d, P_w), and let query q_u and update $safe(u) = (P_d, P_i, P'_w)$ result from applying Alg. 1 resp. Alg. 2 to u and $\mathcal{T}$. Then, the following properties hold for an arbitrary RDFS$_\neg$ triple store $G = \mathcal{T} \cup \mathcal{A}$:*
(1) $q_u(G) =$ True iff $\exists \mu, \mu' \in [\![P_w]\!]^u_G$ s.t. $\mu(P_i) \wedge \mu'(P_i) \wedge \mathcal{T} \models \bot$;
(2) $[\![P_w]\!]^u_G \setminus [\![P'_w]\!]^u_G = \{\mu \in [\![P_w]\!]^u_G \mid \exists \mu' \in [\![P_w]\!]^u_G$ s.t. $\mu(P_i) \wedge \mu'(P_i) \wedge \mathcal{T} \models \bot\}$.

4 Materialization Preserving Update Semantics

In this section we discuss resolution of inconsistencies between triples already in the triple store and newly inserted triples. Our baseline requirement for each update semantics is formulated as the following property.

Definition 7 (Consistency-preserving). *Let G be a triple store and $u(P_d, P_i, P_w)$ an update. A materialization preserving update semantics Sem is called* consistency preserving *in RDFS$_\neg$ if the evaluation of update u, i.e., $G^{Sem}_{u(P_d,P_i,P_w)}$, results in a consistent triple store.*

Our two consistency preserving semantics are respectively called *brave* and *cautious*. The brave semantics always gives priority to newly inserted triples by discarding all pre-existing information that contradicts the update. The cautious semantics is exactly the opposite, discarding inserts that are inconsistent with facts already present in the triple store; i.e., the cautious semantics never deletes facts unless explicitly required by the DELETE clause of the SPARQL update.

Algorithm 3: *Brave semantics* $\mathbf{Sem}^{mat}_{brave}$

Input: Materialized triple store $G = \mathcal{T} \cup \mathcal{A}$, SPARQL update $u(P_d, P_i, P_w)$
Output: $G^{\mathbf{Sem}^{mat}_{brave}}_{u(P_d,P_i,P_w)}$

1 $P'_d := P_d^{\text{caus}}$;
2 **foreach** *triple pattern* $(?X \text{ a } C)$ *in* P_i^{eff} **do**
3 **foreach** C' *s.t.* $C \sqsubseteq \neg C' \in \mathcal{T}$ *or* $C' \sqsubseteq \neg C \in \mathcal{T}$ **do**
4 **if** $(?X \text{ a } C') \notin P'_d$ **then**
5 $P'_d := P'_d \,.\, \{?X \text{ a } C'\}^{\text{caus}}$
6 **return** $G_{u(P'_d, P_i^{\text{eff}}, \{P_w\} P_d^{fvars})}$

Both semantics rely upon incremental update semantics $\mathbf{Sem}_2^{mat}$, introduced in Sec. 2, which we aim to extend to take into account class disjointness. Note that for the present section we assume updates to be intrinsically consistent, which can be checked or enforced beforehand in a preprocessing step by the safe rewriting discussed in Sec. 3. In this section, we lift our definition of update operation to include also updates (P_d, P_i, P_w) with P_w produced by the safe rewriting Alg. 2 from some update satisfying Def. 4. What remains to be defined is the handling of clashes between newly inserted triples and triples already present in the triple store.

The intuitions of our semantics for a SPARQL update $u(P_d, P_i, P_w)$ in the context of an RDFS$_\neg$ TBox are as follows:

- *brave semantics* $\mathbf{Sem}^{mat}_{brave}$: *(i)* delete all instantiations of P_d and their causes, *plus all the non-deleted triples in G clashing with instantiations of triples in P_i to be inserted*, again also including the causes of these triples; *(ii)* insert the instantiations of P_i plus all their effects.
- *cautious semantics* $\mathbf{Sem}^{mat}_{caut}$: *(i)* delete all instantiations of P_d and their causes; *(ii)* insert all instantiations of P_i plus all their effects, *unless they clash with some non-deleted triples in G*: in this latter case, perform neither deletions nor insertions.

For a SPARQL update u, we will define rewritings of u implementing the above semantics, which can be shown to be materialization preserving and consistency preserving.

4.1 Brave Semantics

The rewriting in Alg. 3 implements the brave update semantics $\mathbf{Sem}^{mat}_{brave}$; it can be viewed as combining the idea of *FastEvol* [4] with $\mathbf{Sem}_2^{mat}$ to handle inconsistencies by giving priority to triples that ought to be inserted, and deleting all those triples from the store that clash with the new ones.

The DELETE clause P'_d of the rewritten update is initialized with the set P_d of triples from the input update u. Rewriting ensures that also all "causes" of deleted facts are removed from the store, since otherwise deleted triples will be re-inserted by materialization. To this end, line 1 of Alg. 3 adds to P'_d all facts from which P_d can be derived. Then, for each triple implied by P_i (that is, for each triple in P_i^{eff}) the algorithm computes clashing patterns and adds them to the DELETE clause P'_d, along with their causes. Note that it suffices to only consider disjointness assertions that are syntactically

contained in $\mathcal{T}$ (and not all that are implied by $\mathcal{T}$), since we assume that the triple store is materialized.

Finally, the WHERE clause of the rewritten update is extended to satisfy the syntactic restriction that all variables in P'_d must have matches in the WHERE clause: bindings of "fresh" variables introduced to P'_d by eventual domain or range constraints are provided by the part P_d^{fvars}, cf. Def. 5 and Ex. 3 below. The rewritten update is evaluated over the triple store, computing its new materialized and consistent state.

Example 3. Ex. 2 in Sec. 3 provided a safe rewriting $safe(u)$ of the update u from Sec. 1. According to Alg. 3, this safe update is rewritten to:

```
DELETE {?X a :Professor . ?X1 :studentOf ?X .
        ?Y a :Student . ?Y :studentOf ?Y1}
INSERT {?X :studentOf ?Y . ?X a :Student . ?Y a :Professor}
WHERE {{?X :attendsClassOf ?Y
  MINUS{{?X2 :attendsClassOf ?X} UNION {?Y :attendsClassOf ?Y2}}}
  OPTIONAL {?X1 :studentOf ?X} OPTIONAL {?Y :studentOf ?Y1} }
```

The DELETE clause removes potential clashes for the inserted triples. Note that also property assertions implying clashes need to be deleted, and the respective triples in P'_d contain fresh variables $?X1$ and $?Y1$. These variables have to be bound in the WHERE clause, and therefore P_d^{fvars} adds two optional clauses to P_w of $safe(u)$, which is a computationally reasonable implementation of the concept P^{fvars} from Def. 5. ■

Theorem 2. *Alg. 3, given a SPARQL update u and a consistent materialized triple store $G = \mathcal{T} \cup \mathcal{A}$, computes a new consistent and materialized state w.r.t.* brave semantics.

4.2 Cautious Semantics

Unlike $\mathbf{Sem}^{mat}_{brave}$, its *cautious* version $\mathbf{Sem}^{mat}_{caut}$ always gives priority to triples that are already present in the triple store, and dismisses any inserts that are inconsistent with it. We implement this semantics as follows: *(i)* the DELETE command does not generate inconsistencies and thus is assumed to be always possible; *(ii)* the update is actually executed only if the triples introduced by the INSERT clause do not clash with state of the triple graph *after all deletions have been applied*.

Cautious semantics thus treats insertions and deletions asymmetrically: the former depend on the latter but not the other way round. The rationale is that deletions never cause inconsistencies and can remove clashes between the old and the new data.

As in the case of brave semantics, cautious semantics is implemented using rewriting, presented in Alg. 4. First, the algorithm issues an ASK query to check that no clashes will be generated by the INSERT clause, provided that the DELETE part of the update is executed. If no clashes are expected, in which case the ASK query returns $False$, the brave update from the previous section is applied.

For a safe update $u = (P_d, P_i, P_w)$, the ASK query is generated as follows. For each triple pattern $\{?X \texttt{ a } C\}$ among the effects of P_i, at line 3 Alg. 4 enumerates all concepts C' that are explicitly mentioned as disjoint with C in $\mathcal{T}$. As in the case of brave semantics, this syntactic check is sufficient due to the assumption that the update is applied to a materialized store; by the same reason also no property assertions need to be taken into account.

Algorithm 4: *Cautious semantics* $\mathbf{Sem}^{mat}_{caut}$

Input: Materialized triple store $G = \mathcal{T} \cup \mathcal{A}$, SPARQL update $u(P_d, P_i, P_w)$
Output: $G^{\mathbf{Sem}^{mat}_{caut}}_{u(P_d,P_i,P_w)}$

1 $W := \{\,\mathsf{FILTER}(\mathit{False})\}$ *// neutral element w.r.t. union*
2 **foreach** $(?X\ \mathsf{a}\ C) \in P_i^{\text{eff}}$ **do**
3 **foreach** C' *s.t.* $C \sqsubseteq \neg C' \in \mathcal{T}$ *or* $C' \sqsubseteq \neg C \in \mathcal{T}$ **do**
4 $\Theta^{-}_{C'} := \{\,\mathsf{FILTER}(\mathit{False})\}$
5 **foreach** $(?Y\ \mathsf{a}\ C') \in P_d^{\text{caus}}$ **do**
6 $\Theta^{-}_{C'} := \Theta^{-}_{C'}\ \mathsf{UNION}\ \{P_w\theta[?Y \mapsto ?X]\}$
7 $W := W\ \mathsf{UNION}\ \{\{?X\ \mathsf{a}\ C'\}\ \mathsf{MINUS}\ \{\Theta^{-}_{C'}\}\}$
8 $Q := \mathsf{ASK\ WHERE}\ \{\{P_w\}.\{W\}\}$;
9 **if** $Q(G)$ **then**
10 **return** G
11 **else**
12 **return** $G^{\mathbf{Sem}^{mat}_{brave}}_{u(P_d,P_i,P_w)}$

For each concept C' disjoint from C, we need to check that a triple matching the pattern $\{?X\ \mathsf{a}\ C'\}$ is in the store G and will not be deleted by u. Deletion happens if there is a pattern $\{?Y\ \mathsf{a}\ C'\} \in P_d^{\text{caus}}$ such that the variable $?Y$ can be bound to the same value as $?X$ in the WHERE clause P_w. Line 6 of Alg. 4 produces such a check, using a copy of P_w, in which the variable $?Y$ is replaced by $?X$ and all other variables are replaced with distinct fresh ones. Since there can be several such triple patterns in P_d^{caus}, testing for clash elimination via the DELETE clause requires a disjunctive graph pattern $\Theta^{-}_{C'}$ constructed at line 6 and combined with $\{?X\ \mathsf{a}\ C'\}$ using MINUS at line 7.

Finally, the resulting pattern is appended to the list W of clash checks using UNION. As a result, $\{P_w\}.\{W\}$ queries for triples that are not deleted by u and clash with an instantiation of some class membership assertion $\{?X\ \mathsf{a}\ C\} \in P_i^{\text{eff}}$.

Theorem 3. *Alg. 4, given a SPARQL update* u *and a consistent materialized triple store* $G = \mathcal{T} \cup \mathcal{A}$*, computes a new consistent and materialized state w.r.t.* cautious semantics.

Example 4. Alg. 4 rewrites the safe update $\mathit{safe}(u)$ from Ex. 2 as follows:

```
ASK WHERE{{?X :attendsClassOf ?Y
 MINUS{{?X1 :attendsClassOf ?X} UNION {?Y :attendsClassOf ?Y2}}}
 .{{?Y a :Student} UNION {?X a :Professor}}}
```

Now, consider an update u' having both INSERT and DELETE clauses:

```
DELETE {?Y a :Professor} INSERT{?X a :Student}
WHERE {?X :attendsClassOf ?Y}
```

The update u' inserts a single class membership fact and thus is always intrinsically consistent. The ASK query in Alg. 4 takes the DELETE clause of u' into account:

```
ASK WHERE {{?X :attendsClassOf ?Y}
.{{?X a :Professor} MINUS {?Z :attendsClassOf ?X }}}
```

■

5 Discussion and Conclusions

In this paper, we have taken a step further from our previous work, in combining SPARQL Update and RDFS entailment by also adding class/concept disjointness as a first step towards dealing with inconsistencies in the context of SPARQL Update. As discussed throughout the paper, previous approaches to handle inconsistencies in DL KB evolution (e.g., [4, 5, 9]) have assumed that the set of ABox assertions to be inserted is intrinsically consistent w.r.t. the TBox, and thus inconsistencies are treated only w.r.t. the old state of the knowledge base. As we have shown, this assumption is not trivially verifiable in the context of SPARQL updates, where DELETE/INSERT atoms are instantiated by a WHERE clause, and clashing triples could be instantiated within the same INSERT operation. We have addressed this problem by providing means to check whether a SPARQL update is intrinsically consistent and defining a safe rewriting that removes intrinsic clashes during inserts on-the-fly.

Next, taken that the problem of intrinsic consistency is solved, we have demonstrated how to extend the approach of [4] to SPARQL updates. We have defined a materialization and consistency preserving rewriting for SPARQL updates that essentially combines the ideas of [4] and our previous work on SPARQL updates under RDFS for materialized triple stores [1], dealing with clashes due to class disjointness axioms in a brave manner. That is, we overwrite inconsistent information in the triple store in favor of information being inserted. Alternatively, we have also defined a dual consistency-preserving update semantics that on the contrary discards insertions that would lead to inconsistencies.

Besides practical evaluation of the proposed algorithms, we plan to further extend our work towards increasing coverage of more expressive logics and OWL profiles, namely additional axioms from OWL 2 RL or OWL 2 QL [10]. Also, it could be useful to investigate further semantics, allowing for compromises between fully discarding the inconsistent old data and refusing the entire update due to clashes, and lift our methods to work with stores that are not fully materialized.

The consideration of negative information is an important issue also in other related works on knowledge base updates: for instance, the seminal work on database view maintenance by Gupta et al. [7] is also used in the context of materialized views using Datalog rules with stratified negation. Likewise, let us mention the work of Winslett [13] on formula-based semantics to updates, where negation is also considered.

Acknowledgements. This work was supported by the Vienna Science and Technology Fund (WWTF), project ICT12-SEE, and EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338.

References

1. Ahmeti, A., Calvanese, D., Polleres, A.: Updating RDFS aboxes and tboxes in SPARQL. In: Proc. of the 13th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 8796, pp. 441–456. Springer (2014)
2. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: RDF 1.1 Turtle – Terse RDF Triple Language. W3C Recommendation, World Wide Web Consortium (Feb 2014), available at `http://www.w3.org/TR/turtle/`

3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)
4. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL-Lite* knowledge bases. In: Proc. of the 9th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 6496, pp. 112–128. Springer (2010)
5. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. of Logic and Computation 19(5), 745–770 (2009)
6. Gearon, P., Passant, A., Polleres, A.: SPARQL 1.1 update. W3C Recommendation, World Wide Web Consortium (Mar 2013), available at `http://www.w3.org/TR/sparql11-update/`
7. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data. pp. 157–166 (1993)
8. Hayes, P., Patel-Schneider, P.: RDF 1.1 semantics. W3C Recommendation, World Wide Web Consortium (Feb 2014), available at `http://www.w3.org/TR/rdf11-mt/`
9. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR). pp. 46–56. AAAI Press (2006)
10. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at `http://www.w3.org/TR/owl2-profiles/`
11. Muñoz, S., Pérez, J., Gutiérrez, C.: Minimal deductive systems for RDF. In: Proc. of the 4th European Semantic Web Conf. (ESWC). Lecture Notes in Computer Science, vol. 4519, pp. 53–67. Springer (2007)
12. Polleres, A., Hogan, A., Delbru, R., Umbrich, J.: RDFS & OWL reasoning for linked data. In: Reasoning Web. Semantic Technologies for Intelligent Data Access – 9th Int. Summer School Tutorial Lectures (RW), Lecture Notes in Computer Science, vol. 8067, pp. 91–149. Springer (2013)
13. Winslett, M.: Updating Logical Databases. Cambridge University Press (2005)

Interval Temporal Description Logics*

A. Artale[1], R. Kontchakov[2], V. Ryzhikov[1], and M. Zakharyaschev[2]

[1]Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
[2]Department of Computer Science and Information Systems
Birkbeck, University of London, U.K.

1 Introduction

In this paper, we construct a combination $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}}$ of the Halpern-Shoham interval temporal logic $\mathcal{HS}$ [15] with the description logic *DL-Lite*$_{horn}^{\mathcal{H}}$ [12, 1], which is a Horn extension of the standard language OWL 2 QL. The temporal operators of $\mathcal{HS}$ are of the form $\langle \mathsf{R} \rangle$ ('diamond') and $[\mathsf{R}]$ ('box'), where R is one of Allen's interval relations *After*, *Begins*, *Ends*, *During*, *Later*, *Overlaps* and their inverses ($\bar{A}$, $\bar{B}$, $\bar{E}$, $\bar{D}$, $\bar{L}$, $\bar{O}$). The propositional variables of $\mathcal{HS}$ are interpreted by sets of closed intervals $[i, j]$ of some flow of time (e.g., $\mathbb{Z}$, $\mathbb{R}$), and a formula $\langle \mathsf{R} \rangle \varphi$ ($[\mathsf{R}]\varphi$) is regarded to be true in $[i, j]$ iff φ is true in some (respectively, all) interval(s) $[i', j']$ such that $[i, j]\mathsf{R}[i', j']$ in Allen's interval algebra.

In $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}}$, we represent temporal data by means of assertions such as *SummerSchool*(RW, t_1, t_2) and *teaches*(US, DL, s_1, s_2), which say that RW is a summer school that takes place in the time interval $[t_1, t_2]$ and US teaches DL in the time interval $[s_1, s_2]$. Note that temporal databases store data in a similar format [17]. Temporal concept and role inclusions are used to impose constraints on the data and introduce new concepts and roles. For example, $\mathit{AdvCourse} \sqcap \langle \bar{D} \rangle \mathit{MorningSession} \sqsubseteq \bot$ says that advanced courses are not given in the morning sessions described by $\langle \bar{B} \rangle \mathit{LectureDay} \sqcap \langle A \rangle \mathit{Lunch} \sqsubseteq \mathit{MorningSession}$; $\mathit{teaches} \sqsubseteq [D]\mathit{teaches}$ claims that the role *teaches* is downward hereditary (or stative) in the sense that if it holds in some interval then it also holds in all of its sub-intervals; $[D](\langle O \rangle \mathit{teaches} \sqcup \langle \bar{D} \rangle \mathit{teaches}) \sqcap \langle B \rangle \mathit{teaches} \sqcap \langle E \rangle \mathit{teaches} \sqsubseteq \mathit{teaches}$, on the contrary, states that *teaches* is coalesced (or upward hereditary). The inclusions $\mathit{teaches} \sqsubseteq [D]\mathit{teaches}$ and $[D](\langle O \rangle \mathit{teaches} \sqcup \langle \bar{D} \rangle \mathit{teaches}) \sqsubseteq \mathit{teaches}$ ensure that *teaches* is both upward and downward hereditary. On the other hand, 'rising stock market' and 'high average speed' are typical examples of concepts that are not downward hereditary; for a discussion of these notions see [6, 21, 18].

Although the complexity of full $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}}$ remains unknown, in this paper we define two fragments, $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}/flat}$ and $\mathcal{HS}$-*Lite*$_{horn}^{\mathcal{H}[G]}$, where satisfiability and instance checking are P-complete for both combined and data complexity.

Our interest in tractable description logics with interval temporal operators is motivated by possible applications in ontology-based data access (OBDA) [12] to *temporal databases*. In this context, we naturally require reasonably expressive yet tractable ontology and query languages with temporal constructs (although

* This extended abstract is an abridged version of [4] presented at AAAI 2015.

some authors advocate the use of standard atemporal OWL 2 QL with temporal queries [16, 7]). Our choice of $\mathcal{HS}$ as the temporal component of $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ is explained by the fact that modern temporal databases adopt the (downward hereditary) interval-based model of time [17, 13] and use coalescing to group time points into intervals [6]. We show that, unfortunately, the logics $\mathcal{HS}$-*Lite*$^{\mathcal{H}/flat}_{horn}$ and $\mathcal{HS}$-*Lite*$^{\mathcal{H}[G]}_{horn}$ cannot guarantee first-order rewritability of even atomic queries, though we conjecture that datalog rewritings are possible.

2 Description Logic $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$

The language of $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ contains *individual names* $a_0, a_1, \dots$, *concept names* $A_0, A_1, \dots$, and *role names* $P_0, P_1, \dots$. *Basic roles* R, *basic concepts* B, *temporal roles* S and *temporal concepts* C are given by the grammar

$$\begin{array}{lcl} R & ::= & P_k \mid P_k^-, \\ S & ::= & R \mid [\mathsf{R}]S \mid \langle \mathsf{R} \rangle S, \end{array} \qquad \begin{array}{lcl} B & ::= & A_k \mid \exists R, \\ C & ::= & B \mid [\mathsf{R}]C \mid \langle \mathsf{R} \rangle C, \end{array}$$

where R is one of Allen's interval relations or the universal relation G. Over the closed intervals $[i, j] = \{n \in \mathbb{Z} \mid i \leq n \leq j\}$, for $i \leq j$, we set:

- $[i, j]A[i', j']$ iff $j = i'$, (After)
- $[i, j]B[i', j']$ iff $i = i'$ and $j \geq j'$, (Begins)
- $[i, j]E[i', j']$ iff $i \leq i'$ and $j = j'$, (Ends)
- $[i, j]D[i', j']$ iff $i \leq i'$ and $j' \leq j$, (During)
- $[i, j]L[i', j']$ iff $j \leq i'$, (Later)
- $[i, j]O[i', j']$ iff $i \leq i' \leq j \leq j'$ (Overlaps)

and define their inverses in the standard way. Note that we allow single-point intervals $[i, i]$ and use non-strict $\leq$ instead of the more common $<$ (in fact, one can show that the use of $<$ would make reasoning non-tractable). An $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ TBox is a finite set of *concept* and *role inclusions* and *disjointness constraints* of the form

$$\begin{array}{ll} C_1 \sqcap \dots \sqcap C_k \sqsubseteq C^+, & S_1 \sqcap \dots \sqcap S_k \sqsubseteq S^+, \\ C_1 \sqcap \dots \sqcap C_k \sqsubseteq \bot, & S_1 \sqcap \dots \sqcap S_k \sqsubseteq \bot, \end{array}$$

where C^+, R^+ denote temporal concepts and roles without diamond operators $\langle \mathsf{R} \rangle$. An $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ ABox is a finite set of atoms of the form $A_k(a, i, j)$ and $P_k(a, b, i, j)$ in which *temporal constants* $i \leq j$ are given in binary. An $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ *knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox.

An $\mathcal{HS}$-*Lite*$^{\mathcal{H}}_{horn}$ *interpretation*, $\mathcal{I}$, consists of a family of standard (atemporal) DL interpretations $\mathcal{I}[i, j] = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}[i,j]})$, for all $i, j \in \mathbb{Z}$ with $i \leq j$, in which $\Delta^{\mathcal{I}} \neq \emptyset$, $a_k^{\mathcal{I}[i,j]} = a_k^{\mathcal{I}}$ for some (fixed) $a_k^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $A_k^{\mathcal{I}[i,j]} \subseteq \Delta^{\mathcal{I}}$ and $P_k^{\mathcal{I}[i,j]} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The role and concept constructs are interpreted in $\mathcal{I}$ as follows:

$$(P_k^-)^{\mathcal{I}[i,j]} = \big\{(x, y) \mid (y, x) \in P_k^{\mathcal{I}[i,j]}\big\}, \qquad (\exists R)^{\mathcal{I}[i,j]} = \big\{x \mid (x, y) \in R^{\mathcal{I}[i,j]}\big\},$$

$$([\mathsf{R}]S)^{\mathcal{I}[i,j]} = \bigcap_{[i,j]\mathsf{R}[i',j']} S^{\mathcal{I}[i',j']}, \qquad ([\mathsf{R}]C)^{\mathcal{I}[i,j]} = \bigcap_{[i,j]\mathsf{R}[i',j']} C^{\mathcal{I}[i',j']}$$

and dually for the 'diamond' operators $\langle \mathsf{R} \rangle$.

The *satisfaction relation* $\models$ is defined by taking:

$$\begin{aligned}
\mathcal{I} \models A(a,i,j) \quad &\text{iff} \quad a^{\mathcal{I}} \in A^{\mathcal{I}[i,j]},\\
\mathcal{I} \models P(a,b,i,j) \quad &\text{iff} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}[i,j]},\\
\mathcal{I} \models \bigsqcap_k C_k \sqsubseteq C \quad &\text{iff} \quad \bigcap_k C_k^{\mathcal{I}[i,j]} \subseteq C^{\mathcal{I}[i,j]}, \quad \text{for } \textit{all} \text{ intervals } [i,j],\\
\mathcal{I} \models \bigsqcap_k S_k \sqsubseteq S \quad &\text{iff} \quad \bigcap_k S_k^{\mathcal{I}[i,j]} \subseteq S^{\mathcal{I}[i,j]}, \quad \text{for } \textit{all} \text{ intervals } [i,j],
\end{aligned}$$

and similarly for disjointness constraints. Note that concept and role inclusions as well as disjointness constraints are interpreted *globally*. For a TBox inclusion or an ABox assertion α, we write $\mathcal{K} \models \alpha$ if $\mathcal{I} \models \alpha$, for all models $\mathcal{I}$ of $\mathcal{K}$.

3 Propositional $\mathcal{HS}_{horn}$ is Tractable

Denote by $\mathcal{HS}_{horn}$ the fragment of $\mathcal{HS}$-$Lite^{\mathcal{H}}_{horn}$ *without* role names and with ABoxes that contain a *single* individual name. TBoxes in this restricted language can be regarded as Horn formulas of the propositional interval temporal logic $\mathcal{HS}$, which is notorious for its nasty computational behaviour; for results on the (un)decidability of various fragments of $\mathcal{HS}$, see, e.g., [14, 10, 9, 8, 19, 11, 20]. The designed logic $\mathcal{HS}_{horn}$ appears to be the first *tractable* fragment of $\mathcal{HS}$:

Theorem 1. *$\mathcal{HS}_{horn}$ is* P*-complete for both combined and data complexity.*

Membership in P follows from the polynomial canonical model and P-hardness for (data) complexity is by reduction of the monotone circuit value problem.

So far, we have managed to lift this result to two proper interval temporal description logics, both of which are fragments of $\mathcal{HS}$-$Lite^{\mathcal{H}}_{horn}$.

4 Tractability of $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$ and $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$

The first fragment, denoted $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$, only allows those $\mathcal{HS}$-$Lite^{\mathcal{H}}_{horn}$ TBoxes that are *flat* in the sense that their concept inclusions do not contain $\exists R$ on the right-hand side. Our second fragment, denoted $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$, allows only the operator $[G]$ in the definition of temporal roles S (with no restrictions imposed on temporal concepts). Thus, unlike $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$, the fragment $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$ contains full *DL-Lite*$^{\mathcal{H}}_{horn}$.

Theorem 2. *(i) The satisfiability problem for $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$ and $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$ KBs is* P*-complete for combined complexity.*

(ii) Instance checking for $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$ and $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$ is P*-complete for data complexity.*

This result contrasts with the lower data complexity (AC^0 and NC^1) of instance checking with point-based temporal *DL-Lite* [5, 3, 2].

In view of Theorem 2 *(ii)*, the temporal ontology languages $\mathcal{HS}$-$Lite^{\mathcal{H}/flat}_{horn}$ and $\mathcal{HS}$-$Lite^{\mathcal{H}[G]}_{horn}$ cannot guarantee first-order rewritability of even atomic queries, though we believe that datalog rewritings are possible. We leave the query rewritability issues, in particular, the design of *DL-Lite*$^{\mathcal{H}}_{core}$-based fragments supporting first-order rewritability as well as temporal extensions of the OWL 2 EL and OWL 2 RL profiles of OWL 2 for future research.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)
2. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: First-order rewritability of temporal ontology-mediated queries. In: Proc. IJCAI 2015. AAAI (2015)
3. Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Temporal OBDA with LTL and DL-Lite. In: Proc. DL 2014. CEUR-WS, vol. 1193, pp. 21–32. (2014)
4. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Tractable interval temporal propositional and description logics. In: Proc. AAAI 2015. (2015)
5. Artale, A., Kontchakov, R., Wolter, F., Zakharyaschev, M.: Temporal description logic for ontology-based data access. In: Proc. IJCAI 2013. pp. 711–717. IJCAI/AAAI (2013)
6. Böhlen, M.H., Snodgrass, R.T., Soo, M.D.: Coalescing in temporal databases. In: Proc. VLDB'96. pp. 180–191. Morgan Kaufmann (1996)
7. Borgwardt, S., Lippmann, M., Thost, V.: Temporal query answering in the description logic DL-Lite. In: Proc. FroCoS 2013. LNCS, vol. 8152, pp. 165–180. Springer (2013)
8. Bresolin, D., Della Monica, D., Goranko, V., Montanari, A., Sciavicco, G.: The dark side of interval temporal logic: marking the undecidability border. Annals of Mathematics and Artificial Intelligence 71(1–3), 41–83 (2014)
9. Bresolin, D., Della Monica, D., Montanari, A., Sala, P., Sciavicco, G.: Interval temporal logics over finite linear orders: the complete picture. In: Proc. ECAI 2012. pp. 199–204. IOS Press (2012)
10. Bresolin, D., Della Monica, D., Montanari, A., Sala, P., Sciavicco, G.: Interval temporal logics over strongly discrete linear orders: the complete picture. In: Proc. GandALF 2012. EPTCS, vol. 96, pp. 155–168 (2012)
11. Bresolin, D., Della Monica, D., Montanari, A., Sciavicco, G.: The light side of interval temporal logic: the Bernays-Schönfinkel fragment of CDT. Annals of Mathematics and Artificial Intelligence 71(1–3), 11–39 (2014)
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
13. Date, C.J., Darwen, H., Lorentzos, N: Temporal data and the relational model. Elsevier. (2002)
14. D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Many-dimensional modal logics: theory and applications. Studies in Logic. Elsevier, 2003.
15. Halpern, J.Y., Shoham, Y.: A propositional modal logic of time intervals. Journal of the ACM 38(4), 935–962 (1991)
16. Klarman, S.: Practical querying of temporal data via OWL 2 QL and SQL:2011. In: Short Papers Proc. LPAR 2013. EPiC Series 26, pp. 52–61. Easychair (2014)
17. Kulkarni, K.G., Michels, J.E.: Temporal features in SQL:2011. SIGMOD Record 41(3), 34–43 (2012)
18. Leo, J., Parsia, B., Sattler, U.: Temporalising $\mathcal{EL}$ concepts with time intervals. In: Proc. DL. CEUR-WS, vol. 1193, pp. 620–632. (2014)
19. Marcinkowski, J., Michaliszyn, J.: The undecidability of the logic of subintervals. Fundamenta Informaticae 131(2), 217–240 (2014)

20. Montanari, A., Puppis, G., Sala, P.: Decidability of the interval temporal logic $A\bar{A}B\bar{B}$ over the rationals. In: Proc. MFCS 2014. LNCS, vol. 8634, pp. 451–463. Springer (2014)
21. Terenziani, P., Snodgrass, R.T.: Reconciling point-based and interval-based semantics in temporal relational databases: A treatment of the Telic/Atelic distinction. IEEE Transactions on Knowledge and Data Engineering 16(5), 540–551 (2004)

Dismatching and Local Disunification in $\mathcal{EL}$ (Extended Abstract)

Franz Baader, Stefan Borgwardt, and Barbara Morawska*

Theoretical Computer Science, TU Dresden, Germany
{baader,stefborg,morawska}@tcs.inf.tu-dresden.de

Motivation

Unification in Description Logics was introduced in [6] as a novel inference service that can be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe head injury* using the $\mathcal{EL}$-concepts

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Head\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_finding} \sqcap \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}). \tag{2}$$

Formally, these expressions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names $\mathsf{Head\_injury}$ and $\mathsf{Severe\_finding}$ as variables, and substituting them by $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\exists \mathsf{severity}.\mathsf{Severe}$, respectively. In this case, we say that the concepts are unifiable, and call the substitution that makes them equivalent a *unifier*. In [5], we were able to show that unification in $\mathcal{EL}$ is NP-complete. The main idea underlying the proof of this result is to show that any solvable $\mathcal{EL}$-unification problem has a local unifier, i.e., a unifier built from a polynomial number of *atoms* (concept names or existential restrictions), which are determined by the unification problem. This yields a brute-force NP-algorithm for unification, which guesses a local substitution and then checks (in polynomial time) whether it is a unifier.

Intuitively, a unifier of two $\mathcal{EL}$ concepts proposes definitions for the concept names that are used as variables: in our example, we know that, if we define $\mathsf{Head\_injury}$ as $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\mathsf{Severe\_finding}$ as $\exists \mathsf{severity}.\mathsf{Severe}$, then the two concepts (1) and (2) are equivalent w.r.t. these definitions. Of course, this example was constructed such that the unifier (which is local) provides sensible definitions for the concept names used as variables. In general, the existence of a unifier only says that there is a structural similarity between the two concepts. The developer that uses unification needs to inspect the unifier(s) to see whether the definitions it suggests really make sense. For example, the substitution that replaces $\mathsf{Head\_injury}$ by $\mathsf{Patient} \sqcap \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\mathsf{Severe\_finding}$ by $\mathsf{Patient} \sqcap \exists \mathsf{severity}.\mathsf{Severe}$ is also a local unifier, which however does not make sense. Unfortunately, even small unification problems like the one

* Supported by DFG under grant BA 1122/14-1

in our example can have too many local unifiers for manual inspection. We propose disunification to avoid local unifiers that do not make sense. In addition to positive constraints (requiring equivalence or subsumption between concepts), a disunification problem may also contain negative constraints (preventing equivalence or subsumption between concepts). In our example, the nonsensical unifier can be avoided by adding the dissubsumption constraint

$$\mathsf{Head\_injury} \not\sqsubseteq^? \mathsf{Patient} \tag{3}$$

to the equivalence constraint (1) $\equiv^?$ (2).

Disunification in DLs is closely related to unification and admissibility in modal logics [7, 10–15], as well as (dis)unification modulo equational theories [5, 6, 8, 9]. In the following, we shortly describe the ideas behind our work. More details can be found in [2, 3].

Preliminaries

We designate certain concept names as *variables*, while all others are *constants*. An $\mathcal{EL}$-concept is *ground* if it contains no variables. We consider *(basic) disunification problems*, which are conjunctions of *subsumptions* $C \sqsubseteq^? D$ and *dissubsumptions* $C \not\sqsubseteq^? D$ between concepts C, D.[1] A *substitution* maps each variable to a ground concept, and can be extended to concepts as usual. A substitution σ *solves* a disunification problem Γ if the (dis)subsumptions of Γ become valid when applying σ on both sides. We restrict σ to a finite signature of concept and role names and do not allow variables to occur in a solution—it would not make sense for the new definitions to extend the vocabulary of the ontologies under consideration, nor to define variables in terms of themselves.

In the following, we consider a *flat* disunification problem Γ, i.e. one that contains only (dis)subsumptions where both sides are conjunctions of *flat atoms* of the form A or $\exists r.A$, for a role name r and a concept name A. We denote by At the set of all such atoms that occur in Γ, by Var the set of variables occurring in Γ, and by $\mathsf{At_{nv}} := \mathsf{At} \setminus \mathsf{Var}$ the set of *non-variable atoms* of Γ. Let $S \colon \mathsf{Var} \to 2^{\mathsf{At_{nv}}}$ be an *assignment*, i.e. a function that assigns to each variable $X \in \mathsf{Var}$ a set $S_X \subseteq \mathsf{At_{nv}}$. The relation $>_S$ on Var is defined as the transitive closure of $\{(X, Y) \in \mathsf{Var}^2 \mid Y \text{ occurs in an atom of } S_X\}$. If $>_S$ is irreflexive, then S is called *acyclic*. In this case, we can define the substitution σ_S inductively along $>_S$ as follows: if X is minimal, then $\sigma_S(X) := \bigsqcap_{D \in S_X} D$; otherwise, assume that $\sigma_S(Y)$ is defined for all $Y \in \mathsf{Var}$ with $X > Y$, and define $\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D)$. All substitutions of this form are called *local*.

Results

Unification in $\mathcal{EL}$ is *local*: each problem Γ can be transformed into an equivalent flat problem that has a local solution iff Γ is solvable, and hence (general) solvability of unification problems in $\mathcal{EL}$ is in NP [5]. However, disunification in $\mathcal{EL}$

[1] A *unification problem* contains only subsumptions.

is *not local* in this sense: consider

$$\Gamma := \{X \sqsubseteq^? B,\ A \sqcap B \sqcap C \sqsubseteq^? X,\ \exists r.X \sqsubseteq^? Y,\ \top \not\sqsubseteq^? Y,\ Y \not\sqsubseteq^? \exists r.B\}$$

with variables X, Y and constants A, B, C. If we set $\sigma(X) := A \sqcap B \sqcap C$ and $\sigma(Y) := \exists r.(A \sqcap C)$, then σ is a solution of Γ that is not local. This is because $\exists r.(A \sqcap C)$ is not a substitution of any non-variable atom in Γ. Assume now that Γ has a local solution γ. Since γ must solve the first dissubsumption, $\gamma(Y)$ cannot be $\top$, and due to the third subsumption, none of the constants A, B, C can be a conjunct of $\gamma(Y)$. The remaining atoms $\exists r.\gamma(X)$ and $\exists r.B$ are ruled out by the last dissubsumption since both $\gamma(X)$ and B are subsumed by B. This shows that Γ cannot have a local solution, although it is solvable.

The decidability and complexity of general solvability of disunification problems is still open. However, we can show that each *dismatching* problem Γ, which is a disunification problem where one side of each dissubsumption must be ground, can be polynomially reduced to a flat problem that has a local solution iff Γ is solvable. This shows that deciding solvability of dismatching problems in $\mathcal{EL}$ is in NP. The main idea is to introduce auxiliary variables and flat atoms that allow us to solve the dissubsumptions using a local substitution. For example, we replace the dissubsumption $\top \not\sqsubseteq^? Y$ from above with $Y \sqsubseteq^? \exists r.Z$. The rule we applied here is the following:

Solving Left-Ground Dissubsumptions:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ if X is a variable and $C_1, \ldots, C_n$ are ground atoms.
Action: Choose one of the following options:
- Choose a constant $A \in \Sigma$, replace $\mathfrak{s}$ by $X \sqsubseteq^? A$. If $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq A$, then *fail*.
- Choose a role $r \in \Sigma$, introduce a new variable Z, replace $\mathfrak{s}$ by $X \sqsubseteq^? \exists r.Z$, $C_1 \not\sqsubseteq^? \exists r.Z$, $\ldots$, $C_n \not\sqsubseteq^? \exists r.Z$.

According to the rule, we can choose a constant or create a new existential restriction with a fresh variable, and use it in the new subsumption and dissubsumptions. In our example the left hand side of the dissubsumption $\top \not\sqsubseteq^? Y$ is empty, hence only a subsumption is produced.

However, the brute-force NP-algorithm for checking local solvability of the resulting flat disunification problem is hardly practical. For this reason, we have extended the rule-based algorithm from [5] and the SAT reduction from [4] by additional rules and propositional clauses, respectively, to deal with dissubsumptions. The reason we extend both algorithms is that, in the case of unification, they have proved to complement each other well in first evaluations [1]: the goal-oriented algorithm needs less memory and finds minimal solutions faster, while the SAT reduction generates larger data structures, but outperforms the goal-oriented algorithm on unsolvable problems. The SAT reduction has been implemented in our prototype system UEL.[2] First experiments show that dismatching is indeed helpful for reducing the number and the size of unifiers. The runtime performance of the solver for dismatching problems is comparable to the one for pure unification problems.

[2] version 1.3.0, available at http://uel.sourceforge.net/

References

1. Baader, F., Borgwardt, S., Mendez, J.A., Morawska, B.: UEL: Unification solver for $\mathcal{EL}$. In: Kazakov, Y., Lembo, D., Wolter, F. (eds.) Proc. of the 25th Int. Workshop on Description Logics (DL'12). CEUR Workshop Proceedings, vol. 846, pp. 26–36 (2012)
2. Baader, F., Borgwardt, S., Morawska, B.: Dismatching and local disunfication in $\mathcal{EL}$. LTCS-Report 15-03, Chair for Automata Theory, TU Dresden, Germany (2015), see http://lat.inf.tu-dresden.de/research/reports.html.
3. Baader, F., Borgwardt, S., Morawska, B.: Dismatching and local disunification in $\mathcal{EL}$. In: Fernández, M. (ed.) Proc. of the 26th Int. Conf. on Rewriting Techniques and Applications (RTA'15). LIPIcs, vol. 36. Dagstuhl Publishing (2015), to appear.
4. Baader, F., Morawska, B.: SAT encoding of unification in $\mathcal{EL}$. In: Fermüller, C.G., Voronkov, A. (eds.) Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10). Lecture Notes in Computer Science, vol. 6397, pp. 97–111. Springer-Verlag (2010)
5. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
6. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
7. Babenyshev, S., Rybakov, V.V., Schmidt, R., Tishkovsky, D.: A tableau method for checking rule admissibility in $S4$. In: Proc. of the 6th Workshop on Methods for Modalities (M4M-6). Copenhagen (2009)
8. Buntine, W.L., Bürckert, H.J.: On solving equations and disequations. J. of the ACM 41(4), 591–629 (1994)
9. Comon, H.: Disunification: A survey. In: Lassez, J.L., Plotkin, G. (eds.) Computational Logic: Essays in Honor of Alan Robinson, pp. 322–359. MIT Press, Cambridge, MA (1991)
10. Ghilardi, S.: Unification through projectivity. Journal of Logic and Computation 7(6), 733–752 (1997)
11. Ghilardi, S.: Unification in intuitionistic logic. Journal of Logic and Computation 64(2), 859–880 (1999)
12. Iemhoff, R., Metcalfe, G.: Proof theory for admissible rules. Annals of Pure and Applied Logic 159(1-2), 171–186 (2009)
13. Rybakov, V.V.: Admissibility of logical inference rules, Studies in Logic and the Foundations of Mathematics, vol. 136. North-Holland Publishing Co., Amsterdam (1997)
14. Rybakov, V.V.: Multi-modal and temporal logics with universal formula - reduction of admissibility to validity and unification. Journal of Logic and Computation 18(4), 509–519 (2008)
15. Wolter, F., Zakharyaschev, M.: Undecidability of the unification and admissibility problems for modal and description logics. ACM Transactions on Computational Logic 9(4), 25:1–25:20 (2008)

Extending Consequence-Based Reasoning to $\mathcal{SHIQ}$

Andrew Bate, Boris Motik, Bernardo Cuenca Grau,
František Simančík, and Ian Horrocks

University of Oxford
Oxford, United Kingdom
first.last@cs.ox.ac.uk

1 Introduction

Description logics (DLs) [3] are a family of knowledge representation formalisms with numerous practical applications. $\mathcal{SHIQ}$ is a particularly important DL as it provides the formal underpinning for the Web Ontology Language (OWL). DLs model a domain of interest using *concepts* (i.e., unary predicate symbols) and *roles* (i.e., binary predicate symbols). DL applications often rely on *subsumption*—the problem of checking logical entailment between concepts—and so the development of practical subsumption procedures for DLs such as $\mathcal{SHIQ}$ has received a lot of attention.

Most DLs are fragments of the *guarded fragment* [1] of first-order logic; however, $\mathcal{SHIQ}$ provides a restricted form of counting that does not fall within the guarded fragment. Moreover, most DLs, including $\mathcal{SHIQ}$, can be captured using the *two-variable fragment of first-order logic with counting* ($\mathcal{C}^2$) [11], but this provides us with neither a practical nor a worst-case optimal reasoning procedure ($\mathcal{C}^2$ and $\mathcal{SHIQ}$ are NEXPTIME- and EXPTIME-complete, respectively). Algorithms for more general logics thus do not satisfy the requirements of DL applications, and so numerous alternatives specific to DLs have been explored. Many DLs can be decided in the framework of resolution [18, 13], including $\mathcal{SHIQ}$ [14]. These procedures are usually worst-case optimal and can be practical, but, as we discuss in Section 3, in even very simple cases they can draw unnecessary inferences. Practically successful $\mathcal{SHIQ}$ reasoners, such as FaCT++ [26], HermiT [9], Pellet [25], and Racer [12], use variants of highly-optimised (hyper)tableau algorithms [6]—model-building algorithms that ensure termination by a variant of *blocking* [7]. Although worst-case optimal tableau algorithms are known [10], practical implementations are typically not worst-case optimal. While generally very effective, tableau algorithms still cannot process certain ontologies; for example, the GALEN ontology[1] has proved particularly challenging, mainly because tableau calculi tend to construct very large models.

A breakthrough in practical ontology reasoning came in the form of *consequence-based calculi*. Although not originally presented in the consequence-based framework, the algorithm for the DL $\mathcal{EL}$ [2] can be seen as the first such calculus. This algorithm was later reformulated and extended to Horn-$\mathcal{SHIQ}$ [15] and Horn-$\mathcal{SROIQ}$ [19]—DLs that support functional roles, but not disjunctive reasoning. Recently, consequence-based calculi were also developed for the DLs $\mathcal{ALCH}$ [24] and $\mathcal{ALCI}$ [23], which support disjunctive reasoning, but not counting. Consequence-based calculi can be seen as

[1] http://www.opengalen.org

a combination of resolution and hypertableau (see Section 3 for details). As in resolution, they describe ontology models by systematically deriving certain ontology consequences; and as in hypertableau, the ontology axioms can be used to guide the derivation process, and to avoid drawing unnecessary inferences. Moreover, consequence-based calculi are not just refutationally complete, but can classify an ontology in a single pass, which greatly reduces the overall work. These advantages allowed the CB system to be the first to classify all of GALEN [15].

Existing consequence-based algorithms can handle either disjunctions or counting, but not both. As we discuss in detail in Section 4, it is challenging to extend these algorithms to DLs such as $\mathcal{SHIQ}$ that combine both kinds of construct: counting quantifiers require equality reasoning, which together with disjunctions can impose complex constraints on ontology models. Unlike in existing consequence-based calculi, these constraints cannot be captured using DLs themselves; instead, a more expressive first-order fragment is needed, which makes the reasoning process much more involved.

In Section 5 we present a consequence-based calculus for $\mathcal{SHIQ}$. Borrowing ideas from resolution theorem proving, we encode the required consequences using a special kind of first-order clause; and to handle equality effectively, we base our calculus on *ordered paramodulation* [17]—a state of the art calculus for equational theorem proving used in modern systems such as E [22] and Vampire [20]. To make the calculus efficient on $\mathcal{EL}$, we have carefully constrained the rules so that, on $\mathcal{EL}$ ontologies, it mimics existing $\mathcal{EL}$ calculi. Thus, although a practical evaluation of our calculus is still pending, we believe that it is likely to perform well in practice on 'mostly-$\mathcal{EL}$' ontologies due to is close relationship with existing and well-proven calculi.

2 Preliminaries

First-Order Logic. To simplify matters technically, it is common practice in equational theorem proving to encode atoms as terms. An atomic formula $P(\vec{s})$ can be encoded as $P(\vec{s}) \approx \mathsf{t}$, where t is a new special constant, and P is considered as a function symbol rather than as a predicate symbol. Note however that, in order to avoid meaningless expressions in which predicate symbols occur at proper subterms, a multi-sorted type discipline on terms in the encoding is adopted. Thus, the set of symbols in the signature is partitioned into a set $\mathcal{P}$ of *predicate symbols* (which includes t), and a set $\mathcal{F}$ of *function symbols*.

A *term* is constructed as usual using variables and the signature symbols. Terms containing predicate symbols as their outermost symbol are called $\mathcal{P}$*-terms*, while all other terms are $\mathcal{F}$*-terms*. For example, for P a predicate and f a function symbol, both $f(P(x))$ and $P(P(x))$ are malformed; $P(f(x))$ is a well-formed $\mathcal{P}$-term; and $f(x)$ is a well-formed $\mathcal{F}$-term. An *(in)equality* is an expression of the form $s \approx t$ ($s \not\approx t$) where s and t are both either $\mathcal{F}$- or $\mathcal{P}$-terms. We assume that $\approx$ and $\not\approx$ are implicitly symmetric—that is, $s \bowtie t$ and $t \bowtie s$ are one and the same expression, for $\bowtie \in \{\approx,\ \not\approx\}$. A *literal* is an equality or an inequality. An *atom* is an equality of the form $P(\vec{s}) \approx \mathsf{t}$, and we write it simply as $P(\vec{s})$ whenever it is clear from the context whether $P(\vec{s})$ is intended to be a $\mathcal{P}$-term or an atom. A *clause* is an expression of the form $\Gamma \rightarrow \Delta$ where Γ is a conjunction of atoms called the *body*, and Δ is a disjunction of literals called the

Table 1. Translating Normalised $\mathcal{SHIQ}$ Ontologies into DL-Clauses

$B_1 \sqsubseteq \geqslant n\, S.B_2$	$\rightsquigarrow$	$B_1(x) \to S(x, f_i(x))$	for $1 \leq i \leq n$
		$B_1(x) \to B_2(f_i(x))$	for $1 \leq i \leq n$
		$B_1(x) \to f_i(x) \not\approx f_j(x)$	for $1 \leq i < j \leq n$
$B_1 \sqsubseteq \leqslant n\, S.B_2$	$\rightsquigarrow$	$S(y,x) \wedge B_2(x) \to S_{B_2}(y,x)$	for fresh S_{B_2}
		$B_1(x) \wedge \bigwedge_{1 \leq i \leq n+1} S_{B_2}(x, z_i) \to \bigvee_{1 \leq i < j \leq n+1} z_i \approx z_j$	
$B_1 \sqsubseteq \forall S.B_2$	$\rightsquigarrow$	$B_1(x) \wedge S(x, z_1) \to B_2(z_1)$	
$\bigsqcap_{1 \leq i \leq n} B_i \sqsubseteq \bigsqcup_{1 \leq j \leq m} B_j$	$\rightsquigarrow$	$\bigwedge_{1 \leq i \leq n} B_i(x) \to \bigvee_{1 \leq i \leq m} B_j(x)$	
$S_1 \sqsubseteq S_2$	$\rightsquigarrow$	$S_1(x, z_1) \to S_2(x, z_1)$	
$S_1 \sqsubseteq S_2^-$	$\rightsquigarrow$	$S_1(x, z_1) \to S_2(z_1, x)$	

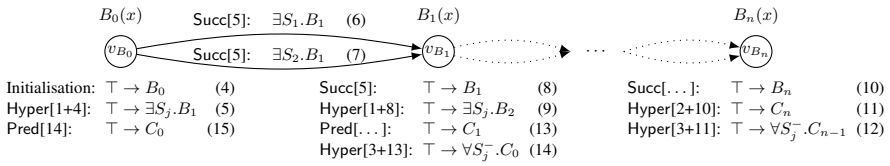
head. We often treat conjunctions and disjunctions as sets; and we write the empty conjunction (disjunction) as $\top$ ($\bot$). We use the standard notion of subterm positions; then, $s|_p$ is the subterm of s at position p; moreover, $s[t]_p$ is the term obtained from s by replacing the subterm at position p with t; finally, position p is *proper* in t if $t|_p \neq t$.

Orders. A *term order* $\succ$ is a *strict* order on the set of all terms. The *multiset extension* $\succ_{mul}$ of $\succ$ compares multisets M and N on a universe U such that $M \succ_{mul} N$ if and only if $M \neq N$ and, for each $n \in N \setminus M$, some $m \in M \setminus N$ exists such that $m \succ n$, where $\setminus$ is the multiset difference. We extend $\succ$ to literals by identifying each $s \not\approx t$ with the multiset $\{s, s, t, t\}$ and each $s \approx t$ with the multiset $\{s, t\}$, and by comparing the result using $\succ_{mul}$. Given an order $\succ$, element $b \in U$, and subset $S \subset U$, the notation $S \succ b$ abbreviates $\exists a \in S : a \succ b$.

Description Logic $\mathcal{SHIQ}$. In this paper, a $\mathcal{SHIQ}$ ontology is represented as a set of *DL-clauses*, which we define next. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be countable sets of unary and binary predicate symbols, and let $\mathcal{F}$ be a countable set of unary function symbols. DL-clauses are constructed using the *central variable* x and variables z_i. A *DL-$\mathcal{F}$-term* has the form x, z_i, or $f(x)$ with $f \in \mathcal{F}$; a *DL-$\mathcal{P}$-term* has the form $B(z_i)$, $B(x)$, $B(f(x))$, $S(x, z_i)$, $S(z_i, x)$, $S(x, f(x))$, $S(f(x), x)$ with $B \in \mathcal{P}_1$ and $S \in \mathcal{P}_2$; and a *DL-term* is a DL-$\mathcal{F}$- or a DL-$\mathcal{P}$-term. A *DL-literal* has the form $A \approx \mathtt{t}$ with A a DL-$\mathcal{P}$-term (called a *DL-atom*), or $f(x) \bowtie g(x)$, $f(x) \bowtie z_i$, or $z_i \bowtie z_j$ with $\bowtie \in \{\approx, \not\approx\}$. A *DL-clause* contains only DL-atoms of the form $B(x)$, $S(x, z_i)$, and $S(z_i, x)$ in the body and only DL-literals in the head, and where each variable z_i occurring in the head also occurs in the body. An *ontology* $\mathcal{O}$ is a finite set of DL-clauses. A *query clause* is a DL-clause in which all atoms are of the form $B(x)$. Given an ontology $\mathcal{O}$ and a query clause $\Gamma \to \Delta$, the *query clause entailment* problem is to decide whether $\mathcal{O} \models \forall x.(\Gamma \to \Delta)$ holds; we often leave out $\forall x$ and write the latter as $\mathcal{O} \models \Gamma \to \Delta$.

$\mathcal{SHIQ}$ ontologies are commonly written using a DL-style syntax, but we can always transform such ontologies into DL-clauses without affecting the entailment of query clauses. Transitivity is encoded away as described in [21, 8], and the resulting axioms are *normalised* to the forms shown on the left-hand side of Table 1 as described in [15, 23]. The normalised axioms are translated to DL-clauses as shown in Table 1.

$$\mathcal{O}_1 = \{ \quad B_i \sqsubseteq \exists S_j.B_{i+1} \quad \text{for } 0 \leq i < n \text{ and } 1 \leq j \leq 2 \quad (1)$$
$$B_n \sqsubseteq C_n \quad (2)$$
$$C_{i+1} \sqsubseteq \forall S_j^-.C_i \quad \text{for } 0 \leq i < n \text{ and } 1 \leq j \leq 2 \quad (3) \ \}$$



Initialisation: $\top \to B_0$ (4)
Hyper[1+4]: $\top \to \exists S_j.B_1$ (5)
Pred[14]: $\top \to C_0$ (15)

Succ[5]: $\top \to B_1$ (8)
Hyper[1+8]: $\top \to \exists S_j.B_2$ (9)
Pred[...]: $\top \to C_1$ (13)
Hyper[3+13]: $\top \to \forall S_j^-.C_0$ (14)

Succ[...]: $\top \to B_n$ (10)
Hyper[2+10]: $\top \to C_n$ (11)
Hyper[3+11]: $\top \to \forall S_j^-.C_{n-1}$ (12)

Fig. 1. Example Motivating Consequence-Based Calculi

3 Why Consequence-Based Calculi?

Consider the ontology $\mathcal{O}_1$ (written using DL notation) shown in Figure 1. Axiom (3) can be reformulated as $\exists S_j.C_{i+1} \sqsubseteq C_i$, and so $\mathcal{O}_1$ is in $\mathcal{EL}$. One can readily verify that $\mathcal{O} \models B_i \sqsubseteq C_i$ holds for each $1 \leq i \leq n$.

To prove, say, $\mathcal{O} \models B_0 \sqsubseteq C_0$, a (hyper)tableau calculus constructs in a forward-chaining manner a tree-shaped model of depth n and of fanout two, where nodes at depth i are labelled by B_i and C_i. Forward chaining ensures that reasoning is goal-oriented and thus amenable to practical implementation. However, all nodes labelled with B_i are of the same type and behave in the same way, which reveals a weakness of (hyper)tableau calculi: the constructed model can be large (exponential in our example) and highly redundant. Techniques such as *caching* [10] or *anywhere blocking* [16] can be used to constrain model construction, but their effectiveness often depends on the order of rule applications. Thus, model size has proved to be a key limiting factor for (hyper)tableau-based reasoners in practice [16].

In contrast, resolution describes models using universally quantified clauses that 'summarise' the model. This prevents redundancy and ensures worst-case optimality of many resolution decision procedures. Nevertheless, resolution can still derive unnecessary clauses. In our example, axioms (1) and (3) are translated into clauses (16) and (17), respectively, which can be used to derive all $2n^2$ clauses of the form (18).

$$B_i(x) \to S_j(x, f_{i,j}(x)) \quad \text{for } i \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \quad (16)$$
$$S_j(z_1, x) \wedge C_{k+1}(x) \to C_k(z_1) \quad \text{for } k \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \quad (17)$$
$$B_i(x) \wedge C_{k+1}(f_{i,j}(x)) \to C_k(x) \quad \text{for } i, k \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \quad (18)$$

Of these $2n^2$ clauses, only those where $i = k$ are relevant to proving $\mathcal{O} \models B_0 \sqsubseteq C_0$. Moreover, if we extend $\mathcal{O}$ with additional clauses that contain B_i and C_i, each of the $2n^2$ clauses from (18) can participate in further inferences, which can give rise to many more irrelevant conclusions. This problem is exacerbated in satisfiable cases since all resolution consequences must then be computed in full.

Consequence-based calculi combine the goal-directed reasoning of (hyper)tableau calculi with the 'summarisation' of resolution. In [23], we presented a very general framework for $\mathcal{ALCI}$ ontologies that captures the key elements of consequence-based calculi such as [2, 15, 19, 24]. We use this framework as basis for our extension to

$\mathcal{SHIQ}$ so, before presenting our extension, we explain the main concepts on $\mathcal{O}_1$. Due to space restrictions we cannot reproduce in full the inference rules from [23]; however, these are similar in spirit to our inference rules for $\mathcal{SHIQ}$ presented in Table 2.

Our calculus constructs a *context structure* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$—a graph whose vertices $\mathcal{V}$ are called *contexts* and whose directed edges are labelled with concepts of the form $\exists S.B$. Let I be a model of $\mathcal{O}$. Instead of representing each element of I individually as in (hyper)tableau calculi, we 'summarise' all elements of a certain kind using a single context v. Each context $v \in \mathcal{V}$ is associated with a (possibly empty) set core_v of *core* concepts that hold in all domain elements that v represents; thus, core_v determines the kind of context v. We use a set $\mathcal{S}_v$ of clauses to capture additional constraints that the elements represented by v must satisfy; in $\mathcal{ALCI}$, we can do so using clauses over DL concepts of the form $\bigsqcap B_i \sqsubseteq \bigsqcup B_j \sqcup \bigsqcup \exists S_k.B_k \sqcup \bigsqcup \forall S_\ell.B_\ell$. Thus, unlike in resolution where all consequences belong to a single set, we assign a consequence a particular set in order to reduce the number of inferences. Clauses in $\mathcal{S}_v$ are 'relative' to core_v: for each $\Gamma \sqsubseteq \Delta \in \mathcal{S}_v$, we have $\mathcal{O} \models \mathsf{core}_v \sqcap \Gamma \sqsubseteq \Delta$—that is, we choose not to include core_v in clause bodies since core_v always holds. Finally, $\succ$ provides each context $v \in \mathcal{V}$ with a concept order $\succ_v$ that restricts resolution inferences in the presence of disjunctions.

Consequence-based calculi are not just refutation-complete: they actually derive the required consequences. Figure 1 shows how this is achieved for $\mathcal{O}_1 \models B_0 \sqsubseteq C_0$; the core and the clauses are shown, respectively, above and below a context. To prove $B_0 \sqsubseteq C_0$, we introduce context v_{B_0} with $\mathsf{core}_{v_{B_0}} = \{B_0\}$ and clause (4) stating that B_0 holds in this context. Next, using the Hyper rule, we derive (5) from (1) and (4); this rule performs hyperresolution, but restricted to one context at a time.

Next, the Succ rule satisfies the existential quantifiers in (5). To this end, the rule uses a parameter called an *expansion strategy*. A strategy is given two sets of constraints that a successor of v_{B_0} must satisfy due to universal restrictions: K_1 contains constraints that must hold, and K_2 contains constraints that might hold. Given such K_1 and K_2, the strategy then decides whether to reuse an existing context or create a fresh one, and in the latter case it also determines how to initialise the new context's core. In our example, there are no universal restrictions and all information in v_{B_0} is deterministic, so $K_1 = K_2 = \{B_1\}$. For $\mathcal{EL}$, a reasonable strategy is to associate with each concept B_i a context v_{B_i} with $\mathsf{core}_{v_{B_i}} = \{B_i\}$, and to always to satisfy existential quantifiers of the form $\exists S.B_i$ using v_{B_i}; thus, in our example we introduce v_{B_1} and initialise it with (8). Note that (5) represents two existential quantifiers, both of which we satisfy (in separate applications of the Succ rule) using v_{B_1}. Different strategies may be used with more expressive DLs; please refer to [23, Section 3.4] for an in-depth discussion.

We construct contexts $v_{B_2}, \ldots, v_{B_n}$ in a similar way, finally deriving (11) by hyperresolving (2) and (10), and then (12) by hyperresolving (3) and (11). Clause (12) imposes a constraint on the predecessor context, which we propagate backwards using the Pred rule, obtaining (13) and (15). Since, however, clauses in $\mathcal{S}_{v_{B_0}}$ are 'relative' to $\mathsf{core}_{v_{B_0}}$, clause (15) actually represents our query clause $B_0 \sqsubseteq C_0$.

Thus, like resolution, consequence-based calculi 'summarise' models to prevent redundant computation, and, like (hyper)tableau calculi, they differentiate elements in a model of $\mathcal{O}$ to prevent the derivation of consequences such as (18).

$$\mathcal{O}_2 = \{ \quad B_0(x) \to S(f_1(x), x) \ (19) \quad B_1(x) \to S(x, f_i(x)) \ \text{for } 2 \le i \le 3 \ (22)$$
$$B_0(x) \to B_1(f_1(x)) \ (20) \quad B_1(x) \to B_i(f_i(x)) \ \text{for } 2 \le i \le 3 \ (23)$$
$$B_2(x) \wedge B_3(x) \to \bot \ (21) \quad B_i(x) \to B_4(x) \ \text{for } 2 \le i \le 3 \ (24)$$
$$B_1(x) \wedge \textstyle\bigwedge_{1 \le j \le 3} S(x, z_i) \to \bigvee_{1 \le j < k \le 3} z_j \approx z_k \ (25) \ \}$$

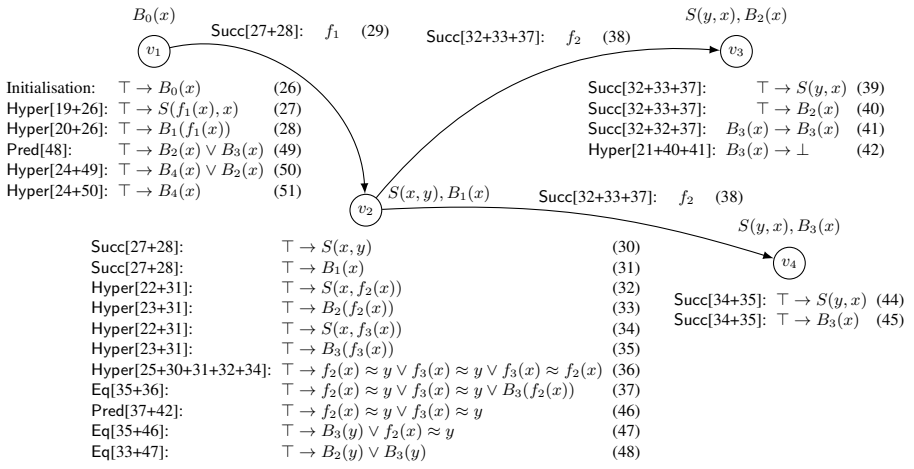


Fig. 2. Challenges in Extending the Consequence-Based Framework to $\mathcal{SHIQ}$

4 Extending the Framework to $\mathcal{SHIQ}$

We now present an example before formalising the calculus. Due to an interaction between counting quantifiers and inverse roles, a $\mathcal{SHIQ}$ ontology can impose more complex constraints on model elements than $\mathcal{ALCI}$. Let $\mathcal{O}_2$ be the $\mathcal{SHIQ}$ ontology shown in Figure 2; we argue that $\mathcal{O}_2 \models B_0(x) \to B_4(x)$ holds. To see why, consider an equality Herbrand interpretation I constructed from $B_0(a)$. Then, (19) and (20) derive $S(f_1(a), a)$ and $B_1(f_1(a))$; moreover, (22) and (23) derive $S(f_1(a), f_2(f_1(a)))$ and $B_2(f_2(f_1(a)))$, and $S(f_1(a), f_3(f_1(a)))$ and $B_3(f_3(f_1(a)))$. Due to (24) we derive $B_4(f_2(f_1(a)))$ and $B_4(f_3(f_1(a)))$. Finally, from (25) we derive (52).

$$f_2(f_1(a)) \approx a \vee f_3(f_1(a)) \approx a \vee f_3(f_1(a)) \approx f_2(f_1(a)) \tag{52}$$

We must satisfy at least one disjunct in (52). Disjunct $f_3(f_1(a)) \approx f_2(f_1(a))$ cannot be satisfied due to (21); but then, regardless of whether we satisfy $f_3(f_1(a)) \approx a$ or $f_2(f_1(a)) \approx a$, we derive $B_4(a)$; hence, the inference holds.

To prove this in our consequence-based framework, we must capture constraint (52) and its consequences. However, this cannot be done using standard description logic notation because DL concepts cannot identify specific successors and predecessors of $f_1(a)$—that is, they cannot say 'either the first or the second successor is equal to the predecessor'. Thus, our main challenges are to devise a method for representing all the relevant constraints that can be induced by $\mathcal{SHIQ}$ ontologies, and to ensure that such constraints are correctly propagated between adjacent contexts.

To address these challenges, we Skolemise existential quantifiers and transform axioms into DL-clauses. Skolemisation introduces function symbols that act as names for successors. Our clauses thus contain terms of the form x, $f_i(x)$, and y which have a special meaning in our setting: variable x represents the elements that a context stands for; $f_i(x)$ represents a successor of x; and y represents the predecessor of x. This allows us to represent constraint (52) as

$$f_2(x) \approx y \vee f_3(x) \approx y \vee f_3(x) \approx f_2(x). \tag{53}$$

Table 2 shows the inference rules of our calculus that are applicable to such a clausal representation. In each clause, literals are ordered from the smallest to the largest, and so the maximal literal is always on the right; moreover, clause numbers correspond to the order of clause derivation. In the rest of this section, we discuss the rules on our running example and show how they verify $\mathcal{O}_2 \models B_0(x) \rightarrow B_4(x)$; for brevity, we present only the inferences needed to produce the desired conclusion.

We first create context v_1 and initialise it with (26); this ensures that each interpretation represented by the context structure contains an element for which B_0 holds. Next, we derive (27) and (28) using hyperresolution. At this point, we could hyperresolve (22) and (28) to obtain $\top \rightarrow S(f_1(x), f_2(f_1(x)))$; however, such inferences could easily lead to nontermination of the calculus due to increased term nesting. Therefore, we require hyperresolution to map variable x in the DL-clauses to variable x in the context clauses; thus, in each context, hyperresolution derives only consequences about x.

Function symbol f_1 in clauses (27) and (28) is akin to an existential quantifier; consequently, the Succ rule introduces a fresh context v_2. Due to Skolemisation, edges in our context structure are labelled with function symbols, rather than concepts of the form $\exists S.B$ as in [23]. The rule uses an expansion strategy analogous to the $\mathcal{EL}$ strategy from Section 3. To determine which information to propagate to a successor, Definition 2 given below introduces a set $\mathsf{Su}(\mathcal{O})$ of *successor triggers*. In our example, DL-clause (25) contains atoms $B_1(x)$ and $S(x, z_i)$ in its body, where z_i can be mapped to a predecessor or a successor of x, so a context in which hyperresolution is applied to (25) will be interested in information about its predecessors; we reflect this by adding $B_1(x)$ and $S(x, y)$ to $\mathsf{Su}(\mathcal{O})$. Thus, the Succ rule introduces context v_2, sets its core to $B_1(x)$ and $S(x, y)$, and initialises the context with (30) and (31).

We next introduce (32)–(35) using hyperresolution, at which point we have sufficient information to apply hyperresolution to (25) to derive (36). Please note how the presence of (30) is crucial for this inference. We use paramodulation to deal with equality in clause (36). As is common in resolution-based theorem proving, we order the literals in a clause and apply inferences only to maximal literals; thus, we derive (37).

Clauses (32), (33), and (37) contain function symbol f_2, so we again apply the Succ rule and introduce context v_3. Due to clause (33), we know that $B_2(x)$ must always hold in v_2, so we add $B_2(x)$ to core_{v_2}. However, $B_3(f_2(x))$ occurs in clause (37) in a disjunction, so it holds only conditionally in v_2; we reflect this by including $B_3(x)$ in the body of clause (41). This allows us derive (42) using hyperresolution.

Clause (42) essentially says '$B_3(f_2(x))$ should not hold in the predecessor', so the Pred rule propagates this information to v_2. This produces clause (46); one can intuitively understand this inference as hyperresolution of (37) and (42), where we take into account that term $f_2(x)$ in context v_2 is represented as variable x in context v_3.

After two paramodulation steps, we derive clause (48), which essentially says 'the predecessor must satisfy $B_2(x)$ or $B_3(x)$'. The set $\mathsf{Pr}(\mathcal{O})$ of *predecessor triggers* from Definition 2 identifies this information as relevant to v_1: the DL-clauses in (24) contain $B_2(x)$ and $B_3(x)$ in their bodies, which are represented in v_2 as $B_2(y)$ and $B_3(y)$. Hence $\mathsf{Pr}(\mathcal{O})$ contains $B_2(y)$ and $B_3(y)$, which allows the Pred rule to derive (49).

After two more hyperresolution steps, we finally derive our target clause (51). Please note, however, that we cannot derive this if $B_4(x)$ were maximal in (50); thus, for completeness we require all atoms in the head of a query clause to be smallest. A similar observation applies to $\mathsf{Pr}(\mathcal{O})$: if $B_3(y)$ were maximal in (47), we would not derive (48) and propagate it to v_1; thus, we require all atoms in $\mathsf{Pr}(\mathcal{O})$ to be smallest too.

5 Formalising the Consequence-Based Algorithm for $\mathcal{SHIQ}$

Our calculus manipulates *context clauses*, which are constructed from *context terms* and *context literals* as described in Definition 1. Unlike in general resolution, we restrict context clauses to contain only variables x and y, which have a special meaning in our setting: variable x represents a point (i.e., a term) in the model, and y represents the predecessor of x; this naming convention is important for the rules of our calculus. This is in contrast to the DL-clauses of an ontology: these can contain variables x and z_i, and the latter can refer to either the predecessor or a successor of x.

Definition 1. *A* context $\mathcal{F}$-term *is a term of the form x, y, or $f(x)$ for $f \in \mathcal{F}$; a* context $\mathcal{P}$-term *is a term of the form $B(y)$, $B(x)$, $B(f(x))$, $S(x,y)$, $S(y,x)$, $S(x,f(x))$, or $S(f(x),x)$ for $B, R \in \mathcal{P}$ and $f \in \mathcal{F}$; and a* context term *is an $\mathcal{F}$-term or a $\mathcal{P}$-term. A* context literal *is a literal of the form $A \approx \mathsf{t}$ (called a* context atom*), $f(x) \bowtie g(x)$, or $f(x) \bowtie y$, for A a context $\mathcal{P}$-term and $\bowtie \in \{\approx, \not\approx\}$. A* context clause *is a clause with only function-free context atoms in the body, and only context literals in the head.*

Definition 2 introduces sets $\mathsf{Su}(\mathcal{O})$ and $\mathsf{Pr}(\mathcal{O})$, that identify the information that must be exchanged between adjacent contexts. Intuitively, $\mathsf{Su}(\mathcal{O})$ contains atoms that are of interest to a context's successor, and it guides the Succ rule whereas $\mathsf{Pr}(\mathcal{O})$ contains atoms that are of interest to a context's predecessor and it guides the Pred rule.

Definition 2. *Let $\mathcal{O}$ be an ontology. The set $\mathsf{Su}(\mathcal{O})$ of* successor triggers *of $\mathcal{O}$ is the smallest set of atoms such that, for each clause $\Gamma \to \Delta \in \mathcal{O}$, we have (i) $B(x) \in \Gamma$ implies $B(x) \in \mathsf{Su}(\mathcal{O})$, (ii) $S(x,z_i) \in \Gamma$ implies $S(x,y) \in \mathsf{Su}(\mathcal{O})$, and (iii) $S(z_i,x) \in \Gamma$ implies $S(y,x) \in \mathsf{Su}(\mathcal{O})$. The set $\mathsf{Pr}(\mathcal{O})$ of* predecessor triggers *is defined as*

$$\mathsf{Pr}(\mathcal{O}) = \{\, A\{x \mapsto y,\ y \mapsto x\} \mid A \in \mathsf{Su}(\mathcal{O}) \,\} \cup \{\, B(y) \mid B \text{ occurs in } \mathcal{O} \,\}.$$

Similarly to resolution, our calculus uses a term order $\succ$ to restrict its inferences. Definition 3 specifies the conditions that the order must satisfy. Conditions 1 and 2 ensure that $\mathcal{F}$-terms are compared uniformly across contexts; however, $\mathcal{P}$-terms can be compared in different ways in different contexts. Conditions 1 through 4 ensure that, when grounded with x and y mapping to a term its predecessor, respectively, the order is a *simplification order* [4]—a kind of term order commonly used in equational theorem proving. Finally, condition 5 ensures that atoms that might be propagated to a context's predecessor via the Pred rule are smallest, which is important for completeness.

Definition 3. *Let $\succcurlyeq$ be a total, well-founded order on function symbols. A* context term order $\succ$ *is an order on context terms satisfying the following conditions:*

1. *for each $f \in \mathcal{F}$, we have $f(x) \succ x \succ y$;*
2. *for all $f, g \in \mathcal{F}$ with $f \succcurlyeq g$, we have $f(x) \succ g(x)$;*
3. *for all terms s_1, s_2, and t and each position p in t, if $s_1 \succ s_2$, then $t[s_1]_p \succ t[s_2]_p$;*
4. *for each term s and each proper position p in s, we have $s \succ s|_p$; and*
5. *for each atom $A \approx \mathsf{t} \in \mathsf{Pr}(\mathcal{O})$ and each context term $s \notin \{x, y\}$, we have $A \not\succ s$.*

Order $\succ$ is extended to literals, also written $\succ$, as described in Section 2.

A lexicographic path order (LPO) [4] over context $\mathcal{F}$-terms and context $\mathcal{P}$-terms, in which x and y are treated as constants such that $x \succ y$, satisfies conditions 1 through 4. Furthermore, $\mathsf{Pr}(\mathcal{O})$ contains only atoms of the form $B(y)$, $S(x, y)$, and $S(y, x)$, which can always be smallest in the order; thus, condition 5 does not contradict the other conditions. Hence, an LPO that is relaxed for condition 5 satisfies Definition 3. In addition to orders, redundancy elimination techniques have proven crucial to the practical effectiveness of resolution calculi. We now define a criterion compatible with our setting.

Definition 4. *A set of clauses U* contains a clause $\Gamma \to \Delta$ up to redundancy*, written $\Gamma \to \Delta \mathrel{\hat{\in}} U$, if*

1. *terms s and s' exist such that $s \approx s \in \Delta$ or $\{s \approx s',\ s \not\approx s'\} \subseteq \Delta$, or*
2. *a clause $\Gamma' \to \Delta' \in U$ exist such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$.*

Proposition 1. *For U a set of clauses and $C \in U$ a clause such that $C \mathrel{\hat{\in}} U \setminus \{C\}$, for each clause C' with $C' \mathrel{\hat{\in}} U$, we have $C' \mathrel{\hat{\in}} U \setminus \{C\}$.*

Proposition 1 shows that our calculus is compatible with backward subsumption (which is captured in the Elim rule). Note that tautologies of the form $A \to A$ are *not necessarily* redundant since they can be used to initialise contexts. However, if our calculus were to derive both $A \to A$ and $A \to A \vee A'$ then the latter is always redundant.

We now formalise the notion of a context structure, and define soundness for a context structure. The latter captures the fact that core_v is not contained in the body of any context clause in $\mathcal{S}_v$.

Definition 5. *A* context structure *for an ontology $\mathcal{O}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$, where $\mathcal{V}$ is a finite set of* contexts*, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{F}$ is a finite set of edges labelled with function symbols, function* core *assigns to each context v a conjunction core_v of atoms over the $\mathcal{P}$-terms from $\mathsf{Su}(\mathcal{O})$, function $\mathcal{S}$ assigns to each context v a finite set $\mathcal{S}_v$ of context clauses, and function $\succ$ assigns to each context v a context term order $\succ_v$. A context structure $\mathcal{D}$ is* sound *for $\mathcal{O}$ if the following conditions both hold:*

S1. $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta$ for each context $v \in \mathcal{V}$ and each clause $\Gamma \to \Delta \in \mathcal{S}_v$, and
S2. $\mathcal{O} \models \mathsf{core}_u \to \mathsf{core}_v\{x \mapsto f(x), y \mapsto x\}$ for each edge $\langle u, v, f \rangle \in \mathcal{E}$.

Definition 6 introduces an expansion strategy—a parameter of our calculus that determines when and how to reuse contexts in order to satisfy existential restrictions. We have discussed the roles of expansion strategies in Section 3; moreover, in [23] we presented several expansion strategies for the DLs contained in $\mathcal{ALCI}$, and adapting these to $\mathcal{SHIQ}$ is straightforward.

Table 2. The rules of the consequence-based calculus for $\mathcal{SHIQ}$

Core	If	$A \in \mathsf{core}_v$,
		and $\top \to A \notin \mathcal{S}_v$,
	then	add $\top \to A$ to $\mathcal{S}_v$.
Hyper	If	$\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$,
		σ is a substitution such that $\sigma(x) = x$,
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succeq_v A_i\sigma$ for $i \in \{1, \ldots, n\}$,
		and $\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i \;\hat{\notin}\; \mathcal{S}_v$,
	then	add $\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i$ to $\mathcal{S}_v$.
Eq	If	$\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \in \mathcal{S}_v$ with $s_1 \succ_v t_1$ and $\Delta_1 \not\succeq_v s_1 \approx t_1$,
		$\Gamma_2 \to \Delta_2 \vee s_2 \bowtie t_2 \in \mathcal{S}_v$ with $\bowtie \in \{\approx, \not\approx\}$ and $s_2 \succ_v t_2$ and $\Delta_2 \not\succeq_v s_2 \bowtie t_2$,
		$s_2\vert_p = s_1$,
		and $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2 \;\hat{\notin}\; \mathcal{S}_v$,
	then	add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2$ to $\mathcal{S}_v$.
Ineq	If	$\Gamma \to \Delta \vee t \not\approx t \in \mathcal{S}_v$ with $\Delta \not\succeq_v t \not\approx t$,
		and $\Gamma \to \Delta \;\hat{\notin}\; \mathcal{S}_v$,
	then	add $\Gamma \to \Delta$ to $\mathcal{S}_v$.
Factor	If	$\Gamma \to \Delta \vee s \approx t \vee s \approx t' \in \mathcal{S}_v$ with $\Delta \cup \{s \approx t\} \not\succeq_v s \approx t'$ and $s \succ_v t'$,
		and $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t' \;\hat{\notin}\; \mathcal{S}_v$,
	then	add $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t'$ to $\mathcal{S}_v$.
Elim	If	$\Gamma \to \Delta \in \mathcal{S}_v$ and
		$\Gamma \to \Delta \;\hat{\in}\; \mathcal{S}_v \setminus \{\Gamma \to \Delta\}$
	then	remove $\Gamma \to \Delta$ from $\mathcal{S}_v$.
Pred	If	$\langle u, v, f\rangle \in \mathcal{E}$,
		$\bigwedge_{i=1}^{m} A_i \to \bigvee_{i=m+1}^{m+n} A_i \in \mathcal{S}_v$,
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_u$ with $\Delta_i \not\succeq_u A_i\sigma$ for $1 \leq i \leq m$,
		$A_i \in \mathsf{Pr}(\mathcal{O})$ for each $m+1 \leq i \leq m+n$,
		and $\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} A_i\sigma \;\hat{\notin}\; \mathcal{S}_u$,
	then	add $\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} A_i\sigma$ to $\mathcal{S}_u$;
	where	$\sigma = \{x \mapsto f(x), y \mapsto x\}$.
Succ	If	$\Gamma \to \Delta \vee A \in \mathcal{S}_u$ where $\Delta \not\succeq_u A$ and A contains $f(x)$, and
		no edge $\langle u, v, f\rangle \in \mathcal{E}$ exists such that $A \to A \;\hat{\in}\; \mathcal{S}_v$ for each $A \in K_2 \setminus \mathsf{core}_v$,
	then	let $\langle v, \mathsf{core}', \succ'\rangle := \mathsf{strategy}(K_1, \mathcal{D})$;
		if $v \in \mathcal{V}$, then let $\succ_v := \succ_v \cap \succ'$, and
		otherwise let $\mathcal{V} := \mathcal{V} \cup \{v\}$, $\mathsf{core}_v := \mathsf{core}'$, $\succ_v := \succ'$, and $\mathcal{S}_v := \emptyset$;
		add the edge $\langle u, v, f\rangle$ to $\mathcal{E}$; and
		add $A \to A$ to $\mathcal{S}_v$ for each $A \in K_2 \setminus \mathsf{core}_v$;
	where	$\sigma = \{x \mapsto f(x), y \mapsto x\}$,
		$K_1 = \{\, A \in \mathsf{Su}(\mathcal{O}) \mid \top \to A\sigma \in \mathcal{S}_u \,\}$, and
		$K_2 = \{\, A \in \mathsf{Su}(\mathcal{O}) \mid \Gamma' \to \Delta' \vee A\sigma \in \mathcal{S}_u$ and $\Delta' \not\succeq_u A\sigma \,\}$.

Definition 6. *An* expansion strategy *is a polynomial-time computable function* $\mathsf{strategy}$ *that takes a set of atoms* K *and a context structure* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$*. The result of* $\mathsf{strategy}(K, \mathcal{D})$ *is a triple* $\langle v, \mathsf{core}', \succ' \rangle$ *where* core' *is a subset of* K*; either* $v \notin \mathcal{V}$ *is a fresh context, or* $v \in \mathcal{V}$ *is an existing context in* $\mathcal{D}$ *such that* $\mathsf{core}_v = \mathsf{core}'$*; and* $\succ'$ *is a context term order.*

We now present the main theorems. Full proofs of all technical results can be found in [5]. Theorem 1 proves that all clauses derived by our calculus are indeed conclusions of the input ontology, and Theorem 2 is our statement of completeness.

Theorem 1. *For any strategy, applying a rule from Table 2 to an ontology* $\mathcal{O}$ *and a context structure* $\mathcal{D}$ *that is sound for* $\mathcal{O}$ *produces a context structure that is sound for* $\mathcal{O}$*.*

Theorem 2. *Let* $\mathcal{O}$ *be an ontology, and let* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$ *be a context structure such that no inference rule from Table 2 is applicable to* $\mathcal{O}$ *and* $\mathcal{D}$*. Then, for each query clause* $\Gamma_Q \to \Delta_Q$ *and each context* $q \in \mathcal{V}$ *that satisfy all of the following conditions, we have* $\Gamma_Q \to \Delta_Q \mathrel{\hat{\in}} \mathcal{S}_q$*.*

C1. $\mathcal{O} \models \Gamma_Q \to \Delta_Q$*.*
C2. For each atom $A \approx \mathsf{t} \in \Delta_Q$ *and each context term* $s \notin \{x, y\}$ *such that* $A \succ_q s$*, we have* $s \approx \mathsf{t} \in \Delta_Q \cup \mathsf{Pr}(\mathcal{O})$*.*
C3. For each $A \in \Gamma_Q$*, we have* $\Gamma_Q \to A \mathrel{\hat{\in}} \mathcal{S}_q$*.*

Theorems 1 and 2 result in the following algorithm for deciding $\mathcal{O} \models \Gamma_Q \to \Delta_Q$.

1. Create an empty context structure $\mathcal{D}$, introduce a context q, and, for each $A \in \Gamma_Q$, add $\Gamma_Q \to A$ to $\mathcal{S}_q$ in order to satisfy condition C3.
2. Initialise $\succ_q$ in a way that satisfies condition C2, and select an expansion strategy.
3. Saturate $\mathcal{D}$ and $\mathcal{O}$ using the inference rules from Table 2.
4. $\Gamma_Q \to \Delta_Q$ holds if and only if $\Gamma_Q \to \Delta_Q \mathrel{\hat{\in}} \mathcal{S}_v$.

Proposition 2. *For each expansion strategy that introduces at most exponentially many contexts, the consequence-based calculus for* $\mathcal{SHIQ}$ *is worst-case optimal.*

Proof. The number $\wp$ of different context clauses that can be generated using the symbols in $\mathcal{O}$ is clearly at most exponential in the size of $\mathcal{O}$, and the number m of clauses participating in each inference is linear in the size of $\mathcal{O}$. Hence, with k contexts, the number of inferences is bounded by $(k \cdot \wp)^m$; if k is at most exponential in the size of $\mathcal{O}$, the number of inferences is exponential as well. Thus, if at most exponentially many contexts are introduced, our algorithm runs in exponential time, which is worst-case optimal for $\mathcal{SHIQ}$ [3]. □

6 Conclusion

We have presented the first consequence based calculus for $\mathcal{SHIQ}$, a DL that includes both disjunction and counting quantifiers. Our calculus combines ideas from state of the art resolution and (hyper)tableau calculi, including the use of ordered paramodulation for efficient equality reasoning. In spite of its increased complexity, the calculus mimics existing and well proven $\mathcal{EL}$ calculi on $\mathcal{EL}$ ontologies. Thus, although implementation and evaluation remain as future work, we believe that the calculus is likely to work well on 'mostly-$\mathcal{EL}$' ontologies—a type of ontology that occurs commonly in practice.

References

1. H. Andréka, J. van Benthem, and I. Németi. Modal Languages and Bounded Fragments of Predicate Logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In L. Pack Kaelbling and A. Saffiotti, editors, *Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.
4. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
5. A. Bate, B. Motik, B. Cuenca Grau, F. Simančík, and I. Horrocks. Extending Consequence-Based Reasoning to $\mathcal{SHIQ}$. Technical report, Department of Computer Science, University of Oxford, May 2015.
6. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA '96)*, number 1126 in LNAI, pages 1–17, Évora, Portugal, September 30–October 3 1996. Springer.
7. P. Baumgartner and R. A. Schmidt. Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 125–139, Seattle, WA, USA, August 17–20 2006. Springer.
8. S. Demri and H. de Nivelle. Deciding Regular Grammar Logics with Converse Through First-Order Logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.
9. Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.
10. Rajeev Goré and Linh Anh Nguyen. EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies. In Nicola Olivetti, editor, *Proc. of the 16th Int. Conf. on Automated Reasoning with Tableaux and Related Methods (TABLEAUX 2007)*, volume 4548 of *LNCS*, pages 133–148, Aix en Provence, France, July 3–6 2007. Springer.
11. E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *Proc. of the 12th IEEE Symposium on Logic in Computer Science (LICS '97)*, pages 306–317, Warsaw, Poland, June 29–July 2 1997. IEEE Computer Society.
12. V. Haarslev and R. Möller. RACER System Description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *LNAI*, pages 701–706, Siena, Italy, June 18–23 2001. Springer.
13. U. Hustadt and R.A. Schmidt. On the Relation of Resolution and Tableaux Proof Systems for Description Logics. In Thomas Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI 1999)*, pages 110–117, Stockholm, Sweden, July 31 – August 6 1999. Morgan Kaufmann.
14. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Deciding Expressive Description Logics in the Framework of Resolution. *Information & Computation*, 206(5):579–601, 2008.
15. Y. Kazakov. Consequence-Driven Reasoning for Horn SHIQ Ontologies. In Craig Boutilier, editor, *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, pages 2040–2045, Pasadena, CA, USA, July 11–17 2009.
16. B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.
17. R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19(4):312–351, 1995.

18. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL*, 8(3):265–292, 2000.
19. M. Ortiz, S. Rudolph, and M. Simkus. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proc. of the 12th Int. Conf. on Knowledge Representation and Reasoning (KR 2010)*, pages 269–279, Toronto, ON, Canada, May 9–13 2010. AAAI Press.
20. A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Communications*, 15(2–3):91–110, 2002.
21. R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In F. Baader, editor, *Proc. of the 19th Int. Conf. on Automated Deduction (CADE-19)*, volume 2741 of *LNAI*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003. Springer.
22. S. Schulz. E—A Brainiac Theorem Prover. *AI Communications*, 15(2–3):111–126, 2002.
23. František Simančík, Boris Motik, and Ian Horrocks. Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics. *Artificial Intelligence*, 209:29–77, 2014.
24. F. Simančík, Y. Kazakov, and I. Horrocks. Consequence-Based Reasoning beyond Horn Ontologies. In Toby Walsh, editor, *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*, pages 1093–1098, Barcelona, Spain, July 16–22 2011.
25. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.
26. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292–297, Seattle, WA, USA, August 17–20 2006. Springer.

Explaining Query Answers under Inconsistency-Tolerant Semantics over Description Logic Knowledge Bases (Extended Abstract)

Meghyn Bienvenu[1], Camille Bourgaux[1], and François Goasdoué[2]

[1]LRI, CNRS & Université Paris-Sud, Orsay, France
[2]IRISA, Université de Rennes 1, Lannion, France

1 Explaining Query Results

The problem of querying description logic (DL) knowledge bases (KBs) using database-style queries (in particular, conjunctive queries) has been a major focus of recent DL research. Since scalability is a key concern, much of the work has focused on lightweight DLs for which query answering can be performed in polynomial time w.r.t. the size of the ABox. The DL-Lite family of lightweight DLs [10] is especially popular due to the fact that query answering can be reduced, via query rewriting, to the problem of standard database query evaluation.

Since the TBox is usually developed by experts and subject to extensive debugging, it is often reasonable to assume that its contents are correct. By contrast, the ABox is typically substantially larger and subject to frequent modifications, making errors almost inevitable. As such errors may render the KB inconsistent, several inconsistency-tolerant semantics have been introduced in order to provide meaningful answers to queries posed over inconsistent KBs. Arguably the most well-known is the *AR semantics* [17], inspired by work on consistent query answering in databases (cf. [4] for a survey). Query answering under AR semantics amounts to considering those answers (w.r.t. standard semantics) that can be obtained from every *repair*, the latter being defined as an inclusion-maximal subset of the ABox that is consistent with the TBox. A more cautious semantics, called *IAR semantics* [17] queries the intersection of the repairs and provides a lower bound on AR semantics. The *brave* semantics [7], which considers the answers holding in some repair, provides a natural upper bound. This extended abstract presents our work [6] on explaining why a tuple is a (non-)answer to a query under AR, IAR, or brave semantics.

The need to equip reasoning systems with explanation services is widely acknowledged by the DL community. Indeed, there have been numerous works on *axiom pinpointing*, in which the objective is to identify (minimal) subsets of a KB that entail a given TBox axiom (or ABox assertion) [18, 9, 21, 16, 22, 20, 14, 15]. With regards to conjunctive queries (CQs), a proof-theoretic approach to explaining positive answers to CQs over DL-Lite$_\mathcal{A}$ KBs was introduced in [8], and, more recently, the problem of explaining negative query answers over DL-Lite$_\mathcal{A}$

KBs has been studied in [11–13]. Explanation facilities are all the more essential when using inconsistency-tolerant semantics, as recently argued in [1, 2]. Indeed, the brave, AR, and IAR semantics allow query answers to be classified into three categories of increasing reliability, and a user may naturally wonder why a given tuple was assigned to, or excluded from, one of these categories. To help the user understand this classification, we introduce the notion of *explanation* for positive and negative query answers under brave, AR, and IAR semantics. Formally, the explanations we consider take either the form of a set of ABox assertions (viewed as a conjunction) or a set of sets of assertions (disjunction of conjunctions).

The simplest answers to explain are positive brave- and IAR-answers (i.e., answers that hold under brave, resp. IAR, semantics). For the former, we can use as explanations the query's *causes*, which are the minimal consistent sets of assertions that entail the answer together with the TBox, and for the latter, we consider the causes that do not participate in any contradictions. To explain why a tuple is an AR-answer, it is no longer sufficient to give a single cause since different repairs may use different causes. We therefore define explanations as (minimal) disjunctions of causes that 'cover' all repairs, i.e., minimal sets of causes such that every repair contains at least one of them. To explain negative AR-answers, the idea is to give a (minimal) subset of the ABox that is consistent with the TBox and contradicts every cause of the query, since any such subset can be extended to a repair that omits all causes. For negative IAR-answers, we need only ensure that every cause is contradicted by some consistent subset.

When there are a large number of explanations for a given result, it may be impractical to present them all to the user. In such cases, one may choose instead to rank the explanations according to some preference criteria, and to present one or a small number of most *preferred explanations*. In the present work, we use *cardinality* to rank explanations for brave- and IAR-answers and negative AR- and IAR-answers. For positive AR-answers, we consider two ways of ranking explanations: the *number of disjuncts*, since fewer disjuncts requires less case-based reasoning, and the total *number of assertions*, to favour disjunctions of causes that share assertions. A complementary approach is to concisely summarize the set of explanations in terms of the *necessary assertions* (that occur in *every* explanation) and the *relevant assertions* (occurring in *some* explanation).

2 Complexity Results and Connections to SAT

In addition to the problem of computing explanations, we consider four natural decision problems: decide whether a given assertion appears in some explanation (REL) or in every explanation (NEC), decide whether a candidate is an explanation (REC), resp. a best explanation according a given criteria (BEST REC). For our study, we consider ontologies formulated in the lightweight logic DL-Lite$_{\mathcal{R}}$ that underlies the OWL 2 QL profile [19].

The results of our complexity analysis are displayed in Figure 1. For the explanation tasks that are shown to be intractable, we have exhibited tight connections with variants of propositional satisfiability that enable us to exploit

	brave, IAR	AR	neg. IAR	neg. AR
REL	in P	Σ_2^p-co	in P	NP-co
NEC	in P	NP-co	in P	coNP-co
REC	in P	BH_2-co	in P	in P
BEST REC†	in P	Π_2^p-co‡	coNP-co*	coNP-co*

† upper bounds hold for ranking criteria that can be decided in P
‡ Π_2^p-hard for smallest disjunction or fewest assertions
* coNP-hard for cardinality-minimal explanations

Fig. 1: Data complexity results for conjunctive queries.

facilities of modern SAT solvers. We use the encoding $\varphi_{\neg q} \wedge \varphi_{cons}$ introduced in [5] which is unsatisfiable iff the corresponding answer is entailed under AR semantics. Intuitively, $\varphi_{\neg q}$ gives the ways of contradicting every cause, and φ_{cons} enforces consistency. We can show that the explanations for positive AR-answers correspond to the minimal unsatisfiable subsets of $\varphi_{\neg q}$ w.r.t. φ_{cons}, while the smallest explanations for negative AR-answers (resp. negative IAR-answers) correspond to the cardinality-minimal models of $\varphi_{\neg q} \wedge \varphi_{cons}$ (resp. $\varphi_{\neg q}$).

3 System and Experiments

We extended the CQAPri system [5] to implement our framework, relying on the SAT4J SAT solver to compute minimal unsatisfiable subsets and cardinality-minimal models [3]. Our prototype runs in two modes: either it explains *some* selected query answers, or *all* the answers as they are being computed. These answers are divided into three classes: *Possible* (brave-answers not entailed under the AR semantics), *Likely* (AR-answers not entailed under IAR semantics), and *Sure* (IAR-answers). Concretely, explaining an answer $\mathbf{a}$ consists in providing, for the relevant semantics S, S' according to the class of $\mathbf{a}$: (i) *all* explanations of $\mathbf{a}$ being an S-answer, as well as necessary and relevant assertions, and (ii) *one* smallest explanation of $\mathbf{a}$ not being an S'-answer, with necessary and relevant assertions when $S' = IAR$, and necessary assertions when $S' = AR$ together with necessary and relevant assertions for explaining $\mathbf{a}$ not being an IAR-answer. Positive explanations are ranked as explained in Section 1.

The experimental evaluation of our prototype system over the slightly modified CQAPri benchmark shows that explanations of query (non-)answers can be generated very quickly (typically less than 1ms), although we did find some rare difficult cases for which computing a smallest explanation for a negative answer is long (more than 1h). Finally, we observed that the average number of explanations per answer is often reasonably low, although some answers have a large number of explanations (e.g., 654 for an IAR-answer, 243 for an AR-answer, and 693 for a brave-answer), showing the practical interest of presenting such explanations in a concise way.

Acknowledgements This work was supported contract ANR-12-JS02-007-01.

References

1. Arioua, A., Tamani, N., Croitoru, M.: On conceptual graphs and explanation of query answering under inconsistency. In: Proc. of ICCS (2014)
2. Arioua, A., Tamani, N., Croitoru, M., Buche, P.: Query failure explanation in inconsistent knowledge bases using argumentation. In: Proc. of COMMA (2014)
3. Berre, D.L., Parrain, A.: The sat4j library, release 2.2. JSAT 7(2-3), 59–64 (2010)
4. Bertossi, L.E.: Database Repairing and Consistent Query Answering. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2011)
5. Bienvenu, M., Bourgaux, C., Goasdoué, F.: Querying inconsistent description logic knowledge bases under preferred repair semantics. In: Proc. of AAAI (2014)
6. Bienvenu, M., Bourgaux, C., Goasdoué, F.: Explaining query answers under inconsistency-tolerant semantics over description logic knowledge bases (2015), Technical Report 1580, LRI, Orsay, France. Available at `https://www.lri.fr/~bibli/Rapports-internes/2015/RR1580.pdf`
7. Bienvenu, M., Rosati, R.: Tractable approximations of consistent query answering for robust ontology-based data access. In: Proc. of IJCAI (2013)
8. Borgida, A., Calvanese, D., Rodriguez-Muro, M.: Explanation in the DL-Lite family of description logics. In: Proc. of OTM (2008)
9. Borgida, A., Franconi, E., Horrocks, I.: Explaining ALC subsumption. In: Proc. of ECAI (2000)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning (JAR) 39(3), 385–429 (2007)
11. Calvanese, D., Ortiz, M., Simkus, M., Stefanoni, G.: Reasoning about explanations for negative query answers in DL-Lite. J. Artif. Intell. Res. (JAIR) 48, 635–669 (2013)
12. Du, J., Wang, K., Shen, Y.: A tractable approach to abox abduction over description logic ontologies. In: Proc. of AAAI (2014)
13. Du, J., Wang, K., Shen, Y.: Towards tractable and practical abox abduction over inconsistent description logic ontologies. In: Proc. of AAAI (2015)
14. Horridge, M., Bail, S., Parsia, B., Sattler, U.: The cognitive complexity of OWL justifications. In: Proc. of ISWC (2011)
15. Horridge, M., Parsia, B., Sattler, U.: Extracting justifications from bioportal ontologies. In: Proc. of ISWC (2012)
16. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in OWL ontologies. J. Web Sem. 3(4), 268–293 (2005)
17. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. of RR (2010)
18. McGuinness, D.L., Borgida, A.: Explaining subsumption in description logics. In: Proc. of IJCAI (1995)
19. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles. W3C Recommendation (11 December 2012), available at `http://www.w3.org/TR/owl2-profiles/`
20. Peñaloza, R., Sertkaya, B.: Complexity of axiom pinpointing in the dl-lite family of description logics. In: Proc. of ECAI (2010)
21. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. of IJCAI (2003)
22. Sebastiani, R., Vescovi, M.: Axiom pinpointing in lightweight description logics via horn-sat encoding and conflict analysis. In: Proc. of CADE (2009)

Combined Complexity of Answering Tree-like Queries in OWL 2 QL

Meghyn Bienvenu[1], Stanislav Kikot[2], and Vladimir Podolskii[3]

[1] LRI – CNRS & Université Paris Sud, Orsay, France
[2] Institute for Information Transmission Problems & MIPT, Moscow, Russia
[3] Steklov Mathematical Institute & Higher School of Economics, Moscow, Russia

Introduction The OWL 2 QL ontology language [11], based upon the description logic *DL-Lite*$_R$, is considered particularly well suited for applications involving large amounts of data. While the data complexity of querying OWL 2 QL knowledge bases is well understood, far less is known about combined complexity of conjunctive query (CQ) answering for restricted classes of conjunctive queries. By contrast, the combined complexity of CQ answering in the relational setting has been thoroughly investigated.

In relational databases, it is well known that CQ answering is NP-complete in the general case. A seminal result by Yannakakis established the tractability of answering tree-shaped (aka acyclic) CQs [14], and this result was later extended to wider classes of queries, most notably to bounded treewidth CQs [5]. Gottlob et al. [6] pinpointed the precise complexity of answering tree-shaped and bounded treewidth CQs, showing both problems to be complete for the class LOGCFL of all languages logspace-reducible to context-free languages [13]. In the presence of arbitrary OWL 2 QL ontologies, the NP upper bound for arbitrary CQs continues to hold [4], but answering tree-shaped queries becomes NP-hard [8]. Interestingly, the latter problem was recently proven tractable in [3] for DL-Lite$_{\mathsf{core}}$ (a slightly less expressive logic than OWL 2 QL), raising the hope that other restrictions might also yield tractability.

This extended abstract summarizes our investigation [2] into the combined complexity of conjunctive query answering in OWL 2 QL for tree-shaped queries, their restriction to linear and bounded leaf queries and their generalization to bounded treewidth queries. Our complexity analysis reveals that all query-ontology combinations that have not already been shown NP-hard are in fact tractable. Specifically, in the case of bounded depth ontologies, we prove membership in LOGCFL for bounded treewidth queries (generalizing the result in [6]) and membership in NL for bounded leaf queries. We also show LOGCFL-completeness for linear and bounded leaf queries in the presence of arbitrary OWL 2 QL ontologies. This last result is the most interesting technically, as the upper and lower bounds rely on two different characterizations of the class LOGCFL.

Preliminaries We assume the reader familiar is OWL 2 QL (or *DL-Lite*$_R$) knowledge bases (KBs), composed of a TBox $\mathcal{T}$ and ABox $\mathcal{A}$ built from countably infinite, mutually disjoint sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of *concept names*, *role names*, and *individual names*. *Roles* R and *basic concepts* B are defined in a standard way, cf. [4]. We use $\mathsf{N}_R^{\pm}$ to refer to the set of all roles. We recall that every consistent *OWL 2 QL* KB $(\mathcal{T}, \mathcal{A})$ possesses a *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ with the property that $\mathcal{T}, \mathcal{A} \models \mathbf{q}(\mathbf{a})$ iff $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \mathbf{q}(\mathbf{a})$ for every CQ $\mathbf{q}$ and tuple $\mathbf{a} \subseteq \mathsf{inds}(\mathcal{A})$. Thus, CQ answering in *OWL 2 QL* corresponds to *deciding the existence of a homomorphism of the query into the canonical model*. Informally,

$\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is obtained from $\mathcal{A}$ by repeatedly applying the axioms in $\mathcal{T}$, introducing fresh elements as needed to serve as witnesses for the existential quantifiers. According to the standard construction (cf. [10]), the domain $\Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ consists of $\mathsf{inds}(\mathcal{A})$ and all words of the form $aR_1R_2 \dots R_{n-1}R_n$ ($n \geq 1$) with $a \in \mathsf{N_C}$ and $R_i \in \mathsf{N}_R^{\pm}$. Intuitively, the element $aR_1R_2 \dots R_{n-1}R_n$ is obtained by applying an axiom with right-hand side $\exists R_n$ to the element $aR_1R_2 \dots R_{n-1} \in \Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$. A TBox $\mathcal{T}$ is *of depth* ω if there is an ABox $\mathcal{A}$ such that the domain of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is infinite; $\mathcal{T}$ is *of depth* d, $0 \leq d < \omega$, if d is the greatest number such that some $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ contains an element of the form $aR_1 \dots R_d$.

Contributions In what follows, we briefly formulate our combined complexity results and provide some intuitions about the proof techniques. See [2] for details.

Theorem 1. *CQ answering is in* LOGCFL *for bounded treewidth queries and bounded depth ontologies.*

Proof sketch. We exploit the fact that CQ answering over a KB $(\mathcal{T}, \mathcal{A})$ corresponds to evaluating the query over the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ viewed as a database. If $\mathcal{T}$ has depth k (with k a fixed constant), then $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ can be computed by a deterministic logspace Turing machine (TM) with access to an NL oracle. Indeed, the depth bound k implies the finiteness of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ and that all domain elements can be described using logarithmically many bits. To complete the argument, we use the fact that answering bounded treewidth queries over databases is in LOGCFL [7] and that the class LOGCFL is closed under $\mathrm{L}^{\mathrm{LOGCFL}}$ (and hence L^{NL}) reductions [13]. □

Theorem 2. *CQ answering is* NL*-complete for bounded leaf queries and bounded depth ontologies.*

Proof sketch. The lower bound is an immediate consequence of the NL-hardness of answering atomic queries in OWL 2 QL. To prove the upper bound, we apply a straightforward non-deterministic procedure for deciding $(\mathcal{T}, \mathcal{A}) \models q$:

1. Fix a directed tree T compatible with q. Let v_0 be the root variable.
2. Guess $u_0 \in \Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$. Return **no** if v_0 *cannot be mapped* to u_0.
3. Initialize Frontier to $\{(v_0, u_0)\}$.
4. While Frontier $\neq \emptyset$
   (a) Remove (v_1, u_1) from Frontier.
   (b) For every child v_2 of v_1
      i. Guess an element u_2 from $\Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$.
      ii. Return **no** if (v_1, v_2) *cannot be mapped* to (u_1, u_2).
      iii. Add (v_2, u_2) to Frontier.
5. Return **yes**.

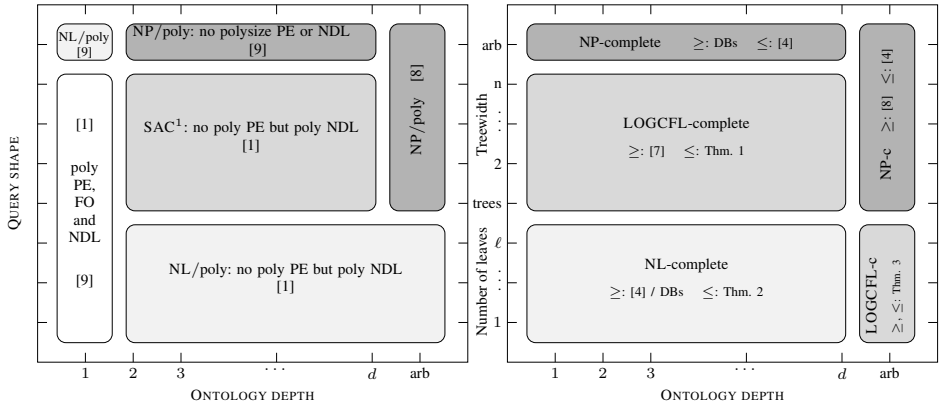
For lack of space, we have not specified how to check whether a variable (resp. pair of variables) can be mapped to an element (resp. pair of elements), but this can be done in NL using a small number of entailment checks. Also note that the bound on the number of leaves yields the bound on size of Frontier, and the bound on the TBox depth guarantees that we only need logarithmically many bits per pair in Frontier. □

Theorem 3. *CQ answering is* LOGCFL*-complete for bounded leaf queries and arbitrary ontologies. The lower bound holds already for linear queries.*

Proof sketch. Concerning the upper bound, it is easy to adapt the previous algorithm to handle arbitrary TBoxes: we simply replace $\Delta^{\mathcal{C}_\mathcal{T},\mathcal{A}}$ by $\{aw \in \Delta^{\mathcal{C}_\mathcal{T},\mathcal{A}} \mid |w| \leq 2|\mathcal{T}| + |q|\}$. The modified algorithm gives the correct answers, but it does not have the required complexity, because it might need more than logarithmically many bits to store guessed elements aw. To show LOGCFL membership, we further modify the procedure so that it can be implemented by a non-deterministic polytime logspace-bounded Turing machine augmented with a stack (such TMs are known to capture LOGCFL computation [12]). The stack is used to store the word part w of a domain element aw. The modification is not at all obvious since we need to store several words at a time while the specified machine has only a single stack; the trick is to employ a careful 'synchronization' of traversals of different branches of the query.

The lower bound is by reduction from the problem of deciding whether an input of length l is accepted by the lth circuit of a *logspace-uniform* family of SAC^1 circuits (proven LOGCFL-hard in [13]). This problem was used in [7] to show LOGCFL-hardness of evaluating tree-shaped CQs over databases. We follow a broadly similar approach, but with one crucial difference: the power of OWL 2 QL TBoxes allows us to 'unravel' the circuit into a tree and to use linear queries instead of tree-shaped ones. □

Discussion If we compare the new and existing results for OWL 2 QL with those from relational databases, we observe that adding an OWL 2 QL TBox of bounded depth does not change the combined complexity for query answering, while for TBoxes of unbounded depth, the complexity class shifts one 'step' higher: from NL to LOGCFL for bounded leaf queries and from LOGCFL to NP for tree-shaped and bounded treewidth CQs. It is also interesting to compare the combined complexity landscape (below right) with the succinctness landscape for query rewriting (below left) from [1].



Observe that for our newly identified tractable classes, polynomial-size non-recursive datalog (NDL) rewritings are guaranteed to exist, whereas this is not the case for the positive existential (PE) rewritings more typically considered. In future work, we plan to marry these positive succinctness and complexity results by developing concrete NDL-rewriting algorithms for OWL 2 QL for which both the rewriting and evaluation phases run in polynomial time (as was done in [3] for DL-Lite$_{\mathsf{core}}$).

Acknowledgments. Partial support was provided by ANR grant 12-JS02-007-01, Russian Foundation for Basic Research and the programme "Leading Scientific Schools".

References

1. M. Bienvenu, S. Kikot, and V. V. Podolskii. Succinctness of query rewriting in OWL 2 QL: the case of tree-like queries. In *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, pages 45–57, 2014.
2. M. Bienvenu, S. Kikot, and V. V. Podolskii. Tree-like queries in OWL 2 QL: Succinctness and complexity results. In *Proc. of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*. IEEE, 2015.
3. M. Bienvenu, M. Ortiz, M. Simkus, and G. Xiao. Tractable queries for lightweight description logics. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI 2013)*. AAAI Press, 2013.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *Theoretical Computer Science*, 239(2):211–229, 2000.
6. G. Gottlob, N. Leone, and F. Scarcello. Computing LOGCFL certificates. In *ICALP-99*, pages 361–371, 1999.
7. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001.
8. S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *Proc. of the 39th Int. Colloquium on Automata, Languages, and Programming (ICALP 2012), Part II*, volume 7392 of *LNCS*, pages 263–274. Springer, 2012.
9. S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. On the succinctness of query rewriting over OWL 2 QL ontologies with shallow chases. In *Proc. of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014)*. ACM Press, 2014.
10. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in DL-Lite. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press, 2010.
11. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language profiles. W3C Recommendation, 11 December 2012. Available at http://www.w3.org/TR/owl2-profiles/.
12. I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25(3):405–414, 1978.
13. H. Venkateswaran. Properties that characterize LOGCFL. *J. Computer and System Sciences*, 43(2):380–404, 1991.
14. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases (VLDB'81)*, pages 82–94. IEEE Computer Society, 1981.

Query-based comparison of OBDA specifications

Meghyn Bienvenu[1] and Riccardo Rosati[2]

[1] Laboratoire de Recherche en Informatique
CNRS & Université Paris-Sud, France

[2] Dipartimento di Ingegneria informatica, automatica e gestionale
Sapienza Università di Roma, Italy

Abstract. An ontology-based data access (OBDA) system is composed of one or more data sources, an ontology that provides a conceptual view of the data, and declarative mappings that relate the data and ontology schemas. In order to debug and optimize such systems, it is important to be able to analyze and compare OBDA specifications. Recent work in this direction compared specifications using classical notions of equivalence and entailment, but an interesting alternative is to consider query-based notions, in which two specifications are deemed equivalent if they give the same answers to the considered query or class of queries for all possible data sources. In this paper, we define such query-based notions of entailment and equivalence of OBDA specifications and investigate the complexity of the resulting analysis tasks when the ontology is formulated in *DL-Lite*$_R$.

1 Introduction

Ontology-based data access (OBDA) [13] is a recent paradigm that proposes the use of an *ontology* as a conceptual, reconciled view of the information stored in a set of existing *data sources*. The connection between the ontology and the data sources is provided by declarative *mappings*, that relate the elements of the ontology with the elements of the data sources. The ontology layer is the virtual interface used to access data, through *queries* over the elements of the ontology.

Due to the recent availability of techniques and systems for query processing in this setting [5, 14], the OBDA approach has recently started to be experimented in real applications (see e.g. [1, 7, 10]). In these projects, the construction, debugging and maintenance of the OBDA specification, consisting of the ontology, the schemas of the data sources, and the mapping, is a non-trivial task. Actually, the size and the complexity of the ontology and, especially, the mappings makes the management of such specifications a practical issue in these projects. Providing formal tools for supporting the above activities is therefore very important for the successful deployment of OBDA solutions.

In addition, the OBDA specification plays a major role in query answering, since the form of the specification may affect the system performance in answering queries: different, yet semantically equivalent specifications may give rise to very different execution times for the same query. So, the study of notions of equivalence and formal comparison of OBDA specifications is also important for optimizing query processing in OBDA systems. Indeed, some systems already implement forms of optimization based on transformations of the OBDA specification (an example is [14]).

So far, most of the work in OBDA has focused on query answering, often in a simplified setting without any mappings. Very little attention has been devoted to the formal analysis of OBDA specifications. The first approach that explicitly focuses on the formal analysis of OBDA specifications is [12], whose aim is the identification of semantic anomalies in mappings. Such an approach is based on a classical notion of logical equivalence and entailment between OBDA specifications. While it is very natural to resort to such classical notions, a significant alternative in many cases may be the adoption of *query-based* notions of equivalence and comparison, in which two specifications are compared with respect to a given query or a given class of queries, and are deemed equivalent if they give the same answers to the considered queries for all possible extensions of the data sources. This idea has been already explored in the data exchange and schema mapping literature (see, e.g., [9]) and for description logics for comparing TBoxes and knowledge bases [11, 4]. To the best of our knowledge, it has never been explicitly considered for OBDA specifications.

The majority of work on on OBDA has considered *conjunctive queries (CQs)* as the query language. Therefore, a first natural choice would be to compare OBDA specifications with respect to the whole class of CQs. We thus define and study a notion of *CQ-entailment* between OBDA specifications that formalizes this case. We also consider the important subclass of *instance queries (IQs)*, i.e., queries that ask for the instances of a single concept or role, and analyze the notion of *IQ-entailment* between specifications. Moreover, in many application contexts only a (small) set of predefined conjunctive queries are of interest for the OBDA user(s): in such cases, it may be more appropriate to tailor the comparison of specifications to a specific set of queries. For this reason, we also study in this paper the notions of *single CQ-entailment* and *single IQ-entailment*, which compare specifications with respect to a single CQ or IQ, respectively.

We present a first investigation of the computational complexity of deciding the above forms of entailment for a pair of OBDA specifications. We study ontologies specified in *DL-Lite*$_R$ and three different mapping languages (linear, GAV and GLAV). In all cases, we provide exact complexity bounds for the entailment problem. Our results are summarized in Figure 1. As shown in the table, the complexity of the entailment check ranges from NL (non-deterministic logarithmic space) for linear mappings and IQ-entailment to EXPTIME for CQ-entailment. To obtain these results, we show that instead of considering all possible data instances, it is sufficient to consider a small number of databases of a particular form. We also exploit connections to query containment in the presence of signature restrictions [3] and KB query inseparability [4].

2 Preliminaries

We start from four pairwise disjoint countably infinite set of names: the set of concept names $\mathsf{N_C}$, the set of role names $\mathsf{N_R}$, the set of relation names $\mathsf{N_{rel}}$, the set of constant names $\mathsf{N_I}$ (also called individuals).

To introduce OBDA specifications, we first recall the notion of knowledge base (KB) in Description Logics (DLs). A DL KB is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where: $\mathcal{T}$, called the *TBox*, is the intensional component of the KB, and is constituted by a finite set of axioms expressing intensional knowledge; and $\mathcal{A}$, called the *ABox*, is a finite set of atomic concept and role assertions (set of ground facts). We assume that the concept,

role and constant names occurring in every TBox and ABox belong to $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$, respectively. We denote by $\mathsf{sig}(\mathcal{T})$ and $\mathsf{sig}(\mathcal{A})$ the set of concept and role names occurring in $\mathcal{T}$ and $\mathcal{A}$, respectively.

Although the definitions of Section 3 are general, in Section 4 we will focus on the DL *DL-Lite*$_R$ [6]. A *DL-Lite*$_R$ TBox consists of a finite set of concept inclusions $B \sqsubseteq C$ and role inclusions $R \sqsubseteq S$, where B, C, R, and S are defined according to the following syntax (where A is a concept name and P is a role name):

$$B \to A \mid \exists R \qquad C \to B \mid \neg B \qquad R \to P \mid P^- \qquad S \to R \mid \neg R$$

We now introduce OBDA specifications. As already explained, a mapping assertion specifies the semantic relationship between elements of a DL ontology, specified through a TBox, to elements of a database. Such a relationship is specified through a pair of queries, one over the TBox signature, and the other one over the database signature. In this paper, we focus on the case where both queries involved in the mapping assertion are conjunctive queries: such mapping assertions are called GLAV (for 'global-as-view') mappings in the literature [8].

Mappings are formally defined as follows. An *atom* is an expression $r(\boldsymbol{t})$ where r is a predicate and $\boldsymbol{t}$ is a tuple of variables and constants. Then, a *(GLAV) mapping assertion* m is an expression of the form $q_s(\boldsymbol{x}) \to q_o(\boldsymbol{x})$, where $q_s(\boldsymbol{x})$ (called the *body* of m, $body(m)$) is a conjunction of atoms over predicates from $\mathsf{N_{rel}}$ and constants from $\mathsf{N_I}$, $q_o(\boldsymbol{x})$ (called the *head* of m, $head(m)$) is a conjunction of atoms using predicates from $\mathsf{N_C} \cup \mathsf{N_R}$ and constants from $\mathsf{N_I}$, and $\boldsymbol{x}$, called the *frontier variables* of m, are the variables that appear both in q_o and in q_s. The *arity* of m is the number of its frontier variables. When $q_o(\boldsymbol{x})$ has the form $p(\boldsymbol{x})$ (i.e., $q_o(\boldsymbol{x})$ is a single atom whose arguments are $\boldsymbol{x}$), we call m a *GAV* mapping assertion. A *linear* mapping assertion is a GAV assertion whose body consists of a single atom. A *(GLAV) mapping* $\mathcal{M}$ is a set of mapping assertions. A *GAV mapping* is a mapping constituted of GAV mapping assertions. A *linear mapping* is a set of linear mapping assertions. Without loss of generality, we assume that in every mapping $\mathcal{M}$, every pair of distinct mapping assertions uses pairwise disjoint sets of variables.

An *OBDA specification* is a pair $\Gamma = \langle \mathcal{T}, \mathcal{M} \rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{M}$ is a mapping. Given a mapping assertion m of arity n and an n-tuple of constants $\boldsymbol{a}$, we denote by $m(\boldsymbol{a})$ the assertion obtained from m by replacing the frontier variables with the constants in $\boldsymbol{a}$.

Given a set of atoms AT, $gr(AT)$ is the function that returns the set of ground atoms obtained from AT by replacing every variable symbol x with a fresh constant symbol c_x. We assume that such constant symbols do not occur elsewhere in the application context of the function gr (i.e., in the TBoxes, mappings and databases involved).

In this paper, a *database* (instance) is a set of ground atoms using relation names from $\mathsf{N_{rel}}$ and constant names from $\mathsf{N_I}$. Given a mapping $\mathcal{M}$ and a database instance D, we define the *ABox for D and $\mathcal{M}$*, denoted as $\mathcal{A}_{\mathcal{M},D}$, as the following ABox:

$$\{\ \beta \in gr(head(m(\boldsymbol{a}))) \mid m \in \mathcal{M} \text{ and and } D \models \exists \boldsymbol{y}.body(m(\boldsymbol{a}))\ \}$$

where we assume that $\boldsymbol{y}$ are the variables occurring in $body(m(\boldsymbol{a}))$. Given an OBDA specification $\Gamma = \langle \mathcal{T}, \mathcal{M} \rangle$ and a database instance D, we define the *models of Γ and*

D, denoted as $Mods(\Gamma, D)$ as the set of models of the KB $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle$. When such a set is empty, we write $\langle \mathcal{T}, \mathcal{M}, D \rangle \models \bot$ (analogously, when a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ has no models, we write $\langle \mathcal{T}, \mathcal{A} \rangle \models \bot$).

We are interested in the problem of answering instance queries and conjunctive queries over a pair composed of an OBDA specification and a database. A *Boolean conjunctive query (CQ)* is an expression of the form $\exists \mathbf{x}(\alpha_1 \wedge \ldots \wedge \alpha_n)$ where every α_i is an atom whose arguments are either constants or variables from $\mathbf{x}$. For a non-Boolean CQ q with answer variables $v_1, \ldots, v_k$, a tuple of constants $\boldsymbol{a} = \langle a_1, \ldots, a_k \rangle$ occurring in $\mathcal{A}$ is said to be a *certain answer* for q w.r.t. $\mathcal{K}$ just in the case that $\mathcal{K} \models q(\boldsymbol{a})$, where $q(\boldsymbol{a})$ is the Boolean query obtained from q by replacing each v_i by a_i. We call *instance query (IQ)* a CQ consisting of a single atom of the form $A(x)$ or $R(x, y)$, with A concept name, R role name, and x, y distinct free variables. We denote by $\mathsf{sig}(q)$ the set of concept and role names occurring in a query q. We use CQ (resp. IQ) to refer the set of all CQs (resp. IQs) over the DL signature $\mathsf{N_C} \cup \mathsf{N_R}$.

Given an OBDA specification $\Gamma = \langle \mathcal{T}, \mathcal{M} \rangle$, a database instance D, and a conjunctive query q, we define the certain answers for q w.r.t. (Γ, D) as the tuples of constants from D that are certain answers for q w.r.t. $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle$. In particular, for Boolean CQs, we say that q is entailed by (Γ, D), denoted by $(\Gamma, D) \models q$ (or $\langle \mathcal{T}, \mathcal{M}, D \rangle \models q$), if $\mathcal{I} \models q$ for every $\mathcal{I} \in Mods(\Gamma, D)$. Note that for non-Boolean queries, we only consider tuples of constants from D, in order to avoid including those fresh constants introducing in $\mathcal{A}_{\mathcal{M},D}$ by grounding existential variables in mapping heads.

3 Query-based Entailment for OBDA Specifications

We start by recalling the classical notion of entailment between OBDA specifications.

Definition 1 (Logical entailment). *An OBDA specification $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle$* logically entails *$\langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, written $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathsf{log}} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ if and only the first-order theory $\mathcal{T}_1 \cup \mathcal{M}_1$ logically entails the first-order theory $\mathcal{T}_2 \cup \mathcal{M}_2$.*

We now define the formal notions of query-based entailment between OBDA specifications considered in this paper. First, we introduce a notion of entailment that compares specifications based upon the constraints they impose regarding consistency.

Definition 2 ($\bot$-entailment). *Let q be a query. An OBDA specification $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle$* $\bot$-entails *$\langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, written $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\bot} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, iff, for every database D,*

$$\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models \bot \quad \Rightarrow \quad \langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$$

Next, we define a notion of query entailment between OBDA specifications with respect to a *single* query.

Definition 3 (Single query entailment). *Let q be a query. An OBDA specification $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle$* q-entails *$\langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, written $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_q \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, if and only if $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\bot} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ and for every database D,*

$$\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models q(\boldsymbol{a}) \quad \Rightarrow \quad \langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q(\boldsymbol{a})$$

When q is an IQ, we call the entailment relation in the preceding definition *single IQ-entailment*, while we call it *single CQ-entailment* if q is an arbitrary CQ.

We can generalize the previous definition to classes of queries as follows.

Definition 4 (Query entailment). *Let $\mathcal{L}$ be a (possibly infinite) set of queries. An OBDA specification $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ $\mathcal{L}$-*entails *$\langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, written $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathcal{L}} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ iff $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\perp} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ and $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{q} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ for every query $q \in \mathcal{L}$.*

When $\mathcal{L} = \mathsf{IQ}$, we call the preceding entailment relation *IQ-entailment*, and for $\mathcal{L} = \mathsf{CQ}$, we use the term *CQ-entailment*.

Note that, for each of the above notions of entailment, a notion of equivalence between OBDA specifications can be immediately derived, corresponding to entailment in both directions (we omit the formal definitions due to space limitations).

The following property immediately follows from the above definitions.

Proposition 1. *Let $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle, \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ be two OBDA specifications, and let $\mathcal{L}_1$ be a set of queries. Then, $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathsf{log}} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ implies $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathcal{L}_1} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$. Moreover, if $\mathcal{L}_2 \subseteq \mathcal{L}_1$, then $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathcal{L}_1} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ implies $\langle \mathcal{T}_1, \mathcal{M}_1 \rangle \models_{\mathcal{L}_2} \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$.*

As a consequence of the above property, we have that logical entailment implies CQ-entailment, and CQ-entailment implies IQ-entailment. The converse implications do not hold, as the following examples demonstrate.

Example 1. We start by illustrating the difference between logical entailment and CQ-entailment. Consider a database containing instances for the relation *EXAM*(*studentName,courseName,grade,date*). Then, let $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$, where

$$\mathcal{T}_1 = \{\textit{Student} \sqsubseteq \textit{Person},\ \textit{PhDStudent} \sqsubseteq \textit{Student}\}$$
$$\mathcal{M}_1 = \{\textit{EXAM}(x, y, z, w) \rightarrow \textit{Student}(x)\}$$

and let $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, where $\mathcal{T}_2 = \{\textit{Student} \sqsubseteq \textit{Person}\}$ and $\mathcal{M}_2 = \mathcal{M}_1$. It is immediate to verify that $\Gamma_2 \not\models_{\mathsf{log}} \Gamma_1$. However, we have that $\Gamma_2 \models_{\mathsf{CQ}} \Gamma_1$. Indeed, $\Gamma_2 \models_{\mathsf{CQ}} \Gamma_1$ can be intuitively explained by the fact that the mapping $\mathcal{M}_1$ does not retrieve any instances of the concept *PhDStudent* (and there are no subclasses that can indirectly populate it), so the presence of the inclusion $\textit{PhDStudent} \sqsubseteq \textit{Student}$ in $\mathcal{T}_1$ does not have any effect on query answering; in particular, every CQ that mentions the concept *PhDStudent* cannot be entailed both under Γ_1 and under Γ_2. Notice also that, if we modify the mapping $\mathcal{M}_1$ to map *PhDStudent* instead of *Student* (i.e., if $\mathcal{M}_1$ were $\{\textit{EXAM}(x, y, z, w) \rightarrow \textit{PhDStudent}(x)\}$), then CQ-entailment between Γ_2 and Γ_1 would no longer hold.

Next, consider $\Gamma_3 = \langle \mathcal{T}_3, \mathcal{M}_3 \rangle$, where $\mathcal{T}_3 = \emptyset$ and

$$\mathcal{M}_3 = \{\textit{EXAM}(x, y, z, w) \rightarrow \textit{Student}(x),\ \ \textit{EXAM}(x, y, z, w) \rightarrow \textit{Person}(x)\}$$

Again, it it immediate to see that $\Gamma_3 \not\models_{\mathsf{log}} \Gamma_2$, while we have that $\Gamma_3 \models_{\mathsf{CQ}} \Gamma_2$. Indeed, $\Gamma_3 \models_{\mathsf{CQ}} \Gamma_2$ follows informally from the fact that the mapping $\mathcal{M}_3$ is able to "extensionally" simulate the inclusion $\textit{Student} \sqsubseteq \textit{Person}$ of $\mathcal{T}_2$, which is sufficient for Γ_3 to entail every CQ in the same way as Γ_2.

Example 2. We slightly modify the previous example to show the difference between CQ-entailment and IQ-entailment. Consider $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M} \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M} \rangle$ where

$$\mathcal{T}_1 = \{\textit{Student} \sqsubseteq \textit{Person},\ \textit{Student} \sqsubseteq \exists \textit{takesCourse}\}$$
$$\mathcal{T}_2 = \{\textit{Student} \sqsubseteq \textit{Person}\}$$
$$\mathcal{M} = \{\textit{EXAM}(x, y, z, w) \rightarrow \textit{Student}(x)\}$$

Type of entailment	Type of mapping	Complexity
logical	GAV / GLAV	NP-complete
	linear	NL-complete
$\bot$	GAV / GLAV	NP-complete
	linear	NL-complete
CQ	linear / GAV / GLAV	EXPTIME-complete
IQ	linear	NL-complete
	GAV / GLAV	NP-complete
single CQ	linear / GAV / GLAV	Π_2^p-complete
single IQ	linear	NL-complete
	GAV / GLAV	NP-complete

Fig. 1. Complexity results for entailment between OBDA specifications in *DL-Lite*$_R$

Then, it can be easily verified that $\Gamma_2 \not\models_{\mathsf{CQ}} \Gamma_1$. Indeed, consider the Boolean CQ $\exists x, y\, \mathit{takesCourse}(x, y)$: for every database D, this query is not entailed by the pair (Γ_2, D), while this is not the case when the specification is Γ_1. On the other hand, we have that $\Gamma_2 \models_{\mathsf{IQ}} \Gamma_1$: in particular, for every database D and for every pair of individuals a, b, neither (Γ_1, D) nor (Γ_2, D) entails the IQ $\mathit{takesCourse}(a, b)$. Finally, let q be the non-Boolean CQ $\exists x\, \mathit{takesCourse}(x, y)$: then, it can be easily verified that the single CQ-entailment $\Gamma_2 \models_q \Gamma_1$ holds; while for the CQ q' of the form $\exists y\, \mathit{takesCourse}(x, y)$, the single CQ-entailment $\Gamma_2 \models_{q'} \Gamma_1$ does not hold.

4 Complexity Results for *DL-Lite*$_R$

In this section, we investigate the computational properties of the different notions of entailment between OBDA specifications defined in the previous section. For this first study, we focus on the case in which the TBox is formulated in *DL-Lite*$_R$ [6], as it is the basis for the OWL 2 QL profile and one of the most commonly considered DLs for OBDA. The results of our complexity analysis are displayed in Figure 1.

In what follows, we formally state the different complexity results and provide some ideas about the proofs. We begin by considering the complexity of deciding classical entailment between OBDA specifications.

Theorem 1. *Classical logical entailment for OBDA specifications based upon DL-Lite*$_R$ *TBoxes is* NP*-complete for GAV or GLAV mappings, and NL-complete for linear mappings.*

Proof. Let $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$, $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$. First, it is easy to see that $\Gamma_1 \models_{\mathsf{log}} \Gamma_2$ iff (i) $\mathcal{T}_1 \models \mathcal{T}_2$; and (ii) $\Gamma_1 \models_{\mathsf{log}} \mathcal{M}_2$. Property (i) can be decided in NL [2]. Property (ii) can be decided by an algorithm that, for every assertion $m \in \mathcal{M}_2$, first builds a database D corresponding to $gr(body(m))$ (i.e., obtained by "freezing" the body of m), and then checks whether $\langle \Gamma_1, D \rangle$ entails the CQ corresponding to the head of m whose frontier variables have been replaced by the corresponding constants. This algorithm runs in NP in the case of GAV and GLAV mappings, and in NL in the case of linear mappings,

which implies the overall upper bounds in the theorem statement. The lower bound for GAV mappings can be obtained through an easy reduction of conjunctive query containment to logical entailment, while the one for linear mappings follows from a reduction of the entailment of a concept inclusion axiom in a *DL-Lite*$_R$ TBox. □

We next consider $\bot$-entailment. Our upper bounds rely on the following result that shows it is sufficient to consider a small number of small databases.

Theorem 2. *Let q be a CQ, and let $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ be OBDA specifications such that $\mathcal{T}_1, \mathcal{T}_2$ are formulated in DL-Lite$_R$, and $\mathcal{M}_1, \mathcal{M}_2$ are GLAV mappings. Then $\Gamma_1 \models_\bot \Gamma_2$ if and only if $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$ for every database D satisfying the following condition:*[1]

- *Condition 1: D is obtained by (i) taking two mapping assertions m_1, m_2 from $\mathcal{M}_2$, (ii) selecting atoms α_1 and α_2 from $head(m_1)$ and $head(m_2)$ respectively, (iii) identifying in m_1 and m_2 some variables from α_1 and α_2 in such a way that $\langle \mathcal{T}, gr(\{\alpha_1, \alpha_2\}) \rangle \models \bot$, (iv) setting D equal to $gr(body(m_1) \cup body(m_2))$.*

Proof. The one direction is immediate from the definitions. For the interesting direction, let us suppose that $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$ for every database D satisfying Condition 1. Let us further suppose that we have $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \models \bot$, where D_0 may be any database. We thus have $\langle \mathcal{T}_2, \mathcal{A}_{\mathcal{M}_2, D_0} \rangle \models \bot$. It is well known that every minimal inconsistent subset of a *DL-Lite*$_R$ KB contains at most two ABox assertions, so there must exist a subset $\mathcal{A}' \subseteq \mathcal{A}_{\mathcal{M}_2, D}$ with $|\mathcal{A}'| \leq 2$ such that $\langle \mathcal{T}_2, \mathcal{A}' \rangle \models \bot$. Let γ be the conjunction of atoms obtained by taking for each ABox assertion in $\mathcal{A}'$, a mapping assertion that produced it, identifying those variables (and only those variables) needed to produce the ABox assertion(s), and then taking the conjunction of the atoms in the bodies. We observe that by construction $D_\gamma = gr(\gamma)$ satisfies Condition 1 and is such that $\langle \mathcal{T}_1, \mathcal{M}_1, D_\gamma \rangle \models \bot$. By construction, there is a homomorphism of γ into the original database D_0. It follows that $\langle \mathcal{T}_1, \mathcal{M}_1, D_0 \rangle \models \bot$. □

Using the preceding result, we can pinpoint the complexity of $\bot$-entailment.

Theorem 3. *The $\bot$-entailment problem is NP-complete for OBDA specifications based upon DL-Lite$_R$ TBoxes and GAV / GLAV mappings, and NL-complete in the case of linear mappings.*

Proof. We know from Theorem 2 that $\Gamma_1 \models_\bot \Gamma_2$ iff $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$ for every database D satisfying Condition 1. For the GAV / GLAV case, we guess one such database D and a polynomial-size proof that $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$. For the linear case, we note that the databases satisfying Condition 1 contain at most 2 tuples each and can be enumerated in logarithmic space. For every such database, we can check using an NL oracle whether $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \bot$. Since $\mathrm{L}^{\mathrm{NL}} = \mathrm{NL}$, we obtain an NL procedure. □

Next we consider entailment with respect to a specific query. We again start by showing it is sufficient to consider a finite number of databases of a particular form.

[1] Recall that distinct mapping assertions in a mapping have no common variables.

Theorem 4. *Let q be a CQ, and let $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ be OBDA specifications such that $\mathcal{T}_1, \mathcal{T}_2$ are formulated in DL-Lite$_R$, and $\mathcal{M}_1, \mathcal{M}_2$ are GLAV mappings. Then $\Gamma_1 \models_q \Gamma_2$ if and only if $\Gamma_1 \models_\perp \Gamma_2$ and $\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models q(\boldsymbol{a})$ implies $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q(\boldsymbol{a})$ for every database D satisfying the following condition:*

- *Condition 2: D is obtained by (i) taking $k \leq |q|$ mapping assertions $m_1, m_2, \ldots, m_k$ from $\mathcal{M}_2$, (ii) identifying some of the frontier variables in $m_1, m_2, \ldots, m_k$, (iii) letting $D = gr(body(m_1) \cup body(m_2) \cup \ldots \cup body(m_k))$.*

If q is an IQ, then the latter condition can be replaced by:

- *Condition 3: D is obtained by (i) taking a mapping assertion m from $\mathcal{M}_2$ and choosing an atom $\alpha \in head(m)$, (ii) possibly identifying in m the (at most two) frontier variables appearing in α, and (iii) letting $D = gr(body(m))$.*

Proof. Again the one direction is immediate. To show the non-trivial direction, let us suppose that $\Gamma_1 \models_\perp \Gamma_2$ and that $\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models q(\boldsymbol{c})$ implies $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q(\boldsymbol{c})$ for every tuple $\boldsymbol{c}$ and database D satisfying Condition 2 (we return later to the case of IQs). Let us further suppose that we have $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \models q(\boldsymbol{a})$. The first possibility is that $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \models \perp$, in which case we have $\langle \mathcal{T}_1, \mathcal{M}_1, D_0 \rangle \models \perp$ because of $\Gamma_1 \models_\perp \Gamma_2$. We thus obtain $\langle \mathcal{T}_1, \mathcal{M}_1, D_0 \rangle \models q_0(\boldsymbol{a})$. The other possibility is that $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \models q_0(\boldsymbol{a})$ and $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \not\models \perp$. If $\langle \mathcal{T}_1, \mathcal{M}_1, D_0 \rangle \models \perp$, we immediately obtain $\langle \mathcal{T}_1, \mathcal{M}_1, D_0 \rangle \models q_0(\boldsymbol{a})$. Otherwise, let $\mathcal{A}_{\mathcal{M}_2, D_0}$ be the ABox for $\mathcal{M}_2$ and D_0. Since $\langle \mathcal{T}_2, \mathcal{M}_2, D_0 \rangle \models q_0(\boldsymbol{a})$, we have $\langle \mathcal{T}_2, \mathcal{A}_{\mathcal{M}_2, D_0} \rangle \models q_0(\boldsymbol{a})$. It is a well-known property of *DL-Lite*$_R$ that there exists a subset $\mathcal{A}' \subseteq \mathcal{A}_{\mathcal{M}_2, D_0}$ with $|\mathcal{A}'| \leq |q_0|$ such that $\langle \mathcal{T}_2, \mathcal{A}' \rangle \models q_0(\boldsymbol{a})$. Let $|\mathcal{A}'| = k$, and let $\beta_1, \ldots, \beta_k$ be the ABox assertions in $\mathcal{A}'$. For each β_i, we choose a mapping assertion $m_i \in \mathcal{M}_2$ and a homomorphism h_i of $body(m_i)$ into D_0 such that $gr(h_i(head(m)))$ contains β_i. We also select an atom $\alpha_i \in head(m)$ such that $gr(h(\alpha)) = \beta$. Let m_i' be obtained from m_i by identifying frontier variables y and z if $h_i(y) = h_i(z)$, and set $D' = gr(body(m_1') \cup \ldots \cup body(m_k'))$. It is easy to see that D' satisfies Condition 2. Moreover, by construction, the ABox $\mathcal{A}_{\mathcal{M}_2, D'}$ contains a subset $\mathcal{A}''$ that is isomorphic to $\mathcal{A}'$, and so $\langle \mathcal{T}_2, \mathcal{M}_2, D' \rangle \models q_0(\boldsymbol{a}')$ where $\boldsymbol{a}'$ is tuple corresponding to $\boldsymbol{a}$ according to this isomorphism. Applying our assumption, we obtain $\langle \mathcal{T}_1, \mathcal{M}_1, D' \rangle \models q_0(\boldsymbol{a}')$. Using the fact that there is a homomorphism of $body(m_1') \cup \ldots \cup body(m_k')$ into D_0 that is an isomorphism on the frontier variables, we obtain $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q_0(\boldsymbol{a})$.

Finally, for the case of instance queries, we simply note that we have $k = 1$, and it is only necessary to identify those variables in the head atom of the mapping that leads to introducing the single ABox assertion of interest. This yields Condition 3. □

We pinpoint the complexity of single CQ-entailment, showing it to be Π_2^p-complete.

Theorem 5. *The single CQ-entailment problem is Π_2^p-complete for OBDA specifications based upon DL-Lite$_R$ TBoxes and GLAV mappings. The lower bound holds even for linear mapping assertions and when both TBoxes are empty.*

Proof. For the upper bound, consider two OBDA specifications $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$. From Theorems 2 and 4, we know that $\Gamma_1 \not\models_q \Gamma_2$ if and only if one of the following holds:

– there is a database D satisfying Condition 2 such that $\langle \mathcal{T}_1, \mathcal{M}_1, D\rangle \not\models \bot$;
– there is a database D satisfying Condition 2 such that $\langle \mathcal{T}_2, \mathcal{M}_2, D\rangle \models q(\boldsymbol{a})$, $\langle \mathcal{T}_2, \mathcal{M}_2, D\rangle \not\models \bot$, and $\langle \mathcal{T}_1, \mathcal{M}_1, D\rangle \not\models q(\boldsymbol{a})$.

The first item can be checked using an NP oracle (by Theorem 3). To check the second item, we remark that the size of databases satisfying Condition 2 cannot exceed $max(2, |q|) \cdot \mathsf{maxbody}$, where $\mathsf{maxbody}$ is the maximum number of atoms appearing in the body of a mapping assertion in $\mathcal{M}_2$. It follows that to show that the second item above is violated, we can guess a database D of size at most $max(2, |q|) \cdot \mathsf{maxbody}$ together with a tuple of constants $\boldsymbol{a}$ and a polynomial-size proof that $\langle \mathcal{T}_2, \mathcal{M}_2, D\rangle \models q(\boldsymbol{a})$, and then we can verify using an NP oracle that $\langle \mathcal{T}_1, \mathcal{M}_1, D\rangle \not\models q(\boldsymbol{a})$. We therefore obtain a Σ_2^p procedure for deciding the complement of our problem.

For the lower bound, we utilize a result from [3] on query containment over signature-restricted ABoxes. In that paper, it is shown how, given a 2QBF $\forall \boldsymbol{u} \exists \boldsymbol{v} \varphi(\boldsymbol{u}, \boldsymbol{v})$, one can construct a TBox $\mathcal{T}$, Boolean CQs q_1 and q_2, and a signature Σ such that $\forall \boldsymbol{u} \exists \boldsymbol{v} \varphi(\boldsymbol{u}, \boldsymbol{v})$ is valid iff $\mathcal{T}, \mathcal{A} \models q_1 \Rightarrow \mathcal{T}, \mathcal{A} \models q_2$ for all ABoxes $\mathcal{A}$ with $\mathsf{sig}(\mathcal{A}) \subseteq \Sigma$. We will not detail the construction but simply remark that the same TBox $\mathcal{T} = \{T \sqsubseteq V, F \sqsubseteq V\}$ is used for all QBFs, the signature Σ is given by $(\mathsf{sig}(\mathcal{T}) \cup \mathsf{sig}(q_1) \cup \mathsf{sig}(q_2)) \setminus \{V\}$, and the query q_2 is such that $V \notin \mathsf{sig}(q_2)$.

In what follows, we will show how given $\mathcal{T}$, q_1, q_2, and Σ as above, we can reduce the problem of testing whether $\mathcal{T}, \mathcal{A} \models q_1$ implies $\mathcal{T}, \mathcal{A} \models q_2$ for all Σ-ABoxes to the problem of single CQ entailment. We will use Σ for our database instances, and we create two copies $\Sigma_1 = \{P^1 \mid P \in \Sigma\}$ and $\Sigma_2 = \{P^2 \mid P \in \Sigma\}$ of the signature Σ to be used in the head of mapping assertions. Next, we define sets of mapping assertions $\mathsf{copy}^1(\Sigma)$ and $\mathsf{copy}^2(\Sigma)$ that simply copies all of the predicates in Σ into the corresponding symbol in Σ_1 (resp. Σ_2). Formally, for $j \in \{1, 2\}$,

$$\mathsf{copy}^j(\Sigma) = \{A(x) \to A^j(x) \mid A \in \Sigma \cap \mathsf{N_C}\} \cup \{R(x, y) \to R^j(x, y) \mid R \in \Sigma \cap \mathsf{N_R}\}$$

We further define, given a data signature Λ_1 and DL signature Λ_2, a set $\mathsf{populate}(\Lambda_1, \Lambda_2)$ of mapping assertions that populates the relations in Λ_2 using all possible combinations of the constants appearing in tuples over Λ_1:

$$\begin{aligned}\mathsf{populate}(\Lambda_1, \Lambda_2) = \{&P_1(x_1, \ldots, x_k) \to P_2(x'_1, \ldots, x'_\ell) \mid P_1 \in \Lambda_1, \mathsf{arity}(P) = k,\\ &P_2 \in \Lambda_2, \mathsf{arity}(P) = \ell, \{x'_1, \ldots, x'_\ell\} \subseteq \{x_1, \ldots, x_k)\}\end{aligned}$$

Using $\mathsf{copy}^1(\Sigma)$, $\mathsf{copy}^2(\Sigma)$, $\mathsf{populate}(\Sigma, \Sigma^1)$, and $\mathsf{populate}(\Sigma, \Sigma^2)$, we construct the following mappings:

$$\begin{aligned}\mathcal{M}_1 &= \mathsf{populate}(\Sigma, \Sigma^1) \cup \mathsf{copy}^2(\Sigma)\\ \mathcal{M}_2 &= \mathsf{copy}^1(\Sigma) \cup \mathsf{populate}(\Sigma, \Sigma^2) \cup \{T(x) \to V(x), F(x) \to V(x)\}\end{aligned}$$

Observe that both mappings are linear. For the query, we let q'_1 (resp. q'_2) be obtained from q_1 (resp. q_2) by replacing every predicate P by P^1(resp. P^2). We also rename variables so that q'_1 and q'_2 do not share any variables. We then let q be the CQ obtained by taking the conjunction of q'_1 and q'_2 and existentially quantifying all variables. In the appendix, we show that $\langle \emptyset, \mathcal{M}_1\rangle \models_q \langle \emptyset, \mathcal{M}_2\rangle$ iff $\mathcal{T}, \mathcal{A} \models q_1 \Rightarrow \mathcal{T}, \mathcal{A} \models q_2$ for all Σ-ABoxes. By combining this with the reduction from [3], we obtain a reduction from universal 2QBF to the q-entailment problem, establishing Π_2^p-hardness of the latter. □

If we consider IQs instead, the complexity drops to either NP- or NL-complete.

Theorem 6. *The single IQ-entailment problem is NP-complete for OBDA specifications based upon DL-Lite$_R$ TBoxes and either GAV or GLAV mappings. It is NL-complete if linear mappings are considered.*

Proof. We give the arguments for GAV and GLAV mappings (for linear case, see the appendix). For the NP upper bound, consider two OBDA specifications $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$, and let q be an IQ. By Theorem 4, $\Gamma_1 \models_q \Gamma_2$ if and only if $\Gamma_1 \models_\bot \Gamma_2$ and $\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models \alpha$ implies $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \alpha$ for all databases D satisfying Condition 3 and for all Boolean IQs α obtained by instantiating the variable(s) in q with constant(s) from D.

We already know that it is in NP to test whether $\Gamma_1 \models_\bot \Gamma_2$. For the second property, observe that there are only polynomially many databases satisfying Condition 2, since each corresponds to choosing a mapping assertion m in $\mathcal{M}_2$, an atom $\alpha \in head(m)$, and deciding whether or not to identify variables in α. For every such database D, we compute (in polynomial time) the set of Boolean IQs β obtained by instantiating the IQ q with constants from D for which $\langle \mathcal{T}_2, \mathcal{M}_2, gr(head(m)) \rangle \models \beta$. For every such β, we guess a polynomial-size proof that $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models \beta$. If all of our polynomially many guesses succeed, then the procedure returns yes, and otherwise no. By grouping all of the guesses together, we obtain an NP decision procedure.

The NP lower bound is by reduction from the NP-complete CQ containment problem: given two CQs q_1, q_2 both having a single answer variable x, we have $q_1 \subseteq q_2$ iff $\langle \emptyset, \{q_2 \rightarrow A(x)\} \rangle \models_{A(x)} \langle \emptyset, \{q_1 \rightarrow A(x)\} \rangle$, where A is a concept name that does not appear in either of q_1 and q_2. □

Finally, we consider entailment with respect to entire classes of queries. Again, we can show it is sufficient to consider a small number of databases of a particular form.

Theorem 7. *Let $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$ be as in Theorem 4. For $\mathcal{L} \in \{\mathsf{CQ}, \mathsf{IQ}\}$, $\Gamma_1 \models_\mathcal{L} \Gamma_2$ if and only if $\Gamma_1 \models_\bot \Gamma_2$ and $\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models q(\boldsymbol{a})$ implies $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q(\boldsymbol{a})$ for every $q \in \mathcal{L}$ and every database D that satisfies Condition 3.*

We show that testing CQ-entailment is much more difficult than for single CQs. Both the upper and lower bounds use recent results on KB query inseparability [4].

Theorem 8. *CQ-entailment is EXPTIME-complete for OBDA specifications based upon DL-Lite$_R$ TBoxes and either GLAV, GAV, or linear mappings.*

Proof. We start with the proof of membership in EXPTIME. Consider OBDA specifications $\Gamma_1 = \langle \mathcal{T}_1, \mathcal{M}_1 \rangle$ and $\Gamma_2 = \langle \mathcal{T}_2, \mathcal{M}_2 \rangle$. By Theorem 7, $\Gamma_1 \models_\mathsf{CQ} \Gamma_2$ if and only if $\Gamma_1 \models_\bot \Gamma_2$ and $\langle \mathcal{T}_2, \mathcal{M}_2, D \rangle \models q(\boldsymbol{a})$ implies $\langle \mathcal{T}_1, \mathcal{M}_1, D \rangle \models q(\boldsymbol{a})$ for every choice of $q(\boldsymbol{a})$ and every database D satisfying Condition 3. We know that testing $\Gamma_1 \models_\bot \Gamma_2$ can be done in NP (Theorem 3). To decide whether the second property holds, we consider each of the (polynomially many) databases satisfying Condition 3. For every such database D, we generate the two ABoxes $\mathcal{A}_{\mathcal{M}_1,D}$ and $\mathcal{A}_{\mathcal{M}_2,D}$ and the corresponding KBs $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_{\mathcal{M}_1,D} \rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_{\mathcal{M}_2,D} \rangle$. We then test whether it is the case that for every CQ q over $\mathsf{sig}(\mathcal{K}_2)$, $\mathcal{K}_2 \models q(\boldsymbol{a})$ implies $\mathcal{K}_1 \models q(\boldsymbol{a})$, and we

return no if this is not the case. The preceding check corresponds to the Σ-query entailment problem for *DL-Lite*$_R$ KBs, which has been recently studied in [4] and shown to be EXPTIME-complete. We therefore obtain an EXPTIME procedure for deciding CQ-entailment between OBDA specifications.

Our lower bound also makes use of the recent work on query inseparability of *DL-Lite*$_R$ knowledge bases. In [4], the following problem is shown to be EXPTIME-complete: given *DL-Lite*$_R$ TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ that are consistent with the ABox $\{A(c)\}$, decide whether the certain answers for q w.r.t. $\langle \mathcal{T}_2, \{A(c)\}\rangle$ are contained in those for $\langle \mathcal{T}_1, \{A(c)\}\rangle$ for every CQ q with $\mathsf{sig}(q) \subseteq \mathsf{sig}(\mathcal{T}_2)$. To reduce this problem to the CQ-entailment problem for OBDA specifications, we consider the following linear mapping that populates a fresh concept A' with all constants of a Σ-instance (refer to the proof of Theorem 5 for the definition of $\mathsf{populate}$): $\mathcal{M}_1 = \mathcal{M}_2 = \mathsf{populate}(\Sigma, \{A'\})$. To complete the proof, we show in the appendix that $\langle \mathcal{T}_1, \mathcal{M}_1\rangle \models_{\mathsf{CQ}} \langle \mathcal{T}_2, \mathcal{M}_2\rangle$ iff $\langle \mathcal{T}_2, \{A(c)\}\rangle \models q(\boldsymbol{a})$ implies $\langle \mathcal{T}_1, \{A(c)\}\rangle \models q(\boldsymbol{a})$ for every CQ q, where $\mathcal{T}_1'$ and $\mathcal{T}_2'$ are obtained from $\mathcal{T}_1$ and $\mathcal{T}_2$ by replacing A with A'. □

Our final result shows that IQ-entailment has the same complexity as single IQ-entailment. The proof proceeds similarly to the proof of Theorem 6.

Theorem 9. *IQ-entailment is NP-complete for OBDA specifications based upon DL-Lite*$_R$ *TBoxes and either GAV or GLAV mappings. It is NL-complete if linear mappings are considered.*

5 Conclusion and Future Work

In this paper, we have introduced notions of query-based entailment of OBDA specifications and have analyzed the complexity of checking query-based entailment for different classes of queries and mappings and for TBoxes formulated in *DL-Lite*$_R$.

The present work constitutes only a first step towards a full analysis of query-based forms of comparing OBDA specifications, and can be extended in several directions:

- First, it would be interesting to extend the computational analysis of query entailment to other DLs beyond *DL-Lite*$_R$. For instance, one interesting question for DLs with functional or cardinality restrictions concerns the impact of the Unique Name Assumption on the complexity of (and the techniques for) query entailment.
- Second, other forms of mapping beyond GAV and GLAV could be analyzed. In particular, we would like to see whether decidability of query entailment is preserved if we add some restricted form of inequality or negation to the mapping bodies.
- Third, we could introduce a query signature and only test entailment for queries formulated in the given signature, as has been done for TBox and KB query inseparability [4]. In fact, all of the complexity upper bounds in this paper hold also if we introduce a query signature, but this may not be the case for other DLs.
- Finally, to explore the impact of restricting the set of possible databases, we could extend the computational analysis to database schemas with integrity constraints.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338) and by the French National Research Agency under ANR project PAGODA (grant n. ANR-12-JS02-007-01).

References

1. N. Antonioli, F. Castanò, C. Civili, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, D. F. Savo, and E. Virardi. Ontology-based data access: the experience at the Italian Department of Treasury. In *Proc. of the Industrial Track of the 25th Int. Conf. on Advanced Information Systems Engineering (CAiSE)*, 2013.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, 2012.
4. E. Botoeva, R. Kontchakov, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Query inseparability for description logic knowledge bases. In *Proc. of the 14th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, 2014.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The Mastro system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
7. D. Calvanese, M. Giese, P. Haase, I. Horrocks, T. Hubauer, Y. Ioannidis, E. Jiménez-Ruiz, E. Kharlamov, H. Kllapi, J. Klüwer, M. Koubarakis, S. Lamparter, R. Möller, C. Neuenstadt, T. Nordtveit, Ö. Özcep, M. Rodriguez-Muro, M. Roshchin, F. Savo, M. Schmidt, A. Soylu, A. Waaler, and D. Zheleznyakov. Optique: OBDA solution for big data. In *Revised Selected Papers of ESWC 2013 Satellite Events*, volume 7955 of *Lecture Notes in Computer Science*, pages 293–295, 2013.
8. A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
9. G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *Very Large Database J.*, 20(2):277–302, 2011.
10. E. Kharlamov, M. Giese, E. Jiménez-Ruiz, M. G. Skjæveland, A. Soylu, D. Zheleznyakov, T. Bagosi, M. Console, P. Haase, I. Horrocks, S. Marciuska, C. Pinkel, M. Rodriguez-Muro, M. Ruzzi, V. Santarelli, D. F. Savo, K. Sengupta, M. Schmidt, E. Thorstensen, J. Trame, and A. Waaler. Optique 1.0: Semantic access to big data: The case of Norwegian Petroleum Directorate's FactPages. In *Proc. of the ISWC Posters & Demos Track*, pages 65–68, 2013.
11. B. Konev, R. Kontchakov, M. Ludwig, T. Schneider, F. Wolter, and M. Zakharyaschev. Conjunctive query inseparability of OWL 2 QL TBoxes. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI)*, 2011.
12. D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen. Towards mapping analysis in ontology-based data access. In *Proc. of the 8th Int. Conf. on Web Reasoning and Rule Systems (RR)*, pages 108–123, 2014.
13. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
14. M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC)*, 2013.

Schema-Agnostic Query Rewriting for OWL QL*

Stefan Bischof[1], Markus Krötzsch[2], Axel Polleres[3], and Sebastian Rudolph[2]

[1] Vienna University of Technology, Austria and Siemens AG Österreich, Austria
[2] Technische Universität Dresden, Germany
[3] Vienna University of Economics and Business, Austria

Ontology-based query answering (OBQA) has long been an important topic in applied and foundational research, and in particular in the area of description logics (DLs). Query answering has been studied for every major DL, but the most prominent use of DLs in query answering is based on the DLs of the DL-Lite family. In particular, these have widely been used in ontology-based data access (OBDA), e.g., to integrate disparate data sources or to provide views over legacy databases [3,14,4,9]. DL-Lite$_R$ is also the basis of the W3C OWL 2 Web Ontology Language profile OWL QL, which was specifically designed for OBDA applications [12].

On the other hand, research on query languages has led to a range of expressive features beyond basic pattern matching, e.g., by supporting navigational constructs or other forms of recursion. These developments have also affected practical query languages. SPARQL 1.1, the recent revision of the W3C SPARQL standard, introduces significant extensions to the capabilities of the popular RDF query language [7]. Even at the very core of the query language, we can find many notable new features, including *property paths*, *value creation* (BIND), inline data (VALUES), negation, and extended filtering capabilities. In addition, SPARQL 1.1 now supports query answering over OWL ontologies, taking full advantage of ontological information in the data [6,5]. Thus, with the arrival of SPARQL 1.1, every aspect of OBQA is supported by W3C technologies.

In practice, however, SPARQL and OWL QL are rarely integrated. Most works on OBDA address the problem of answering *conjunctive queries* (CQs), which correspond to SELECT-PROJECT-JOIN queries in SQL, and (to some degree) to Basic Graph Patterns in SPARQL. The most common approach for OBDA is *query rewriting*, where a given CQ is rewritten into a (set of) CQs that fully incorporate the schema information of the ontology. The answers to the rewritten queries (obtained without considering the ontology) are guaranteed to agree with the answers of the original queries (over the ontology). This approach separates the ontology (used for query rewriting) from the rest of the data (used for query answering), and it is typical that the latter is stored in a relational database. Correspondingly, the rewritten queries are often transformed into SQL for query answering. SPARQL and RDF do not play a role in this.

In a recent paper [1], we took a fresh look on the problem of OBQA query rewriting with SPARQL 1.1 as our target query language. The additional expressive power of SPARQL 1.1 allows us to introduce a new paradigm of *schema-agnostic query rewriting*, where the ontological schema is not needed for rewriting queries. Rather, the ontology is stored *together with the data* in a single RDF database. This is how many ontologies are managed today, and it corresponds to the W3C view on OWL and RDF, which does

* This extended abstract summarises the recent results of the authors' paper *Schema-Agnostic Query Rewriting in SPARQL 1.1* [1].

not distinguish schema and data components.[4] The fact that today's OBQA approaches separate both parts testifies to their focus on relational databases. Our work, somewhat ironically, widens the scope of OWL QL to RDF-based applications, which have hitherto focused on OWL RL as their ontology language of choice.

Another practical advantage of schema-agnostic query rewriting is that it supports frequent updates of both data and schema. The rewriting system does not need any information on the content of the database under query, while the SPARQL processor that executes the query does not need any support for OWL. This is particularly interesting if a database can only be accessed through a restricted SPARQL query interface that does not support reasoning. For example, we have used our approach to detect an inconsistency of DBpedia under OWL semantics, using only the public Live DBpedia SPARQL endpoint at http://live.dbpedia.org/sparql (the problem has since been corrected).

The main contributions of our work are:

- We expressed standard reasoning tasks for OWL QL, including consistency checking, classification, and instance retrieval, in *single, fixed* SPARQL 1.1 queries that are independent of the ontology. It turned out that SPARQL 1.1 property paths are powerful enough for OWL QL reasoning.
- We showed how to rewrite arbitrary SPARQL Basic Graph Patterns (BGPs) into single SPARQL 1.1 queries of polynomial size. This task was simplified by the fact that SPARQL does not support "non-distinguished" variables as used in CQs.
- We presented a schema-agnostic rewriting of general CQs in SPARQL 1.1, again into single queries of polynomial size. This rewriting is more involved, and we used two additional features: inline data (VALUES) and (in)equality checks in filters.
- We showed the limits of schema-agnostic rewriting in SPARQL 1.1 by proving that many other OWL features cannot be supported in this way. This includes even the most basic features of OWL EL and OWL RL, and mild extensions of OWL QL. It also is not possible to rewrite regular path queries (and thus basic graph patterns of SPARQL 1.1) into SPARQL 1.1, even for RDFS knowledge bases with assumption of standard use. This would require a more expressive query language, such as *monadically defined queries* [15].

Worst-case reasoning complexity remains the same in all cases, yet our approach is certainly more practical in the case of standard reasoning and BGP rewriting. For general CQs, the rewritten queries are usually too complex for today's RDF databases to handle. Nevertheless, we think that our "SPARQL 1.1 implementation" of OWL QL query answering is a valuable contribution, since it reduces the problem of supporting OWL QL in an RDF database to the task of optimizing a single (type of) query. Since OWL QL subsumes RDFS, one can also apply our insights to implement query answering under RDFS ontologies, which again leads to much simpler queries.

The full details of our rewritings are beyond this abstract, as the length of the rewritten queries – polynomial or not – is too long to include full examples here. However, our basic approach to reasoning with SPARQL 1.1 can be motivated by some very simple

[4] Nevertheless, it is also true that some RDF stores treat terminological triples in special ways, e.g., by keeping them in a dedicated named graph.

observations. Consider a situation where our TBox is guaranteed to contain only axioms of the form $A \sqsubseteq B$ with A and B class names. Such axioms are encoded in RDF using triples of the form A rdfs:subClassOf B. Clearly, one can now query for all (inferred) instances of a class A using the graph pattern

$$\{?X\ (\mathsf{rdf{:}type}\ /\ \mathsf{rdfs{:}subClassOf}^{*})\ A\},$$

which looks for elements ?X that are connected to class A through property rdf:type followed by zero or more uses of property rdfs:subClassOf. The queries used in our work are significantly more involved, since they need to take into account a much larger vocabulary used in OWL, including equivalent classes, subproperties and equivalent properties, inverses, n-ary class intersections, and existential restrictions. Moreover, the queries need to take into account the special semantics of $\top$ and the universal property, as well as the potential inconsistency caused by $\bot$ and the empty property.

An interesting observation here is that syntactic sugar can make schema-agnostic query rewriting more difficult. Clearly, taking into account many possible syntactic encodings must lead to larger queries, which will usually affect execution times. However, the OWL feature owl:SymmetricProperty even makes query rewriting impossible altogether. This is surprising, since property symmetry can easily be expressed using inverses and subproperties, which are fully supported by our approach. Such effects can occur since SPARQL 1.1 is not a universal computing formalism (for its complexity class). Note that these problems vanish if one allows even the most basic kinds of normalisation, but this is not always practical (e.g., when querying the DBpedia SPARQL endpoint).

An interesting side effect of our work is that it provides a simple, worst-case optimal method for terminological reasoning in OWL QL (and thus DL-Lite$_R$). As we treat TBox axioms as data, we can actually formulate queries over terminological knowledge, or even answer conjunctive queries where some class or property names are replaced by variables, with the intended meaning that these "meta-variables" range over vocabulary symbols of the ontology.[5]

Future steps in this line of research include empirical evaluations, where the main challenge is to identify OWL QL benchmarks with non-trivial TBoxes. It should not be assumed that the good theoretical properties of the approach translate directly into good performance, and further optimisations and adjustments might be needed. Implementation techniques such as partial materialisation would lead to a form of *combined rewriting* (cf. [8] for a different approach to combined rewriting). Moreover, it is interesting to extend our work towards more expressive ontology and query languages. On the one hand, one can look towards more expressive DLs, such as $\mathcal{EL}$, which have also been considered in query rewriting [13]. A schema-agnostic approach in this case would resemble Datalog-based reasoning calculi for these logics [10], and indeed one could view Datalog as a query language here. On the other hand, one might consider ontology languages that extend the expressiveness of DL-Lite by using existential rules, e.g., *linear TGDs* [2]. Such cases might actually be somewhat simpler to handle, since one is free to choose a (possibly normalised) database representation of rules, given that there is not standard RDF encoding available.

[5] The term *higher-order query* has been used in this context [11], although true higher-order variables would rather represent arbitrary sets without any relationship to the vocabulary.

Acknowledgements This work has been funded by the Vienna Science and Technology Fund (WWTF, project ICT12-015), and by the DFG in project DIAMOND (Emmy Noether grant KR 4381/1-1).

References

1. Bischof, S., Krötzsch, M., Polleres, A., Rudolph, S.: Schema-agnostic query rewriting in SPARQL 1.1. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandečić, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) Proc. 13th Int. Semantic Web Conf. (ISWC'14). LNCS, vol. 8796, pp. 584–600. Springer (2014)
2. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Semantics 14, 57–83 (2012)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Automated Reasoning 39(3), 385–429 (2007)
4. Di Pinto, F., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Proceedings of the 16th International Conference on Extending Database Technology. pp. 561–572. ACM (2013)
5. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: Patel-Schneider, P.F., Pan, Y., Glimm, B., Hitzler, P., Mika, P., Pan, J., Horrocks, I. (eds.) Proc. 9th Int. Semantic Web Conf. (ISWC'10). LNCS, vol. 6496, pp. 241–256. Springer (2010)
6. Glimm, B., Ogbuji, C. (eds.): SPARQL 1.1 Entailment Regimes. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-entailment/
7. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (21 March 2013), available at http://www.w3.org/TR/sparql11-query/
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Walsh [16], pp. 2656–2661
9. Kontchakov, R., Rodriguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: A short course. In: Rudolph, S., Gottlob, G., Horrocks, I., van Harmelen, F. (eds.) Reasoning Web, LNCS, vol. 8067, pp. 194–229. Springer, Mannheim, Germany (2013)
10. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh [16], pp. 2668–2673
11. Lenzerini, M., Lepore, L., Poggi, A.: Practical query answering over Hi(DL-Lite$_R$) knowledge bases. In: Bienvenu, M., Ortiz, M., Rosati, R., Simkus, M. (eds.) Proc. 27th Int. Workshop on Description Logics (DL'14). CEUR Workshop Proceedings, vol. 1193, pp. 608–619. CEUR-WS.org (2014)
12. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
13. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic 8(2), 186–209 (2010)
14. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) Proc. 12th Int. Semantic Web Conf. (ISWC'13). LNCS, vol. 8218, pp. 558–573. Springer (2013)
15. Rudolph, S., Krötzsch, M.: Flag & check: Data access with monadically defined queries. In: Hull, R., Fan, W. (eds.) Proc. 32nd Symposium on Principles of Database Systems (PODS'13). pp. 151–162. ACM (2013)
16. Walsh, T. (ed.): Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). AAAI Press/IJCAI (2011)

Singular Referring Expressions in Conjunctive Query Answers: the case for a $\mathcal{CFD}$ DL Dialect

Alexander Borgida[†], David Toman[‡] and Grant Weddell[‡]

[†]Department of Computer Science
Rutgers University, New Brunswick, USA
borgida@cs.rutgers.edu

[‡]Cheriton School of Computer Science
University of Waterloo, Canada
{david,gweddell}@uwaterloo.ca

Abstract. A referring expression in linguistics is any noun phrase identifying an object in a way that will be useful to interlocutors. In the context of conjunctive queries over a description logic knowledge base (DL KB), typically *constant symbols* (usually treated as rigid designators) are used as referring expressions in a certain answer to the query. In this paper, we begin to explore how this can be usefully generalized by allowing more general DL concept descriptions, called *singular referring expression*s, to replace constants in this role. In particular, we lay the foundation for singular referring expressions in conjunctive query answers over a DL KB using a member of the CFD family of DL dialects. In the process, we introduce a specific language for referring concept types, and present initial results on how conjunctive queries with referring concept types can be efficiently supported.

1 Introduction and Motivation

Query answering in logic-based approaches to data and knowledge bases has traditionally been viewed as finding constant names, appearing in the knowledge-base, which can be substituted for the variables of the query. More formally, a query $q(x_1, \ldots, x_n)$ is viewed as a formula with free variables $x_1, \ldots, x_n$ and, if the knowledge-base $\mathcal{K}$ contains individual constant names IN, query answering consists of computing the set $\{\, (a_1, \ldots, a_n) \mid a_i \in \mathsf{IN}, \mathcal{K} \models q(a_1/x_1, \ldots, a_k/x_k) \,\}$. We believe that in a number of circumstances this is less than ideal.

(1) In object-based KBMSs (including Object-Relational, XML and Object-Oriented DBMSs, as well as DLs with UNA), all known individual objects must have unique (internal) distinguishing identifiers. However, these identifiers are often insufficient to allow users to figure out what real-world object they refer to, especially for large KBs. For example, system generated `ref` expressions in object-oriented databases [10] and blank node identifiers in RDF are semantically opaque to end-users. A specific example of this are identifiers that individual authors or the system must invent in community-developed ontologies such as

Freebase [2]. There, for example, the `id` of the "Synchronicity" album by the Police is `"/guid/9202a8c04000641f8000000002f9e349"` (as of April, 2015.)

(2) In Relational DBMSs, the above problem is supposedly avoided by using "external keys": tuples of attributes whose values (strings, integers, ...) uniquely identify rows of tables. Problem (1) above will then arise in OBDA access to legacy relational systems, since the ontology will surely be object-based.[1]

We note that even in databases, universally unique keys are hard to find (e.g., newly arrived foreign students do not have *ssn#*), though they may work for subsets of individuals, such as those returned by queries.

(3) Additional problems for finding identifying attributes for classes of objects arise in conceptual modeling. For example, consider all cases where Extended Entity-Relationship modeling creates a new heterogeneous entity set by "generalization" [5] from others. For example, we want to generalize *Person* (whose key might be *ssn#*) and *Company* (whose key might be *tickerSymbol*) to *LegalEntity*, which can own things. In EER modeling, such a situation forces the introduction of a new, artificial attribute as a key, with the attendant problems. Yet when we retrieve a set of legal entities, we can reference them in different, more natural ways, depending on which subclass they belong to.

(4) The next example illustrates a subtler version of the above: consider the following hierarchy of concepts relating to publications:

$$\mathit{Journal} \sqsubseteq \mathit{EditedCollection} \quad , \quad \mathit{EditedCollection} \sqsubseteq \mathit{Publication}$$

And suppose edited collections are identified by *isbn#*, while journals are identified by *title* and *publisher*. When we retrieve a set of objects in *Publication*, we would like to describe them in different ways, depending on the subclass they belong to; but in this case, there would be an additional preference for $\mathit{textittitle}, \mathit{publisher}$) over *isbn#* for elements of *Journal*.

(5) Many kinds of KBMSs, including those based on DLs and FOL, allow us to describe situations where objects can be inferred to exist, without having an explicit (internal) identifier. For example, if Michelle is a person, then she has a mother, and if she is married then she has a spouse. Normally, such objects cannot be returned in the list of answers. This is all the more unpleasant if we can capture information about this unknown person, such as the phone number of Michelle's mother: $\{\mathit{Michelle}\} \sqsubseteq \exists \mathit{hasMother}.\exists \mathit{hasPhone}.\{1234567\}$. Yet it is common in human communication to identify objects by their relationship to other known objects. For example, "Michelle's mother" is a perfectly reasonable *intensional description* of someone who has phone `1234567`.

The standard response to some of the above problems would be to have the user modify the query by finding the appropriate values for identifying attributes (external keys). For example, instead of the query `q1(x) :- Journal(x)`, the programmer would be expected to write

$$\mathtt{q2(t,p)} : -\mathtt{Journal(x), hasTitle(x,t), hasPublisher(x,p)}.$$

[1] OBDA systems such as Mastro [4] and others try to deal with this issue by using function symbols over database keys to generate "object terms" that act as object identifiers. The name of the function symbols is however not semantically motivated.

This approach has several problems: (i) In the enumeration of answers to `q2`, the relationship between the original object of interest, `x`, and its descriptors, `t` and `p`, is lost; something akin to *"objects x with* `title =` t *and* `publisher =` p*"* would be more desirable. (ii) The above reformulation cannot be done using regular conjunctive queries in the case of item 4 above, because the answer for edited collections that are not journals should be identified by *isbn#*, for which the query is

$$\texttt{q5}(\texttt{isb}) : -\texttt{EditedCollection}(\texttt{x}), \neg\texttt{Journal}(\texttt{x}), \texttt{hasIsbn}(\texttt{x}, \texttt{isbn})$$

which is not a conjunctive query, since it includes a negation. (iii) From the point of view of software engineering, the task of choosing these identifying references is *mentally distinct* from the task of selecting the objects of interest to begin with. Both SQL's **select** clause, and XQuery's **return** clause are examples of separating these two aspects in existing query languages.

This paper is then dedicated to the task of proposing a *first solution* to (some of) the issues raised by providing "singular referring expressions" in the place of individuals returned by conjunctive queries, in the context of DL KBs.

Our plan and contributions are as follows: We will start by proposing a language for referring concept expressions and types. This language will generalize the usual case of presenting answers to queries as individual names to situations that: (i) allow object identification by key (paths), possibly within the limited context of some concept instances; (ii) deal with heterogeneous answer sets, such as *LegalEntity*; and (iii) allow preferential choice of referring expressions, as for *EditedCollection*. We will use this language to define answers for conjunctive queries over DL KBs. More generally, the proposed approach introduces a new separation of concerns for knowledge bases (identification vs. qualification). In our case, the query head will annotate each variable returned with an *answer concept type*; this will be instantiated to an *answer concept* (a subset of our DL concepts) for each answer; such concepts will eventually bottom out to individual constants, rather than atomic concepts.

Because we wish to generalize the usual case of constant names in answers, we desire referring concept types to be singular expressions – i.e., to identify **one** individual.[2] Unfortunately, without knowing anything else, it is impossible, for example, to tell whether an expression such as "object with p-value 3" will be singular or not: if p is a key, then yes, but not otherwise. Therefore, we need to use information from the ontology to verify the singularity of referring concept type. This can be extended by examining the body of the query (and hence learning more about what possible values variables may take). We will concentrate in this paper on: (i) the (compile-time) analysis of conjunctive query bodies in the context of the TBox to determine whether a referring concept *type* will return a reference to at most one individual; (ii) the reformulation of the query to

[2] Researchers interested in so-called *co-operative query answering* have considered *returning predicates/concepts describing sets of individuals* (e.g., [1, 3, 6, 8]), where an answer to the query "Who can take the Data Structures course?" might include, "Anyone who has passed the Intro to Computer Science course with at least a C grade". Please note that we are not considering that problem in this paper.

guarantee that the referring expression will indeed return exactly 1 value. Our technical results will show that this can be done in polynomial time for the DL $\mathcal{CFD}^{\forall}_{nc}$, which allows the capture of identification constraints such as keys.

2 Preliminaries: the Description Logic $\mathcal{CFD}^{\forall}_{nc}$

The knowledge bases that we consider are based on the logic $\mathcal{CFD}^{\forall}_{nc}$, a recent member of the $\mathcal{CFD}$ family of DL dialects. All members of this family are fragments of FOL with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *attributes*. Although attributes deviate from the normal practice of using binary predicate symbols called *roles*, they make it easier to incorporate concept constructors suited to the capture of relational data sources that include various dependencies, e.g., by a straightforward reification of arbitrary n-ary predicates, and also make it easier to explore varieties of concepts that can serve as referring expressions.

Definition 1 ($\mathcal{CFD}^{\forall}_{nc}$ Concepts) Let F, PC and IN be disjoint sets of (names of) *attributes*, *primitive concepts* and *individuals*, respectively. A *path function* Pf is a word in F^* with the usual convention that the empty word is denoted by id and concatenation by ".". The set of $\mathcal{CFD}^{\forall}_{nc}$ *concepts* C is given by the following grammar, where $a \in \mathsf{IN}$, $A \in \mathsf{PC}$, Pf and Pf_i are path functions, $k > 0$ and $f_i \in \mathsf{F}$.

$$\begin{aligned} C ::= {} & \{a\} \mid A \mid \forall Pf.C \mid C_1 \sqcap C_2 \mid \neg A \mid Pf_1 = Pf_2 \mid \\ & A : Pf_1.Pf, Pf_2, \ldots, Pf_k \to Pf_1 \mid A : Pf_1.Pf, Pf_2, \ldots, Pf_k \to Pf_1.f \end{aligned} \tag{1}$$

Semantics is defined in the standard way with respect to an interpretation $\mathcal{I} = (\triangle, (\cdot)^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretation of attributes f to be total functions on $\triangle$, primitive concepts A to be subsets of $\triangle$, and individuals a, b to be elements of $\triangle$. The

$$\begin{aligned} (\{a\})^{\mathcal{I}} &= \{(a)^{\mathcal{I}}\}, \\ (\forall Pf.C)^{\mathcal{I}} &= \{x \in \triangle \mid (Pf)^{\mathcal{I}}(x) \in (C)^{\mathcal{I}}\}, \\ (C_1 \sqcap C_2)^{\mathcal{I}} &= (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}, \\ (\neg A)^{\mathcal{I}} &= \triangle \setminus (A)^{\mathcal{I}}, \\ (Pf_1 = Pf_2)^{\mathcal{I}} &= \{x \in \triangle \mid (Pf_1)^{\mathcal{I}}(x) = (Pf_2)^{\mathcal{I}}(x)\} \text{ and} \\ (A : Pf_1, \ldots, Pf_k \to Pf)^{\mathcal{I}} &= \{x \in \triangle \mid \forall\, y \in (A)^{\mathcal{I}} : \\ & \qquad \textstyle\bigwedge_i (Pf_i)^{\mathcal{I}}(x) = (Pf_i)^{\mathcal{I}}(y) \to (Pf)^{\mathcal{I}}(x) = (Pf)^{\mathcal{I}}(y)\} \end{aligned}$$

Fig. 1. Semantics of $\mathcal{CFD}^{\forall}_{nc}$ Concepts.

interpretation function is extended to path expressions by interpreting id as the identity function and concatenation as function composition. The semantics of the remaining $\mathcal{CFD}^{\forall}_{nc}$ concepts are then defined in Figure 1. □

Concepts having the last two forms in (1) are called a *(path) key* and a *path functional dependency* (PFD), respectively. Informally, such concepts each de-

note a set of objects, each of which, whenever agreeing with any A-object on all left-hand-side path functions, also agrees with that object on the right-hand-side path function. Thus, the axiom $EditedCollection \sqsubseteq EditedCollection : isbn\# \rightarrow id$, expresses that $isbn\#$ is a key for edited collections, while $Person \sqsubseteq Person : home.phone\# \rightarrow home.address$, says that if two persons have the same home phone then they have the same home address.

Definition 2 ($\mathcal{CFD}^{\forall}_{nc}$ Knowledge Bases) Generic knowledge/metadata and specific facts/data in a $\mathcal{CFD}^{\forall}_{nc}$ knowledge base $\mathcal{K}$ are respectively defined by a *TBox* $\mathcal{T}_{\mathcal{K}}$ and an *ABox* $\mathcal{A}_{\mathcal{K}}$.

A TBox $\mathcal{T}_{\mathcal{K}}$ consists of a finite set of *general concept inclusion axioms*, which adhere to one of the following six forms, where A and A_i are primitive concepts, f is an attribute in F, B is a primitive concept or a negation of a primitive concept, and where Pf and Pf_i are path functions:

$A \sqsubseteq B;\ A \sqsubseteq \forall f.B;\ \forall f.A \sqsubseteq B;\ A \sqsubseteq (Pf_1 = Pf_2);\ A_1 \sqsubseteq A_2 : Pf_1, \ldots, Pf_k \rightarrow Pf.$

$\mathcal{T}_{\mathcal{K}}$ must also satisfy the following condition:

stratification of path function equalities: If $A \sqsubseteq (Pf_1 = Pf_2) \in \mathcal{T}_{\mathcal{K}}$ then A is a primitive concept that does not occur on the right-hand-side of *any* inclusion axiom in $\mathcal{T}_{\mathcal{K}}$.

An ABox $\mathcal{A}_{\mathcal{K}}$ consists of a finite set of axioms that express facts, each of which has one of the following two forms, respectively called *individual membership assertions* and *individual relationship assertions*,

$$A(a) \quad \text{and} \quad Pf_1(a) = Pf_2(b),$$

for individuals $a, b \in \mathsf{IN}$, primitive concept $A \in \mathsf{PC}$, and path functions $Pf_i \in \mathsf{F}^*$. Note that $(\{a\})^{\mathcal{I}} = \{(a)^{\mathcal{I}}\}$ and thus we can use nominal concepts as proxies for individuals. An interpretation $\mathcal{I}$ satisfies an inclusion axiom $C_1 \sqsubseteq C_2$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$. It satisfies ABox axioms $A(a)$ and $Pf_1(a) = Pf_2(b)$ if $(a)^{\mathcal{I}} \in (A)^{\mathcal{I}}$ and $(Pf_1)^{\mathcal{I}}((a)^{\mathcal{I}}) = (Pf_2)^{\mathcal{I}}((b)^{\mathcal{I}})$, respectively. $\mathcal{I}$ satisfies a knowledge base $\mathcal{K}$ if it satisfies each axiom in $\mathcal{K}$. □

The need for the additional restriction on a TBox to avoid undecidability of TBox reasoning due to equational constraints derives in a straightforward way from the undecidability of the word problem for monoids [7, 9].

Proposition 3 (Consistency and Logical Implication in $\mathcal{CFD}^{\forall}_{nc}$ [11, 13]) Knowledge base consistency and logical implication for $\mathcal{CFD}^{\forall}_{nc}$ are complete for PTIME. □

3 Conjunctive Queries and Certain Answers

In this section we re-evaluate the way answers to conjunctive queries are understood and presented.

Definition 4 (Conjunctive Queries) A *conjunctive query* (CQ) Q, with free variables $\{x_1, \ldots, x_k\}$, has the form $\exists x_{k+1}, \ldots, \exists x_m : \textsc{Body}(Q)$ where $\textsc{Body}(Q)$,

the *query body of* Q, is a first order formula over the signature $\mathsf{C} \cup \mathsf{F}$ of the form

$$\left(\bigwedge C(x_i)\right) \wedge \left(\bigwedge f(x_j) = x_k)\right), \tag{2}$$

where each x_i, x_j and x_k occurs in $\{x_1, \ldots, x_m\}$.[3] □

Recall that our objective in this paper is to study how $\mathcal{CFD}^{\forall}_{nc}$ concepts can help serve the role of a singular referring expression in certain answers to conjunctive queries. To review current practice, assume $\mathcal{K}$ is a knowledge base over some DL dialect, and consider one of the purposes served by an ABox in defining the *certain answers* over $\mathcal{K}$ to a CQ Q: the finite collection of individual names $\{a_1, \ldots, a_n\}$ in the ABox defines a space of n^k *potential answers* to Q: k-tuples θ_i that map query variables x_j to individual names a_{i_j},

$$\{x_1 \mapsto a_{i_1}, \ldots, x_k \mapsto a_{i_k}\}.$$

Viewed as a substitution, recall that each θ_i is a *certain answer* to Q over $\mathcal{K}$ exactly when $\mathcal{K} \models Q\theta_i$. Note that the occurrence of an individual a_{i_j} in a certain answer is a simple example of a *referring expression*, i.e., a syntactic artifact that identifies elements of an underlying domain.

This notion of a potential answer is easily modified to accommodate a much larger variety of referring expressions. To start, one can view a potential answer θ_i as a set of size k that maps query variables to *nominal concepts* instead of individuals, as in

$$\{x_1 \mapsto \{a_{i_1}\}, \ldots, x_k \mapsto \{a_{i_k}\}\},$$

and then extend this idea by allowing *arbitrary* $\mathcal{CFD}^{\forall}_{nc}$ concepts in potential answers to queries, i.e., potentially by allowing answers to have the form

$$\{x_1 \mapsto C_1, \ldots, x_k \mapsto C_k\}. \tag{3}$$

In this alternative setting "certain answers" are defined as follows:

Definition 5 (Referring Concepts and Certain Answers)
Let $\mathcal{K}$ be a $\mathcal{CFD}^{\forall}_{nc}$ knowledge base, Q a conjunctive query with free variables $x_1, \ldots, x_k$ and $C_1, \ldots, C_k$ $\mathcal{CFD}^{\forall}_{nc}$ concept descriptions. We say that $C_1, \ldots, C_k$ are *referring concepts* in a *certain answer* $\{x_1 \mapsto C_1, \ldots, x_k \mapsto C_k\}$ *to* Q if the following two conditions hold:

1. $\mathcal{K} \models \forall x_1, \ldots, x_k.C_1(x_1) \wedge \ldots \wedge C_k(x_k) \rightarrow Q$, and
2. $|\{o \in \triangle \mid \mathcal{I}, [x_i \mapsto o] \models C_i(x_i) \wedge (\exists x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_k.Q)\}| = 1$
   for every $\mathcal{I} \models \mathcal{K}$ and $0 < i \leq k$.

where $C(x)$ is the first-order formula derived from the concept description C. □

The first condition states that C_i objects (as values of x_is) satisfy Q and the second one that we are interested in *singular* referring expressions C_i, as generalizations of simple individual names. In the rest of the paper we call $\{x_1 \mapsto C_1, \ldots, x_k \mapsto C_k\}$ a *candidate answer to* Q if condition (1) in the above definition

[3] To improve readability in the rest of the paper we allow constants to appear in CQs. However, conjuncts of the form $x = a$ are just syntactic sugar for a conjunct $\{a\}(x)$ formed using a concept $\{a\}$. Similarly, $f(x) = a$ is $\exists y(f(x) = y \wedge \{a\}(y))$, etc.

is satisfied,and call it a *weakly identifying answer to* Q if only an *upper bound* (of one) is guaranteed to hold in condition (2) We call concepts C_i that are used in this way *singular referring concepts* since they replace the role of individual names as referring expressions in certain answers to conjunctive queries.

To illustrate, assume $\mathcal{K}$ captures information about persons, and consider a query with body Person(x). In the rest of the paper, we will use $Pf = a$ as an alternative syntax for $\forall Pf.\{\, a \,\}$ (to improve readability). Possibilities for certain answers to the query now include one or more of the following:

$$\{x \mapsto (ssn\# = 1234)\}$$

or

$\{x \mapsto Female \sqcap (hasSpouse.name = \text{'Enya'}) \sqcap (hasSpouse.phone\# = 1234567)\}$.

Note that Definition 5(2) disallows answers of the form $\{x \mapsto Female\}$ or more generally, rules out any C for x in which $|(C)^{\mathcal{I}}| \neq 1$ when $\mathcal{I} \models \mathcal{K}$. Thus, the earlier examples of certain answers would be contingent on $\mathcal{T}_{\mathcal{K}}$ ensuring that persons have unique $ssn\#$, as well as unique spouses, who can be identified by a $(name, phone\#)$ pair.

4 Referring Concept Types in Conjunctive Queries

Allowing referring concepts beyond nominals in certain answers to a query Q may lead to infinitely many syntactically distinct certain answers. In this section, we develop a framework that ensures the set of certain answers to any conjunctive query is finite by introducing a specific language for *referring concept types*, which bottom out at individual nominals.

Important note: while Definition 5 allows general $\mathcal{CFD}^{\forall}_{nc}$ concepts to serve as *referring concepts*, in the rest of the paper we restrict our attention to the subset of referring concepts adhering to the more limited grammar

$$C \;::=\; \{a\} \;\mid\; A \;\mid\; \forall Pf.C \;\mid\; C \sqcap C$$

where $\{a\}$ is a nominal, A is a primitive concept name, and $Pf \in \mathsf{F}^*$. These C are intuitively instances of the following referring *types*.

Definition 6 (Referring Concept Types)
A *referring concept type* Rt is given by the following grammar, where T denotes a finite conjunction of primitive concepts (or $\top$ standing for empty conjunction), to be called henceforth a *simple type*.

$$Rt \;::=\; \{?\} \;\mid\; Pf = \{?\} \;\mid\; Rt_1 \sqcap Rt_2 \;\mid\; T \to Rt \;\mid\; Rt_1; Rt_2$$

The *referring concept set* $\mathrm{RC}(Rt, \mathcal{K})$ is the "extension" of a referring concept type Rt with respect to KB $\mathcal{K}$, and is defined inductively as follows, where S_i is short for $\mathrm{RC}(Rt_i, \mathcal{K})$:

1. $\mathrm{RC}(\{?\}, \mathcal{K}) = \{\{a\} \mid a$ occurs in $\mathcal{A}_{\mathcal{K}}\}$;
2. $\mathrm{RC}(Pf = \{?\}, \mathcal{K}) = \{(Pf = \{a\}) \mid a$ occurs in ABox $\mathcal{A}_{\mathcal{K}}\}$;
3. $\mathrm{RC}(Rt_1 \sqcap Rt_2, \mathcal{K}) = \{C_1 \sqcap C_2 \mid C_i \in S_i\}$;
4. $\mathrm{RC}(T \to Rt_1, \mathcal{K}) = \{T \sqcap C \mid C \in S_1\}$; and

5. $\mathrm{RC}(Rt_1; Rt_2, \mathcal{K}) = S_1 \cup \{C_2 \in S_2 \mid \neg\exists C_1 \in S_1 \text{ s.t. } \mathcal{K} \models C_1 \equiv C_2\}$.

A referring concept type is *homogeneous* if it is free of any occurrence of the construct in 5. □

Examples of their use will follow immediately after the next definition. Note that, as desired, the set of referring concepts associated with a single referring concept type is finite if $\mathcal{A}_\mathcal{K}$ is finite.

Definition 7 (Certain Answers and Singular Referring Concepts) Let Q be a conjunctive query with free variables $\{x_1, \ldots, x_k\}$. A *query head* H for Q is a set of pairs $\{x_1 : Rt_1, \ldots, x_k : Rt_k\}$ that associates a referring concept type Rt_i with each x_i.

The *set of certain answers* to Q with respect to a head H and a knowledge base $\mathcal{K}$, denoted $\textsc{Ans}(Q, H, \mathcal{K})$, is the set of all certain answers $\{x_i \mapsto C_i \mid 0 < i \leq k\}$ to Q over $\mathcal{K}$ for which $C_i \in \mathrm{RC}(Rt_i, \mathcal{K})$, for $0 < i \leq k$. □

After the examples below, the objective in this section is to show that computing $\textsc{Ans}(Q, H, \mathcal{K})$ can be achieved in PTIME for $\mathcal{CFD}^{\forall}_{nc}$, provided that referring concept types satisfy a "weak identification condition".

The following examples illustrate the use of referring concepts with conjunctive queries to extend the current situation with the more expressive cases motivated in the Introduction (a conjunctive query Q with a head H will be written in the following SQL-like style: **select** H **where** $\textsc{Body}(Q)$):

1. (expressing the current case) *"Any journals published by Italians"*
   select $x_1 : \{?\}$
   where $Journal(x_1) \wedge (publishedBy(x_1) = x_2) \wedge Italian(x_2)$
2. (reference via single key) *"The ssn# of any person with phone 12345567"*
   select $x : ssn\# = \{?\}$
   where $Person(x) \wedge (phone\#(x) = 1234567)$
3. (multiple attribute key) *"The title and publisher of any journals"*
   select $x : title = \{?\} \sqcap publishedBy = \{?\}$
   where $Journal(x)$
4. (choice of identification in heterogeneous set) *"Any legal entity"*
   select $x : Person \rightarrow ssn\# = \{?\}$; $Company \rightarrow tickerSymbol = \{?\}$
   where $LegalEntity(x)$
   An example certain answer would be $x \mapsto Person \sqcap (ssn\# = 7654)$ while another might be $x \mapsto Company \sqcap (tickerSymbol = \texttt{'IBM'})$.
5. (preferred identification) *"Any publication, identified by its most specific identifier, when available."*
   select $x : Journal \rightarrow (title = \{?\} \sqcap publisher = \{?\})$;
   $EditedCollection \rightarrow isbn\# = \{?\}$; $\{?\}$
   where $Publication(x)$

We now make concrete our requirement that referring concepts occurring in query answers do indeed satisfy the ability to identify objects.

Definition 8 (Weak Identification in Certain Answers) Let Q be a conjunctive query, H a head for Q, and $\mathcal{T}$ a $\mathcal{CFD}^{\forall}_{nc}$ TBox. Q is *weakly identifying for H with respect to $\mathcal{T}$* if $|(C)^{\mathcal{I}}| \leq 1$ for every ABox $\mathcal{A}$, model $\mathcal{I}$ of $\mathcal{T} \cup \mathcal{A}$, and candidate answer θ to Q with respect to H and $\mathcal{T} \cup \mathcal{A}$ in which $x \mapsto C \in \theta$. □

Lemma 9 (Normal Form of Referring Concept Types) For every referring concept type Rt, there is an *equivalent normal form*

$$Rt_1; \ldots; Rt_n,$$

denoted $\textsc{Norm}(Rt)$, consisting of *tagged record types* Rt_i that are, in turn, homogeneous referring concept types of the form

$$T_i \to ((Pf_{i,1} = \{?\}) \sqcap \ldots \sqcap (Pf_{i,m_i} = \{?\})). \tag{4}$$

By *equivalent*, we mean that, for any KB $\mathcal{K}$, $C_1 \in \mathrm{RC}(Rt, \mathcal{K})$ implies there exists $C_2 \in \mathrm{RC}(\textsc{Norm}(Rt), \mathcal{K})$ for which $\{\,\} \models C_1 \equiv C_2$, and vise versa. □

Proof (sketch): By application of the following equivalence preserving rewrites.

$$\begin{array}{rcl}
\{a\} & \rightsquigarrow & \{\,\} \to (id = \{a\}) \\
(T \to Rt_1) \sqcap Rt_2 \text{ or } Rt_1 \sqcap (T \to Rt_2) & \rightsquigarrow & T \to (Rt_1 \sqcap Rt_2) \\
(Rp_1\,; Rt_2) \sqcap Rt_3 & \rightsquigarrow & (Rt_1 \sqcap Rt_3)\,; (Rt_2 \sqcap Rt_3) \\
Rt_1 \sqcap (Rt_2\,; Rt_3) & \rightsquigarrow & (Rt_1 \sqcap Rt_2)\,; (Rt_1 \sqcap Rt_3) \\
T \to (Rt_1\,; Rt_2) & \rightsquigarrow & (T \to Rt_1)\,; (T \to Rt_2) \\
T_1 \to (T_2 \to Rt) & \rightsquigarrow & (T_1 \sqcap T_2) \to Rt
\end{array}$$

□

We now present our first main result on deciding weak identification. Our theorem refers to the following auxiliary function $\mathcal{M}(\cdot)$ which abstracts query bodies as DL concepts (note that the function assumes, without harm, that variables are also elements of F):

$$\mathcal{M}(\phi) = \begin{cases} \forall\, x.C & \text{if } \phi = \text{“}C(x)\text{”}; \\ x_1.f = x_2 & \text{if } \phi = \text{“}f(x_1) = x_2\text{”}; \\ \mathcal{M}(\psi_1) \sqcap \mathcal{M}(\psi_2) & \text{if } \phi = \text{“}\psi_1 \wedge \psi_2\text{”}; \text{ and} \\ \mathcal{M}(\psi) & \text{otherwise, when } \phi = \text{“}\exists x_{k+1}, \ldots, \exists x_m : \psi\text{”}. \end{cases}$$

Theorem 10 (Deciding Weak Identification) Let Q be a conjunctive query, H a head for Q, and $\mathcal{T}$ a $\mathcal{CFD}^{\forall}_{nc}$ TBox. Q is *weakly identifying for H with respect to $\mathcal{T}$* if and only if

$$\mathcal{T} \cup \{A_Q \sqsubseteq ((\forall x.T) \sqcap \mathcal{M}(\textsc{Body}(Q))) \models A_Q \sqsubseteq A_Q : x.Pf_{i,1}, \ldots, x.Pf_{i,m_i} \to x$$

for all $(x : Rt) \in H$, and all $T \to (Pf_1 = \{?\}) \sqcap \ldots \sqcap (Pf_m = \{?\}) \in \textsc{Norm}(Rt)$.

Proof (sketch): The if direction is straightforward. For the only-if direction, consider the earliest tagged record type $Rt' \in \textsc{Norm}(Rt)$ for some $x : Rt \in H$ for which the check for the logical consequence of the key PFD fails. Then one can construct an ABox $\mathcal{A}$ where there exists $C \in \mathrm{RC}(Rt', \mathcal{T} \cup \mathcal{A})$ that occurs in some candidate answer θ for Q for which one can also construct an interpretation $\mathcal{I}$ for $\mathcal{T} \cup \mathcal{A}$ for which $|(C)^{\mathcal{I}}| > 1$. □

Definition 11 Let Q be a CQ with free variables $\{x_1, \ldots, x_k\}$. We say that H is a *homogeneous head for Q* if it is of the form

$$H = \{x_i : T_i \to (Pf_{i,1} = \{?\}) \sqcap \ldots \sqcap (Pf_{i,\ell_i} = \{?\}) \mid 0 < i \leq k\}.$$

For every CQ Q and a homogeneous head H for Q we define a conjunctive query

$$Q_H = Q \wedge \bigwedge_{i=1}^{k} (T_i(x_i) \wedge (Pf_{i,1}(x_i) = x_{i,1} \wedge \ldots \wedge (Pf_{i,\ell_i} = x_{i,\ell_i})$$

with additional free variables $\{x_{i,j} \mid 0 < i \leq k, 0 < j \leq \ell_i\}$. □

To handle preferences, i.e., non-homogeneous heads H of queries of the form $\{x_i : Rp_{i,1}; \ldots; Rp_{i,n_i} \mid 0 < i \leq k\}$ we first define a sequence of homogeneous heads

$$H_{j_1,\ldots,j_k} = \{x_i : Rp_{i,j_i} \mid 0 < i \leq k\}$$

over all $0 < j_i < n_i$. In addition, we define a *normalized* ABox $\mathcal{A}'$ for an ABox $\mathcal{A}$ to contain only individual relationship assertions of the form $f(a) = b$ obtained from $\mathcal{A}$ by introducing additional individuals for the intermediate individuals participating in $\mathcal{A}$'s individual relationship assertions.[4]

Theorem 12 Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, Q a CQ, and H a head for Q, such that Q is weakly identifying for H in $\mathcal{T}$. Then

$$\{x_i \mapsto T_i \sqcap (Pf_{i,1} = \{a_{i,1}\}) \sqcap \ldots \sqcap (Pf_{i,\ell_i} = \{a_{i,\ell_i}\})\} \in \textsc{Ans}(Q, H, \mathcal{K})$$

if and only if

$$\{x_i \mapsto \{b_i\}, x_{i,j} \mapsto \{a_{i,j}\} \mid 0 < i \leq k, 0 < j \leq \ell_i\} \in \textsc{Ans}(Q_{H_{j_1,\ldots,j_k}}, H_0, \mathcal{K}')$$

and there is no

$$\{x_i \mapsto \{b_i\}, x_{i,j} \mapsto \{a'_{i,j}\} \mid 0 < i \leq k, 0 < j \leq \ell_i\} \in \textsc{Ans}(Q_{H_{j'_1,\ldots,j'_k}}, H_0, \mathcal{K}')$$

for $H_{j'_1,\ldots,j'_k}$ dominates $H_{j_1,\ldots,j_k}$, where $H_0 = \{x_i : \{?\} \mid 0 < i \leq k\} \cup \{x_{i,j} : \{?\} \mid 0 < i \leq k, 0 < j \leq \ell_i\}$, $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ where $\mathcal{A}'$ is a normalized $\mathcal{A}$, and $a_{i,j}$ and $a'_{i,j}$ are constants in $\mathcal{A}$.

Proof (sketch): Consider first the case where H is homogeneous. Then we can reconstruct an answer to Q and H from an answer to Q_H and H_0 (which consists of nominal concepts only. Conversely, from answer to Q and H we can extract nominal concepts that are an answer to $\exists x_1, \ldots, x_k.Q_H$ and H_0 (restricted to free variables of $\exists x_1, \ldots, x_k.Q_H$). This answer is then extended to Q_H and (full) H_0 by using individual names introduced in the normalized ABox $\mathcal{A}'$, this extension is unique since Q is weakly identifying for H in $\mathcal{T}$.

For the non-homogeneous case we simply consider all possible homogeneous subcases and then filter the answers by the valuations of the variables $x_1, \ldots, x_k$ (since those describe the possibly anonymous individuals in $\mathcal{A}$ using their system-assigned names A'). □

Note that the queries $Q_{H_{j'_1,\ldots,j'_k}}$ are answered in $\mathcal{K}$ with respect to H_0, a *trivial* and homogeneous head: this reduces the answering to *standard* CQ query answering in $\mathcal{CFD}^{\forall}_{nc}$ knowledge base $\mathcal{K}$ as introduced in [12].

Corollary 13 Computing certain answers to conjunctive queries with respect to referring concepts and $\mathcal{CFD}^{\forall}_{nc}$ knowledge bases is complete for PTIME.

[4] Such a transformation is already part of the CQ answering algorithm [12]. Note that in our setting, however, the new individuals cannot participate in Q's answers.

5 Conclusions

The paper's contributions are as follows.

First and foremost, on the non-technical side, it recognized and motivated the utility of "singular referring expressions" for query answers, which are more complex than just nominals, and it argued for the need for a new separation of concerns in query writing: qualification (what the query body does) vs. identification (how results are presented).

On the specification side, the paper defined formally the notion of "query answering using referring expressions" for certain answers in conjunctive queries over DLs. In the context of $\mathcal{CFD}_{nc}^{\forall}$, it introduced a specific language for referring expressions, which are a subset of $\mathcal{CFD}_{nc}^{\forall}$ concepts, and which allows us to handle all the motivating problems (except intensional descriptions in this paper. This language generalizes the notion of nominal, currently used in OBDA, to handle keys (as found in both the database and DL KB literature), and supports heterogeneous sets (as in the case of *LegalEntity*), as well as preferential choice of referring expressions (as in the *EditedCollection* example).

On the algorithmic and complexity side, it first considered, in the context of $\mathcal{CFD}_{nc}^{\forall}$, the problem of determining (in polynomial time) whether a referring concept type was "weakly" identifying in the context of a query and TBox, in the sense that its instances necessarily referred to at most one object. It also showed how one can transform a query and knowledge base so that the answers had cardinality one. This led to the result that computing certain answers to conjunctive queries with respect to referring concepts and $\mathcal{CFD}_{nc}^{\forall}$ KBs was complete for PTIME.

There are many avenues left to explore in this work. We have already mentioned that lack of space prevented us from considering referring concepts types for "Michelle's mother", which use inverse functions. Another direction to consider are additional forms or desirable properties of referring expressions. For example, the variety of references raises a problem: the same object may be returned in an answer with different references to it, even if the knowledge base works with the UNA. As a simplest example, a journal has multiple candidate keys. We are currently investigating ways to reason about and avoid such duplication.

The reader may also have observed that so far it was up to the programmer to select the referring expression(s) to consider for each variable. A form of type inference on the query variables would be useful, as the basis of a tool which would suggest to the user a (bounded) list of possible referring expressions that are guaranteed to have the singular reference property with respect to a particular TBox.

Of course, all the technical questions considered in this paper will have different answers for different DLs. Therefore, an orthogonal avenue of research is to re-consider these issues in the context of other DLs, especially DL-Lite.

Acknowledgments: We wish to thank those reviewers who have made excellent suggestions for improving the paper, and financial support from NSERC Canada.

References

1. Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. Dl techniques for intensional query answering in oodbs. In *Working Notes of the KI'95 Workshop: KRDB-95 Reasoning about Structured Objects: Knowledge Representation Meets Databases*, page 3 pages, 1995.
2. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM, 2008.
3. Alexander Borgida. Description logics in data management. *Knowledge and Data Engineering, IEEE Transactions on*, 7(5):671–682, 1995.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
5. Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems, 3rd Edition*. Addison-Wesley-Longman, 2000.
6. Tomasz Imielinski. Intelligent query answering in rule based systems. *The Journal of Logic Programming*, 4(3):229–257, 1987.
7. Andrey Andreyevich Markov, Jr. On the impossibility of certain algorithm in the theory of associative systems. *Dokl. Akad. Nauk SSSR*, 55:587–590, 1947.
8. Amihai Motro. Intensional answers to database queries. *Knowledge and Data Engineering, IEEE Transactions on*, 6(3):444–454, 1994.
9. Emil Post. Recursive unsolvability of a problem of Thue. *The Journal of Symbolic Logic*, 12:1–11, 1947.
10. Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, 4th Edition*. McGraw-Hill Book Company, 2005.
11. David Toman and Grant E. Weddell. Conjunctive query answering in $\mathcal{CFD}_{nc}$: A PTIME description logic with functional constraints and disjointness. In *AI 2013: Advances in Artificial Intelligence - 26th Australasian Joint Conference, Dunedin, New Zealand, December 1-6, 2013. Proceedings*, pages 350–361, 2013.
12. David Toman and Grant E. Weddell. Answering Queries over $\mathcal{CFD}^{\forall}_{nc}$ Knowledge Bases. Technical Report CS-2014-14, Cheriton School of Computer Science, University of Waterloo, 2014.
13. David Toman and Grant E. Weddell. On adding inverse features to the description logic $\mathcal{CFD}^{\forall}_{nc}$. In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014.*, pages 587–599, 2014.

Temporal Query Answering in $\mathcal{EL}^*$

Stefan Borgwardt and Veronika Thost

Theoretical Computer Science, TU Dresden, Germany
firstname.lastname@tu-dresden.de

Motivation Context-aware systems use data collected at runtime to recognize predefined situations and trigger adaptations; e.g., an operating system may use sensors to recognize that a video application is out of user focus, and then adapt application parameters to optimize the energy consumption. Using ontology-based data access [12, 19], the situations can be encoded into queries that are answered over an ABox containing the sensor data. In the TBox, we can encode background knowledge about the domain. For example, if the user has been working with another application on a second screen for a longer period, then we may assume that he does not need the video to be displayed in the highest resolution.

In this paper, we focus on the lightweight DL $\mathcal{EL}$. We can state static knowledge about applications ($\mathsf{VideoApplication}(\mathsf{app1})$), dynamic knowledge about the current context ($\mathsf{NotWatchingVideo}(\mathsf{user1})$), as well as background knowledge like

$$\mathsf{VideoApplication} \sqcap \exists \mathsf{hasUser}.\mathsf{NotWatchingVideo} \sqsubseteq \exists \mathsf{hasState}.\mathsf{OutOfFocus},$$

saying that a video application whose user is currently not watching the video is out of user focus. Given such a knowledge base, we can use the *conjunctive query* (CQ) $\psi(x) := \exists y.\mathsf{hasState}(x, y) \wedge \mathsf{OutOfFocus}(y)$ to identify applications x that can potentially be assigned a lower priority. More complex situations typically depend also on the behavior of the environment in the past—the operating system should not switch configurations every time the user is not watching for one second, but only after this has been the case for a longer period.

For that reason, we investigate *temporal conjunctive queries* (TCQs), originally proposed in [3, 4]. They combine conjunctive queries via the operators of the propositional linear temporal logic LTL [14, 18]. We can use the TCQ

$$\big(\bigcirc^- \psi(x)\big) \wedge \big(\bigcirc^- \bigcirc^- \psi(x)\big) \wedge \big(\bigcirc^- \bigcirc^- \bigcirc^- \psi(x)\big) \wedge \big(\neg\big(\exists y.\mathsf{GotPriority}(y) \wedge \mathsf{notEqual}(x, y)\big)\,\mathsf{S}\,\mathsf{GotPriority}(x)\big)$$

to obtain all applications that were out of user focus during the three *previous* ($\bigcirc^-$) moments of observation, were prioritized by the operating system at some point in time, and the priority has *not* ($\neg$) changed *since* (S) then. The semantics of TCQs is based on *temporal knowledge bases* (TKBs), which, in addition to the TBox (which is assumed to hold *globally*, i.e., at every point in time), contains a *sequence* of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$, representing the data collected at specific

* Partially supported by the DFG in CRC 912 (HAEC).

points in time. We designate with n the most recent time of observation (the *current time point*), at which the situation recognition is performed. We also investigate the related temporalized formalism $\mathcal{EL}$-LTL, in which axioms, i.e., assertions or GCIs, are combined using LTL-operators.

Related Work The axioms in a TKB do not explicitly refer to time, but are written in a *classical* (atemporal) DL; only the query is temporalized. In contrast, [1,2,13,17] extend classical DLs by temporal operators that occur within concepts and axioms. However, most of these logics yield high reasoning complexities, even if the underlying atemporal DL is tractable. Lower complexities are obtained by considerably restricting either the temporal operators or the underlying DL.

Regarding temporal properties formulated over atemporal DLs, $\mathcal{ALC}$-LTL, a variant of $\mathcal{EL}$-LTL over the more expressive DL $\mathcal{ALC}$, was first considered in [6]. This was the basis for introducing TCQs over $\mathcal{ALC}$-TKBs in [3], which was extended to $\mathcal{SHQ}$ in [4]. However, reasoning in $\mathcal{ALC}$ is not tractable, and context-aware systems often need to deal with large quantities of data and adapt fast. TCQs over several lightweight logics have been regarded in [7], but only over a fragment of LTL without negation. In [1], the complexity of LTL over axioms of several members of the *DL-Lite* family of DLs has been investigated. However, nothing is known about TCQs over these logics.

Results We investigate the combined and data complexity of the TCQ entailment problem over TKBs formulated in $\mathcal{EL}$. Moreover, we determine the complexity of satisfiability of $\mathcal{EL}$-LTL-formulae, and additionally consider the special case where only *global GCIs* are allowed [6]. As usual, we consider *rigid* concepts and roles, whose interpretation does not change over time. In this regard, we distinguish three different settings, depending on whether concepts or roles (or both) are allowed to be rigid. Since rigid concepts can be simulated by rigid roles [6], only three cases need to be considered: (i) no symbols are allowed to be rigid, (ii) only rigid concepts are allowed, and (iii) both concepts and roles can be rigid. Tables 1 and 2 summarize our results and provide a comparison to related work. The only previously known results that directly apply here are P-hardness of CQ entailment in $\mathcal{EL}$ w.r.t. data complexity [11] and PSPACE-hardness of LTL [20]. Hence, we needed to prove three additional complexity lower bounds.

With a single exception, the complexity of TCQ entailment in $\mathcal{EL}$ turns out to be lower than that in $\mathcal{ALC}$ (and $\mathcal{SHQ}$) [4]. Regarding satisfiability in $\mathcal{EL}$-LTL, Table 2 shows that rigid symbols lead to an increase in complexity that does not affect *DL-Lite*$_{krom}$-LTL [1], and even matches the complexity of $\mathcal{ALC}$-LTL and $\mathcal{SHOQ}$-LTL in case (ii) [6, 15]. Thus, we partially confirm and refute the conjecture of [6] that $\mathcal{EL}$-LTL is as hard as $\mathcal{ALC}$-LTL. In the following, we shortly describe some of the ideas behind them. More details can be found in [8–10].

The upper bounds are obtained by a combination of techniques that were developed for $\mathcal{ALC}$-LTL [6] and refined for TCQs over $\mathcal{SHQ}$-TKBs [4], methods for checking LTL-satisfiability [4,20,21], and algorithms for atemporal reasoning

Table 1. The complexity of TCQ entailment. All results except the one for the data complexity of case (iii) from [4] are tight.

	Data Complexity			Combined Complexity		
	(i)	(ii)	(iii)	(i)	(ii)	(iii)
$\mathcal{EL}$	P	co-NP	co-NP	PSpace	PSpace	co-NExpTime
$\mathcal{ALC}/\mathcal{SHQ}$ [4]	co-NP	co-NP	ExpTime	ExpTime	co-NExpTime	2-ExpTime

Table 2. The complexity of satisfiability in LTL over DL axioms.

				Global GCIs		
	(i)	(ii)	(iii)	(i)	(ii)	(iii)
DL-Lite$_{krom}$ [1]	PSpace	PSpace	PSpace	PSpace	PSpace	PSpace
$\mathcal{EL}$	PSpace	NExpTime	NExpTime	PSpace	PSpace	PSpace
$\mathcal{ALC}$ [6]	ExpTime	NExpTime	2-ExpTime	ExpTime	ExpTime	2-ExpTime

in $\mathcal{EL}$ [5,16]. However, considerable work was necessary to obtain tight complexity bounds in all cases we considered. The main approach is to separate the temporal operators from the CQs (or axioms), which leaves us to solve a variant of the satisfiability problem for LTL (in P w.r.t. data complexity and in PSpace w.r.t. combined complexity), as well as the following problem for the DL part.

Definition 1. *Let* $\mathcal{K} = \langle \mathcal{T}, (\mathcal{A}_i)_{0 \leq i \leq n} \rangle$ *be a TKB and* $\alpha_1, \dots, \alpha_m$ *be CQs.*[1] *A set* $\mathcal{S} = \{X_1, \dots, X_k\} \subseteq 2^{\{\alpha_1, \dots, \alpha_m\}}$ *is* r-satisfiable *w.r.t. a mapping* $\iota \colon \{0, \dots, n\} \to \{1, \dots, k\}$ *and* $\mathcal{K}$ *if there are interpretations* $\mathcal{J}_1, \dots, \mathcal{J}_k$ *and* $\mathcal{I}_0, \dots, \mathcal{I}_n$ *such that*

- *they share the same domain and interpret all rigid symbols in the same way;*
- *each* $\mathcal{J}_i$ *is a model of* $\mathcal{T}$ *and* $\chi_i := \bigwedge X_i \wedge \bigwedge \{\neg \alpha_j \mid \alpha_j \notin X_i\}$*; and*
- *each* $\mathcal{I}_i$ *is a model of* $\langle \mathcal{T}, \mathcal{A}_i \rangle$ *and* $\chi_{\iota(i)}$*.*

Individually, the satisfiability of the conjunctions χ_i can be tested in P w.r.t. data complexity and in PSpace w.r.t. combined complexity. However, the problem is to ensure the first condition, namely that all rigid names are interpreted in the same way by all relevant interpretations.

In case (i), this restriction is obviously irrelevant. For case (iii), one can answer an exponentially large UCQ over an exponentially large atemporal knowledge base instead to obtain the upper bounds. The most difficult cases were case (ii) for the combined complexity of TCQ entailment, and the case of global GCIs in $\mathcal{EL}$-LTL, where we needed to obtain PSpace upper bounds in the presence of rigid names. For these cases, we proved that it suffices to guess additional data of polynomial size that can be added to the knowledge bases in order to separate the satisfiability tests in Definition 1. These tests can then be integrated into a PSpace-Turing machine for LTL-satisfiability [20] without increasing the complexity.

Acknowledgements We want to thank Franz Baader, Marcel Lippmann, and Carsten Lutz for fruitful discussions on the topic of this paper.

[1] In the case of $\mathcal{EL}$-LTL, these are axioms.

References

1. Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: Proc. of the 14th Int. Symp. on Temporal Representation and Reasoning (TIME'07), pp. 11–22. IEEE Press (2007)
2. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. ACM Transactions on Computational Logic 15(3), 25:1–25:50 (2014)
3. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13). pp. 330–344. Springer-Verlag (2013)
4. Baader, F., Borgwardt, S., Lippmann, M.: Temporal query entailment in the description logic $\mathcal{SHQ}$. Journal of Web Semantics (2015), doi:10.1016/j.websem.2014.11.008, in press.
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
6. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. ACM Transactions on Computational Logic 13(3), 21:1–21:32 (2012)
7. Borgwardt, S., Lippmann, M., Thost, V.: Temporalizing rewritable query languages over knowledge bases. Journal of Web Semantics (2015), doi:10.1016/j.websem.2014.11.007, in press.
8. Borgwardt, S., Thost, V.: LTL over $\mathcal{EL}$ axioms. LTCS-Report 15-07, Chair for Automata Theory, TU Dresden (2015), see http://lat.inf.tu-dresden.de/research/reports.html.
9. Borgwardt, S., Thost, V.: Temporal query answering in $\mathcal{EL}$. LTCS-Report 15-08, Chair for Automata Theory, TU Dresden (2015), see http://lat.inf.tu-dresden.de/research/reports.html.
10. Borgwardt, S., Thost, V.: Temporal query answering in the description logic $\mathcal{EL}$. In: Yang, Q. (ed.) Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI'15). AAAI Press (2015), to appear.
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 260–270. AAAI Press (2006)
12. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured information. In: Database Semantics: Semantic Issues in Multimedia Systems. pp. 351–369. Kluwer Academic Publisher (1998)
13. Gutiérrez-Basulto, V., Jung, J.C., Schneider, T.: Lightweight description logics and branching time: A troublesome marriage. In: Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'14). AAAI Press (2014)
14. Lichtenstein, O., Pnueli, A., Zuck, L.: The glory of the past. In: Proc. of the Workshop on Logics of Programs. pp. 196–218. Springer-Verlag (1985)
15. Lippmann, M.: Temporalised Description Logics for Monitoring Partially Observable Events. Ph.D. thesis, TU Dresden, Germany (2014), http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-147977
16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 2070–2075. AAAI Press (2009)

17. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Proc. of the 15th Int. Symp. on Temporal Representation and Reasoning (TIME'08). pp. 3–14. IEEE Press (2008)
18. Pnueli, A.: The temporal logic of programs. In: Proc. of the 18th Annual Symp. on Foundations of Computer Science (SFCS'77). pp. 46–57. IEEE Press (1977)
19. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal of Data Semantics 10, 133–173 (2008)
20. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the ACM 32(3), 733–749 (1985)
21. Wolper, P., Vardi, M.Y., Sistla, A.P.: Reasoning about infinite computation paths. In: Proc. of the 24th Annual Symp. on Foundations of Computer Science (SFCS'83). pp. 185–194. IEEE Press (1983)

Efficient Query Answering in DL-Lite through FOL Reformulation (Extended Abstract)

Damian Bursztyn[1], François Goasdoué[2] and Ioana Manolescu[1]

[1] INRIA & U. Paris-Sud, France [2] U. Rennes 1 & INRIA, France

Abstract. We propose a general query optimization framework for formalisms enjoying FOL reducibility of query answering, for which it reduces to the evaluation of a FOL query against facts. This framework allows searching within a set of alternative equivalent FOL queries, i.e., FOL reformulations, one with minimal evaluation cost when evaluated through a relational database management system. We provide two algorithms, an exhaustive and a greedy, for exploring the optimization space. This framework is applied to the lightweight description logic DL-Lite$_\mathcal{R}$ underpinning the W3C's OWL2 QL profile, for which an experimental evaluation validates the interest and applicability of our technique.

1 Introduction

Query answering in the lightweight DL-Lite$_\mathcal{R}$ description logic [1] has received significant attention in the literature, as it provides the foundations of the W3C's OWL2 QL standard for Semantic Web applications. In particular, query answering techniques based on FOL reducibility, e.g., [1,2,5,6,7], which reduce query answering against a knowledge base (KB) to FOL query evaluation against the KB's facts only (a.k.a. ABox) by compiling the KB's domain knowledge (a.k.a. TBox) into the query, hold great potential for performance. This is because FOL queries can be evaluated by a highly optimized Relational Database Management System (RDBMS) storing the KB's facts.

The goal of our study is to identify efficient techniques for query answering in description logics enjoying FOL reducibility, with a focus on DL-Lite$_\mathcal{R}$. Notably, we reduce query answering to the evaluation of alternative FOL queries, a.k.a. FOL *reformulations*, belonging to richer languages than those considered so far in the literature; in particular, this may allow *several* (equivalent) FOL reformulations for a given input query. This contrasts with related works, e.g., the aforementioned ones, which aim at a *single* FOL reformulation (modulo minimization). Allowing a variety of reformulations is crucial for efficiency, as such alternatives, while computing the same answers, may have very different performance (response time) when evaluated through an RDBMS. Therefore, instead of having a single fixed choice that may or may not be performant, we select the one with lowest estimated evaluation cost among possible alternatives.

2 Cover-based query answering optimization

RDBMS query optimizers consider a set of *evaluation alternatives* (a.k.a. logical and physical plans), and select the one minimizing a *cost estimation function*.

Since the number of alternatives is in $\mathcal{O}(2^n \times n!)$ for a conjunctive query (CQ) of n atoms [4], modern optimizers rely on heuristics to explore only a few alternatives; this works (very) well for *small-to-moderate size* CQs. However, FOL reformulations go *beyond* CQs in general, and may be *extremely large*, leading the RDBMS to perform poorly.

To work around this limitation, we introduce the cover-based query answering technique to define a space of equivalent FOL reformulations of a CQ. A *cover* defines how the query is split into subqueries, that may overlap, called *fragment queries*, such that substituting each subquery with its FOL reformulation (obtained from any state-of-the-art technique) and joining the corresponding (reformulated) subqueries, *may* yield a FOL reformulation for the query to answer. Not every cover of a query leads to a FOL reformulation; but every cover which does, yields an alternative *cover-based* FOL *reformulation* of the original query. Crucially for our problem, *a smart cover choice may lead to a cover-based reformulation whose evaluation is more efficient*. Thus, the cover-based technique amounts to *circumventing* the difficulty of modern RDBMSs to efficiently evaluate FOL reformulations in general.

Problem 1 (Optimization problem). Given a CQ q and a description logic KB $\mathcal{K}$, the *cost-driven cover-based query answering problem* consists of finding a cover-based reformulation of q based on $\mathcal{K}$ with lowest (estimated) evaluation cost.

We solve this problem for DL-Lite$_\mathcal{R}$ in two steps. First, we provide a sufficient condition for a cover to be *safe for query answering*, i.e., to lead to a cover-based FOL reformulation. The main idea for this condition is to have a cautious approximation of the *query atoms* which are interdependent w.r.t. reformulation, i.e., which (directly or after specialization) unify through state-of-the-art reformulation techniques, and keep them in the same cover fragment. The space of all covers of a query q satisfying this condition is denoted $\mathcal{L}_q$; all $\mathcal{L}_q$ covers turn out to correspond to some fusion of fragments from a certain *root cover* we denote C_{root}. We also refine our sufficient condition to identify an extended space of covers $\mathcal{E}_q$, which includes $\mathcal{L}_q$ and also leads to FOL reformulations of q.

Second, based on a function ϵ *estimating the evaluation cost of a given* FOL *query through an RDBMS*, we devise two cover search algorithms. The first one, termed **EC-DL (Exhaustive Covers)**, starts from C_{root} and explores all $\mathcal{E}_q$ covers in the case of DL-Lite$_\mathcal{R}$. The second one, named **GC-DL (Greedy Covers)**, also starts from C_{root} but explores $\mathcal{E}_q$ partially, in greedy fashion. It uses an *explored cover set* initialized with $\{C_{\text{root}}\}$, from which it picks a cover C inducing a q^{FOL} reformulation with minimum cost $\epsilon(C)$, and attempts to build from C a cover C', by fusing two fragments, or adding (copying) an atom to a fragment. GC-DL only adds C' to the explored set if $\epsilon(C') < \epsilon(C)$, thus it only explores a small part of the search space. Both algorithms return a cover-based reformulation with the minimum estimated cost w.r.t. the explored space. When fusing two fragments into one, or adding an atom to a fragment, $\epsilon(C)$ decreases if the new fragment is more selective than the fragment(s) it replaces. Therefore, the RDBMS may find a more efficient way to evaluate the query of this new fragment, and/or its result may be smaller, making the evaluation of q^{FOL} based on the new cover C' faster.

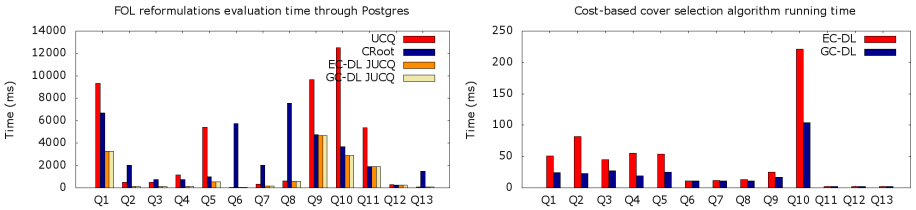


Fig. 1: (a) Evaluation time for FOL reformulations. (b) Cover search running time.

3 Experimental validation

We implemented our cover-based approach in Java 7, on top of PostgreSQL v9.3.2. We used the LUBM$^{\exists}_{20}$ DL-Lite$_{\mathcal{R}}$ TBox and associated EUDG data generator [3]: LUBM$^{\exists}_{20}$ consists of 34 roles, 128 concepts and 212 constraints; the generated ABox comprises 15 million facts. We chose RAPID [2] for CQ-to-UCQ (unions of CQs) reformulation. For ϵ, we used Postgres' own estimation, obtained using the `explain` directive. We devised a set of 13 CQs, ranging from 2 to 10 atoms (5.77 on average); their UCQ reformulations have 35 to 667 CQs (290.2 on average).

Figure 1(a) depicts the evaluation time through Postgres, of four FOL reformulations: (*i*) the UCQ produced by RAPID [2]; (*ii*) the JUCQ (joins of UCQs) reformulation based on C_{root}; (*iii*) the JUCQ reformulation corresponding to the best-performing cover found by our algorithm EC-DL, and (*iv*) the JUCQ reformulation based on the best-performing cover found by GC-DL. First, the figure shows that fixed FOL reformulations are not efficiently evaluated, e.g., UCQ for Q_1, Q_5 and Q_9-Q_{11}, and the one based on C_{root} for Q_6-Q_8 and Q_{13}. This poor performance correlates with the large size of the UCQ reformulations: such very large unions of CQs are very poorly handled by current RDBMS optimizers, which are designed and tuned for small CQs. Second, the reformulation based on the cover returned by EC-DL is always more efficient than UCQ reformulation (more than one order of magnitude for Q_5), respectively, C_{root}-based reformulation (up to a factor of 230 for Q_6). Third, in our experiments, the GC-DL-chosen cover leads to a JUCQ reformulation as efficient as the EC-DL one, demonstrating that even a partial, greedy cover search leads to good performance (this cannot be guaranteed in general). For Q_7 and Q_9-Q_{13}, the best cover we found is safe; for all the others, this is not the case, confirming the interest of the larger space $\mathcal{E}_q$.

Figure 1(b) depicts the running time of the EC-DL and GC-DL algorithms, which can be seen as the overhead of our cover-based technique. The time is very small, between 2 ms (Q_{11}-Q_{13}, with just 2 atoms) and 221 ms (EC-DL on Q_{10}, of 10 atoms). The time is higher for more complex queries, but these are precisely the cases where our techniques are most benefficial, e.g., for Q_{10}, EC-DL runs in less than 2% of the time to evaluate the UCQ reformulation, while the cover we recommend is more than 4 times faster than UCQ. As expected, GC-DL is faster than EC-DL due to the exploration of less covers. Together, Figure 1(a) and 1(b) confirm the benefits and practical interest of our cost-based cover search.

Acknowledgements This work has been partially funded by the *Programme Investissement d'Avenir* Datalyse project and the ANR PAGODA project.

References

1. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. JAR 39(3), 385–429 (2007)
2. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: CADE (2011)
3. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: taming role hierarchies using filters. In: ISWC. pp. 314–330 (2013)
4. Ono, K., Lohman, G.M.: Measuring the complexity of join enumeration in query optimization. In: VLDB (1990)
5. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: ISWC (2009)
6. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR (2010)
7. Venetis, T., Stoilos, G., Stamou, G.B.: Incremental query rewriting for OWL 2 QL. In: Description Logics (2012)

Decidable Contextualized DLs with Rigid Roles

Stephan Böhme** and Marcel Lippmann

Institute for Theoretical Computer Science, Technische Universität Dresden,
{stephan.boehme,marcel.lippmann}@tu-dresden.de

Description logics (DLs) of context can be employed to represent and reason about contextualized knowledge, which naturally occurs in practice [5,4,7,9,8]. Consider, for instance, the rôles played by a person in different contexts. The person Bob, who works for the company Siemens, plays the rôle of an employee of Siemens in the work context, whereas he might play the rôle of a customer of Siemens in the context of private life. Here, access restrictions to the data of Siemens might critically depend on Bob's rôle. Moreover, DLs capable of representing contexts are vital to integrate distributed knowledge as argued in [5,4].

DLs are well-suited to describe contexts as formal objects with formal properties that are organized in relational structures, which are fundamental requirements for modeling contexts [11,12]. However, classical DLs lack expressive power to formalize that some individuals satisfy certain concepts and relate to other individuals depending on a specific context. Therefore, often two-dimensional DLs are employed [7,9,8]: One DL $\mathcal{L}_M$ (the *meta* or *outer logic*) is used to represent the contexts and their relationships to each other. $\mathcal{L}_M$ is combined with a DL $\mathcal{L}_O$ (the *object* or *inner logic*) that captures the relational structure within each context. Moreover, while some pieces of information depend on the context, other pieces of information are shared throughout all contexts. For instance, a person's name is typically independent of the actual context. To be able to express that, some concepts and roles a designated to be *rigid*, i.e. they are required to be interpreted the same in all contexts. Unfortunately, if rigid roles are admitted, reasoning in contextualized DLs is usually undecidable [7].

We propose and investigate a family of two-dimensional DLs $\mathcal{L}_M[\![\mathcal{L}_O]\!]$ that meet the above requirements, but are a restricted form of the one defined in [7] in the sense that we limit the interaction of $\mathcal{L}_M$ and $\mathcal{L}_O$. More precisely, in our family of contextualized DLs the outer DL can refer to the internal structure of each context, but not vice versa. This represents contexts in a top-down perspective. Interestingly, reasoning in $\mathcal{L}_M[\![\mathcal{L}_O]\!]$ stays decidable with such a restriction, even in the presence of rigid roles. In some sense our family of contextualized DLs are similar to temporalized DLs investigated in [2,3,10].

For providing better intuition on how our formalism works, we examine the above mentioned example a bit further. Consider the following axioms:

$$\top \sqsubseteq [\![\exists\, worksFor.\{Siemens\} \sqsubseteq \exists\, hasAccessRights.\{Siemens\}]\!] \quad (1)$$

$$\textsc{Work} \sqsubseteq [\![worksFor(Bob, Siemens)]\!] \quad (2)$$

$$\top \sqsubseteq [\![\exists\, isCustomerOf.\top \sqsubseteq HasMoney]\!] \quad (3)$$

** Funded by DFG in the Research Training Group "RoSI" (GRK 1907).

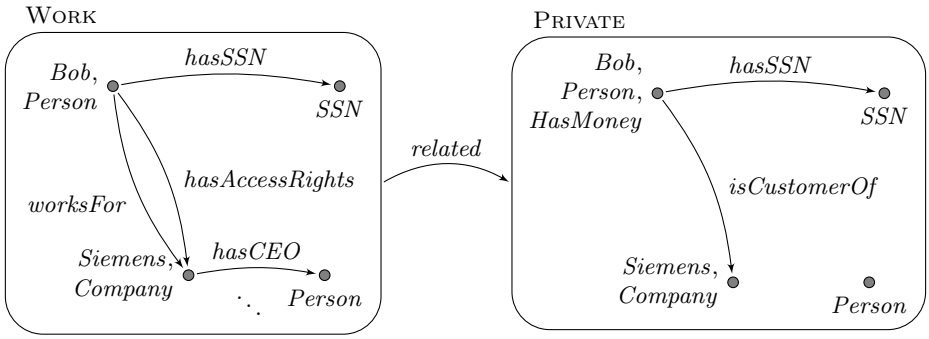


Fig. 1. Model of Axioms 1–7

$$[\![(\exists\, worksFor.\top)(Bob)]\!] \sqsubseteq \exists\, related.(\textsc{Private} \sqcap [\![HasMoney(Bob)]\!]) \quad (4)$$

$$\textsc{Private} \sqsubseteq [\![isCustomerOf(Bob, Siemens)]\!] \quad (5)$$

$$\textsc{Private} \sqcap \textsc{Work} \sqsubseteq \bot \quad (6)$$

$$\neg\textsc{Work} \sqsubseteq [\![\exists\, worksFor.\top \sqsubseteq \bot]\!] \quad (7)$$

Axiom 1 states that it holds true in all contexts that somebody who works for Siemens also has access rights to certain data. Axiom 2 states that Bob is an employee of Siemens in any work context. Axioms 3 and 4 say intuitively that if Bob has a job, he will earn money, which he can spend as a customer. Axiom 5 formalizes that Bob is a customer of Siemens in any private context. Moreover, Axiom 6 ensures that private and work contexts are disjoint. Finally, Axiom 7 states that the *worksFor* relation only exists in work contexts. A fundamental reasoning task is to decide whether a set of axioms is *consistent*. For our example, Figure 1 depicts a model. There, Bob's social security number is linked to him using a rigid role *hasSSN* since it does not change over the contexts.

Our family $\mathcal{L}_M[\![\mathcal{L}_O]\!]$ consists of combinations of two DLs, where we focus on the cases where $\mathcal{L}_M$ and $\mathcal{L}_O$ are $\mathcal{EL}$ or DLs between $\mathcal{ALC}$ and $\mathcal{SHOQ}$. Let $\mathsf{O_C}$, $\mathsf{O_R}$, and $\mathsf{O_I}$ be respectively sets of concept, role, and individual names for the object logic $\mathcal{L}_O$. Analogously, we define the sets $\mathsf{M_C}$, $\mathsf{M_R}$, and $\mathsf{M_I}$ for the meta logic $\mathcal{L}_M$. Let $\mathsf{O} = (\mathsf{O_C}, \mathsf{O_R}, \mathsf{O_I})$ and $\mathsf{M} = (\mathsf{M_C}, \mathsf{M_R}, \mathsf{M_I})$. Concepts, GCIs and assertions are defined over the respective signatures O and M as usual [1].

Definition 1. Concepts of the object logic (o-concepts) *are $\mathcal{L}_O$-concepts over O;* o-axioms *are $\mathcal{L}_O$-axioms over O.* Concepts of the meta logic (m-concepts) *are defined inductively: each $\mathcal{L}_M$-concept over M is an m-concept, and $[\![\alpha]\!]$ is an m-concept for an o-axiom α; and* m-axioms *are defined analogously.* Boolean $\mathcal{L}_M[\![\mathcal{L}_O]\!]$-knowledge bases ($\mathcal{L}_M[\![\mathcal{L}_O]\!]$-BKBs) *are Boolean combinations of m-axioms.*

Note that the syntax of the object level is precisely the one of $\mathcal{L}_O$, whereas the syntax of the meta level also allows to put $\mathcal{L}_O$-axioms in place of concept names. The semantics of $\mathcal{L}_M[\![\mathcal{L}_O]\!]$ is defined using *nested interpretations*, which consist of O-interpretations (usual DL interpretations for the names in O) for the

Table 1. Complexity results for consistency in $\mathcal{L}_M[\![\mathcal{L}_O]\!]$

$\mathcal{L}_M$ \ $\mathcal{L}_O$	no rigid names			only rigid concepts			rigid roles		
	$\mathcal{EL}$	$\mathcal{ALC}$	$\mathcal{SHOQ}$	$\mathcal{EL}$	$\mathcal{ALC}$	$\mathcal{SHOQ}$	$\mathcal{EL}$	$\mathcal{ALC}$	$\mathcal{SHOQ}$
$\mathcal{EL}$	const	ExP	ExP	const	NExP	NExP	const	2ExP	2ExP
$\mathcal{ALC}$	ExP	ExP	ExP	NExP	NExP	NExP	NExP	2ExP	2ExP
$\mathcal{SHOQ}$	ExP	ExP	ExP	NExP	NExP	NExP	NExP	2ExP	2ExP

specific contexts and the relational structure between them (M-interpretation), where all contexts have the same domain. Also, let $\mathsf{O}_{\mathsf{Crig}} \subseteq \mathsf{O}_{\mathsf{C}}$ denote the set of *rigid concepts*, and let $\mathsf{O}_{\mathsf{Rrig}} \subseteq \mathsf{O}_{\mathsf{R}}$ denote the set of *rigid roles*. Moreover, we assume that individuals of $\mathcal{L}_O$ are always interpreted the same in all contexts.

Definition 2. *We call a tuple* $\mathcal{J} = (\mathbb{C}, \cdot^{\mathcal{J}}, \Delta, (\cdot^{\mathcal{I}_c})_{c\in\mathbb{C}})$ *a* nested interpretation*, where* $\mathbb{C}$ *is a non-empty set (called* contexts*) and* $(\mathbb{C}, \cdot^{\mathcal{J}})$ *is an* M*-interpretation. Moreover,* $\mathcal{I}_c := (\Delta, \cdot^{\mathcal{I}_c})$ *is an* O*-interpretation for every* $c \in \mathbb{C}$*, such that it holds for all* $c, c' \in \mathbb{C}$ *that* $x^{\mathcal{I}_c} = x^{\mathcal{I}_{c'}}$ *for all* $x \in \mathsf{O}_{\mathsf{I}} \cup \mathsf{O}_{\mathsf{Crig}} \cup \mathsf{O}_{\mathsf{Rrig}}$*.*

Definition 3. *Let* $\mathcal{J} = (\mathbb{C}, \cdot^{\mathcal{J}}, \Delta, (\cdot^{\mathcal{I}_c})_{c\in\mathbb{C}})$ *be a nested interpretation. The mapping* $\cdot^{\mathcal{J}}$ *is extended as follows:* $[\![\alpha]\!]^{\mathcal{J}} := \{c \in \mathbb{C} \mid \mathcal{I}_c \models \alpha\}$*. Moreover,* $\mathcal{J}$ *is a* model *of the m-axiom* β *if* $(\mathbb{C}, \cdot^{\mathcal{J}})$ *is a model of* β*. This is extended to* $\mathcal{L}_M[\![\mathcal{L}_O]\!]$*-BKBs inductively as usual. We write* $\mathcal{J} \models \mathcal{B}$ *if* $\mathcal{J}$ *is a model of the* $\mathcal{L}_M[\![\mathcal{L}_O]\!]$*-BKB* $\mathcal{B}$*. We call* $\mathcal{B}$ consistent *if it has a model.*

The complexity of consistency in $\mathcal{L}_M[\![\mathcal{L}_O]\!]$ is listed in Table 1. The lower bounds are obtained using the ideas of [2,3] and hold already for the fragment $\mathcal{EL}[\![\mathcal{ALC}]\!]$, even if only *conjunctions* of m-axioms are considered instead of BKBs. Without rigid names, ExP-hardness follows from the complexity of $\mathcal{L}_O$. If rigid concept and role names are allowed, we reduce the word problem for exponentially space-bounded alternating Turing machines to obtain 2ExP-hardness. If only rigid concept names are allowed, a reduction of an exponentially bounded version of the domino problem yields NExP-hardness. For the upper bounds, we proceed similar to what was done for $\mathcal{ALC}$-LTL in [2,3] and reduce the consistency problem to two separate decision problems. First, we abstract our $\mathcal{L}_M[\![\mathcal{L}_O]\!]$-BKB $\mathcal{B}$ by replacing the m-concepts that consist of o-axioms by fresh concept names and test this abstraction $\mathcal{B}'$ for consistency. With this abstraction, we loose, however, the information on the o-axioms. In each model of $\mathcal{B}'$, the fresh concept names have some extensions, and are treated completely independently. The induced o-axioms, however, may not be independent. It could be that some o-axioms are inconsistent together. We check this in a separate step.

To conclude, we have proposed novel combinations of two DLs for representing contextual knowledge and analyzed their complexity. Interestingly, even in the presence of rigid roles the consistency problem is still decidable. For more details, see [6]. As future work apart from others, we envision that our decision procedures can be adapted to deal with temporalized context DLs such as $LTL[\![\mathcal{ALC}[\![\mathcal{ALC}]\!]]\!]$.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2007)
2. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. In: Brewka, G., Lang, J. (eds.) Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008). pp. 684–694. AAAI Press, Sydney, Australia (2008)
3. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. ACM Transactions on Computational Logic 13(3) (2012)
4. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-based description logics. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, Lecture Notes in Computer Science, vol. 5445, pp. 349–371. Springer (2009)
5. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics 2800, 153–184 (2003)
6. Böhme, S., Lippmann, M.: Description logics of context with rigid roles revisited. LTCS-Report 15-04, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden (2015), see `http://lat.inf.tu-dresden.de/research/reports.html`.
7. Klarman, S., Gutiérrez-Basulto, V.: $\mathcal{ALC}_{\mathcal{ALC}}$: A context description logic. In: Janhunen, T., Niemelä, I. (eds.) Proceedings of the 12th European Conference on Logics in Artificial Intelligence (JELIA 2010). Lecture Notes in Computer Science, vol. 6341, pp. 208–220. Springer (2010)
8. Klarman, S., Gutiérrez-Basulto, V.: Two-dimensional description logics for context-based semantic interoperability. In: Burgard, W., Roth, D. (eds.) Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI 2011). AAAI Press (2011)
9. Klarman, S., Gutiérrez-Basulto, V.: Two-dimensional description logics of context. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Proceedings of the 24th International Workshop on Description Logics (DL 2011). vol. 745. CEUR-WS.org (2011)
10. Lippmann, M.: Temporalised Description Logics for Monitoring Partially Observable Events. Ph.D. thesis, TU Dresden, Germany (2014)
11. McCarthy, J.: Generality in artificial intelligence. Communications of the ACM 30(12), 1030–1035 (1987)
12. McCarthy, J.: Notes on formalizing context. In: Bajcsy, R. (ed.) Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993). pp. 555–562. Morgan Kaufmann, Los Altos, Chambéry, France (1993)

Inconsistency Management in Generalized Knowledge and Action Bases*.

Diego Calvanese, Marco Montali, and Ario Santoso

Free University of Bozen-Bolzano, Bolzano, Italy
lastname@inf.unibz.it

1 Introduction

The combination of static and dynamic aspects in modeling complex organizational domains is a challenging task that has led to study the combination of formalisms from knowledge representation, database theory, and process management [18,23,11]. Specifically, *Knowledge and Action Bases* (KABs) [3] have been put forward recently to provide a semantically rich representation of a domain. In KABs, static aspects are modeled using a Description Logic (DL) [1] knowledge base (KB), while actions are used to evolve its extensional part over time, possibly introducing fresh individuals. An important aspect that has received little attention so far in such systems is the management of *inconsistency* with respect to domain knowledge that may arise when the extensional information is evolved over time. In fact, inconsistency, both in KABs and in related approaches, is typically handled naively by just rejecting updates in actions when they would lead to inconsistency, see e.g., [16,4,9,2].

To overcome this limitation, KABs have been extended lately with mechanisms to handle inconsistency [12]. However, this has been done by defining ad-hoc execution semantics and corresponding ad-hoc verification techniques geared towards specific semantics for inconsistency management. It has also been left open whether adding inconsistency management to the rich setting of KABs, actually increases expressive power. This work attacks these issues by: *(i)* Proposing (standard) GKABs, which enrich KABs with a compact action language inspired by Golog [20] that can be conveniently used to specify processes at a high-level of abstraction. As in KABs, standard GKABs still manage inconsistency naively. *(ii)* Defining a parametric execution semantic for GKABs that is able to elegantly accomodate a plethora of inconsistency-aware semantics based on the well-known notion of repair [17,5,19,13]. *(iii)* Providing several reductions showing that verification of sophisticated first-order temporal properties over inconsistency-aware GKABs can be recast as a corresponding verification problem over standard GKABs. *(iv)* Showing that verification of standard and inconsistency-aware GKABs can be addressed using known techniques, developed for standard KABs.

2 Setting

We use *DL-Lite*$_{\mathcal{A}}$ [8,6] to express KBs, and consider queries such as EQL-Lite(UCQ) (briefly ECQs) [7], to access KBs and extract individuals of interest. To handle inconsis-

* This paper is an abridged version of [14]. Full proofs can be found in [15]

tency in KBs, we follow the *repair-based approaches* in [12], and distinguish between two kinds of approaches: *(i)* those that compute repairs agnostically from the updates (the added/deleted facts) [19,12], among which we have *b-repairs*, which are defined as the maximal (w.r.t. set containment) subsets of an ABox that are consistent with the TBox, and *c-repairs*, which are defined as the intersection of all b-repairs; *(ii)* those that take into account the updates by giving higher priority to the new facts during the repair, as in *bold semantics* for instance-level KB evolution (c.f., [13]).

Here, we consider KABs that are obtained by combining the framework in [3,12] with the action specification formalism in [22], which allows us to have actions that only update an ABox (instead of creating a new ABox at each execution, as in [3,12]). Formally, a KAB is composed by *(i)* a *DL-Lite*$_\mathcal{A}$ TBox T; *(ii)* an initial *DL-Lite*$_\mathcal{A}$ ABox A_0; *(iii)* a finite set Γ of parametric actions that evolve the ABox; *(iv)* a finite set Π of condition-action rules that describe when actions can be executed, and with which parameters. The execution semantics of a KAB is given in terms of a possibly infinite-state *transition system*, whose construction depends on the adopted semantics of inconsistency [12]. As in [12], we call S-KAB a KAB under the standard execution semantics, where inconsistency is naively managed by simply rejecting those updates that lead to an inconsistent state.

To specify temporal properties over KABs, we use the $\mu\mathcal{L}_A^{\text{EQL}}$ logic, the FO variant of μ-calculus defined in [3]. Given a transition system Υ and a closed $\mu\mathcal{L}_A^{\text{EQL}}$ formula Φ, *verification* is the problem of checking whether Φ holds in the initial state of Υ.

3 Golog-KABs and Inconsistency Management

We enrich KABs with a high-level action language inspired by Golog [20]. This allows modelers to represents the dynamics of systems much more compactly. On the other hand, we introduce a parametric execution semantics, which elegantly accommodates the different kinds of inconsistency-aware semantics based on the notion of repair.

A *Golog-KAB (GKAB)* is a tuple $\mathcal{G} = \langle T, A_0, \Gamma, \delta \rangle$, where T, A_0, and Γ are as in standard KABs, and δ is the Golog program characterizing the evolution of the GKAB over time, using the atomic actions in Γ. For simplicity, we only consider a core fragment of Golog based on the action language in [10], and define a *Golog program* as:

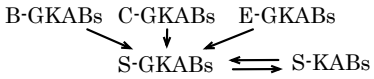
$$\delta ::= \varepsilon \mid \textbf{pick}\ Q(\vec{p}).\alpha(\vec{p}) \mid \delta_1|\delta_2 \mid \delta_1;\delta_2 \mid \textbf{if}\ \varphi\ \textbf{then}\ \delta_1\ \textbf{else}\ \delta_2 \mid \textbf{while}\ \varphi\ \textbf{do}\ \delta$$

where: *(i)* ε is the *empty program*; *(ii)* **pick** $Q(\vec{p}).\alpha(\vec{p})$ is an *atomic action invocation* guarded by an ECQ Q, such that $\alpha \in \Gamma$ is applied by non-deterministically substituting its parameters $\vec{p}$ with an answer of Q; *(iii)* $\delta_1|\delta_2$ is a *non-deterministic choice* between programs; *(iv)* $\delta_1;\delta_2$ is *sequencing*; *(v)* **if** φ **then** δ_1 **else** δ_2, and **while** φ **do** δ are respectively *conditional* and *loop* constructs, using a boolean ECQ φ as condition.

We adopt the functional approach by [21] in defining the semantics of action execution over $\mathcal{G}$, i.e., we assume $\mathcal{G}$ provides two operations: *(i)* ASK, to answer queries over the current KB; *(ii)* TELL, to update the KB through an atomic action. Here the ASK operator corresponds to certain answers computation. The TELL operation is parameterized by *filter relations* f, which are used to refine the way in which an ABox is updated, based on a set of facts to be added and deleted (specified by the action), and we require that the result of the TELL operation is a T-consistent ABox. In this light, filter relations

provide an abstract mechanism to accommodate several inconsistency management approaches in the execution semantics. For instance, we define GKABs with standard execution semantics, briefly S-GKABs, by defining a filter relation f_S that updates an ABox based on the facts to be added and deleted, and does nothing w.r.t. inconsistency (i.e., updates that lead to an inconsistent state are simply rejected). To obtain GKABs with inconsistency-aware semantics, we introduce filter relations f_B, f_C, and f_E, where f_B (resp., f_C) returns a b-repair (resp., c-repair) [12] of the updated ABox, and f_E updates the ABox using the bold semantics of KB evolution [13]. Consecutively, we call the GKABs adopting the execution semantics obtained by employing those filter relations *B-GKABs*, *C-GKABs*, and *E-GKABs*, respectively, and we group them under the umbrella of *inconsistency-aware GKABs (I-GKABs)*.

Verification Results. With respect to verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties, we have proved the results summarized below, where an arrow indicates that we can reduce verification in (G)KABs in the source to verification in (G)KABs in the target:



To encode S-KABs into S-GKABs, we simulate the standard execution semantics using a Golog program that continues forever to non-deterministically pick an executable action with parameters, or stops if no action is executable. For the opposite direction, the key idea is to inductively interpret a Golog program as a structure consisting of nested processes, suitably composed through the Golog operators. We mark the starting and ending point of each Golog subprogram, and use accessory facts in the ABox to track states corresponding to subprograms. Each subprogram is then inductively translated into a set of actions and condition-action rules encoding its entrance and termination conditions. For all reductions from I-GKABs to S-GKABs, our general strategy is to show that S-GKABs are sufficiently expressive to incorporate the repair-based approaches, so that an action executed under a certain inconsistency semantics can be compiled into a Golog program that applies the action with the standard semantics, and then explicitly handles the inconsistency, if needed.

It is also interesting to observe that the semantic property of *run-boundedness* (which guarantees the decidability of S-KAB verification) [2,3] is preserved by all our reductions. It follows that verification of $\mu\mathcal{L}_A^{\text{EQL}}$ properties over run-bounded GKABs and I-GKABs is decidable, and reducible to standard μ-calculus finite-state model checking.

4 Conclusion

We introduced GKABs, which extend KABs with Golog-inspired high-level programs, and allow for an elegant treatment of inconsistency. We have also shown that verification of rich temporal properties over (inconsistency-aware) GKABs can be recast as verification over standard KABs. Our approach is very general, and can be easily extended to account for other inconsistency handling mechanisms, and more in general data cleaning.

Acknowledgments. This research has been partially supported by the EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338, and by the UNIBZ internal project KENDO (*Knowledge-driven ENterprise Distributed cOmputing*).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
2. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., Deutsch, A., Montali, M.: Verification of relational data-centric dynamic systems with external services. In: Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS). pp. 163–174 (2013)
3. Bagheri Hariri, B., Calvanese, D., Montali, M., De Giacomo, G., De Masellis, R., Felli, P.: Description logic Knowledge and Action Bases. J. of Artificial Intelligence Research 46, 651–686 (2013)
4. Belardinelli, F., Lomuscio, A., Patrizi, F.: An abstraction technique for the verification of artifact-centric systems. In: Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR). pp. 319–328 (2012)
5. Bertossi, L.E.: Consistent query answering in databases. SIGMOD Record 35(2), 68–76 (2006)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The *DL-Lite* approach. In: Tessaris, S., Franconi, E. (eds.) Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW), Lecture Notes in Computer Science, vol. 5689, pp. 255–356. Springer (2009)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI). pp. 274–279 (2007)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)
9. Calvanese, D., De Giacomo, G., Lembo, D., Montali, M., Santoso, A.: Ontology-based governance of data-aware processes. In: Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR). Lecture Notes in Computer Science, vol. 7497, pp. 25–41. Springer (2012)
10. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and programs over description logic knowledge bases: A functional approach. In: Lakemeyer, G., McIlraith, S.A. (eds.) Knowing, Reasoning, and Acting: Essays in Honour of Hector Levesque. College Publications (2011)
11. Calvanese, D., De Giacomo, G., Montali, M.: Foundations of data aware process analysis: A database theory perspective. In: Proc. of the 32nd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS) (2013)
12. Calvanese, D., Kharlamov, E., Montali, M., Santoso, A., Zheleznyakov, D.: Verification of inconsistency-tolerant knowledge and action bases. In: Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI) (2013)
13. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL-Lite* knowledge bases. In: Proc. of the 9th Int. Semantic Web Conf. (ISWC). Lecture Notes in Computer Science, vol. 6496, pp. 112–128. Springer (2010)
14. Calvanese, D., Montali, M., Santoso, A.: Verification of generalized inconsistency-aware knowledge and action bases. In: Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI) (2015)
15. Calvanese, D., Montali, M., Santoso, A.: Verification of generalized inconsistency-aware knowledge and action bases (extended version). CoRR Technical Report arXiv:1504.08108, arXiv.org e-Print archive (2015), available at `http://arxiv.org/abs/1504.08108`

16. Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of data-centric business processes. In: Proc. of the 12th Int. Conf. on Database Theory (ICDT). pp. 252–267 (2009)
17. Eiter, T., Gottlob, G.: On the complexity of propositional knowledge base revision, updates and counterfactuals. Artificial Intelligence 57, 227–270 (1992)
18. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE). Lecture Notes in Computer Science, vol. 5332, pp. 1152–1163. Springer (2008)
19. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR). pp. 103–117 (2010)
20. Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.: GOLOG: A logic programming language for dynamic domains. J. of Logic Programming 31, 59–84 (1997)
21. Levesque, H.J.: Foundations of a functional approach to knowledge representation. Artificial Intelligence 23, 155–212 (1984)
22. Montali, M., Calvanese, D., De Giacomo, G.: Verification of data-aware commitment-based multiagent systems. In: Proc. of the 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS). pp. 157–164 (2014)
23. Vianu, V.: Automatic verification of database-driven systems: a new frontier. In: Proc. of the 12th Int. Conf. on Database Theory (ICDT). pp. 1–13 (2009)

Tableau-based revision in $\mathcal{SHIQ}$

Thinh Dong, Chan Le Duc, Philippe Bonnot, Myriam Lamolle

LIASD - EA4383, IUT of Montreuil, University of Paris8, France
{dong, leduc, bonnot, lamolle}@iut.univ-paris8.fr

Introduction The problem of revising a description logic-based ontology (called DL ontology) is closely related to the problem of belief revision which has been widely discussed in the literature. Among early works on belief revision, the AGM theory (Alchourrón *et al.*, 1985) introduced intuitive and plausible constraints (namely AGM postulates) which should be satisfied by any rational belief revision operator. However, it is not trivial to adapt belief revision operators to DLs because DLs have their own features (Flouris *et al.*, 2005) (Qi and Yang, 2008). One main difficult for such revision is that DL ontologies often incur infinitely many models. To address this issue, we propose a finite set of finite structures, namely a set $\mathsf{MT}(\mathcal{O})$ of completion trees, for characterizing a possibly infinite set of models of an ontology $\mathcal{O}$. Then, we define a distance over a set of completion trees. This distance allows one to determine how far an ontology is from another one. Another problem our approach has to address is that there may not exist a revision ontology such that (i) it is expressible in the logic used for expressing initial ontologies $\mathcal{O}, \mathcal{O}'$, and (ii) it admits *exactly* a set of models $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ computed from $\mathsf{MT}(\mathcal{O})$ and $\mathsf{MT}(\mathcal{O}')$. For this reason, we borrow the notion of *maximal approximation* (De Giacomo *et al.*, 2007) which allows us to build a *minimal* revision ontology admitting $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

Construction of the revision ontology First, we define a novel tableau algorithm, namely $\mathbb{TA}$, for a $\mathcal{SHIQ}$ ontology without individuals by replacing expansion $\sqsubseteq$-, $\sqcap$-, $\sqcup$, ch-rules by a new rule, namely sat-rule which chooses a subset $\mathcal{S}$ from a set $\mathsf{sub}(\mathcal{O})$ including all sub-concepts of a $\mathcal{SHIQ}$ ontology $\mathcal{O}$. Note that all concepts in the form of conjunctions or of disjunctions are removed from $\mathsf{sub}(\mathcal{O})$ and replaced with their conjuncts and disjuncts. This can be performed by a function $\mathsf{Flat}(C)$ that flattens conjunctions and disjunctions of a concept C into subsets of sub-concepts occurring in C. For example, $\mathsf{Flat}(A \sqcap (\exists R.B \sqcup C)) = \{\{A, \exists R.B\}, \{A, C\}\}$.

sat**-rule.** If sat-rule has never been applied to a node x then we choose a subset $S \subseteq \mathsf{sub}(\mathcal{O})$ such that $L(x) \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}} f(C \sqsubseteq D) \subseteq S$ where $f(C \sqsubseteq D) \in \mathsf{Flat}(\neg C \sqcup D)$ for each $C \sqsubseteq D \in \mathcal{T}$, and set $L(x) := S \cup \bar{S}$ where $\bar{S} = \{\neg C \mid C \in \mathsf{sub}(\mathcal{O}) \setminus S\}$

In this paper, a *completion tree* for $\mathcal{O}$ is a tree $T = (V, E, L, \widehat{x})$ where V is a set of nodes with the root node $\widehat{x} \in V$. Each node $x \in V$ is labeled with a function $L(x) \subseteq \mathsf{sub}(\mathcal{O})$. E is a set of edges and each edge $\langle x, y \rangle \in E$ is labeled with a function $L(\langle x, y \rangle)$ containing a set of $\mathcal{SHIQ}$ roles.

The sat-rule that is applied to each node of a completion tree introduces a lot of non-determinisms. We need this "bad" behavior of the new tableau algorithm to control the generation process of completion trees in such a way that allows one to infer the ontology when knowing completion trees and its signature. We use $\mathsf{MT}(\mathcal{O})$ to denote the set of all completion trees which are generated by running the novel tableau algorithm $\mathbb{TA}$ for an ontology $\mathcal{O}$. Note that $\mathbb{TA}$ does not necessarily terminate when a complete and

clash-free completion tree is built. It should terminate when all non-determinisms are considered. We can extend straightforwardly $\mathsf{MT}(\mathcal{O})$ to $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ as follows. The set $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ is built by the tableau algorithm $\mathbb{TA}$ for $\mathcal{O}'$ with an extra set of concepts $\mathsf{sub}(\mathcal{O})$ that is taken into account when applying the sat-rule. In this case one can import additional concepts into node labels of a completion tree for $\mathcal{O}'$ while respecting the axioms of $\mathcal{O}'$. Importing $\mathsf{sub}(\mathcal{O})$ to $\mathsf{MT}(\mathcal{O}')$ ensures that $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ captures semantic constraints from $\mathcal{O}$ which are compatible with $\mathcal{O}'$.

Next, we introduce a distance between two completion trees T and T' which allows one to talk about the similarity between two ontologies. This distance is defined for two completion trees which are isomorphic, i.e., there is an isomorphism π that maintains the successor relationship from two nodes of a completion tree to the two corresponding nodes of the other one via π. Note that we can always obtain such an isomorphism between two completion trees by adding empty nodes and edges to completion trees since node and edge labels are ignored in the definition of isomorphisms.

Definition 1 (Distance). *Let $T = \langle V, L, E, \widehat{x}\rangle$ and $T' = \langle V', L', E', \widehat{x}'\rangle$ two completion trees. Let $\Pi(T, T')$ be the set of all isomorphisms between T and T'. The distance between T and T', denoted $T \triangle T'$, is defined as follows: $T \triangle T' = \min_{\pi \in \Pi(T,T')} \{\max_{x \in V}(|L(x) \triangle L'(\pi(x))|) + \max_{\langle x,y\rangle \in E}(|L(\langle x, y\rangle) \triangle L'(\langle \pi(x), \pi(y)\rangle)|)\}$*

We can check that $\triangle$ is a distance over a set of isomorphic trees with the operator $\triangle$ defined over two node or edge labels α, α' as follows: $L(\alpha) \triangle L'(\alpha') = (L(\alpha) \cup L'(\alpha')) \setminus (L(\alpha) \cap L'(\alpha'))$. Based on this distance, we now define a set of completion trees a revision ontology of an ontology $\mathcal{O}$ by another $\mathcal{O}'$ should admit.

Definition 2 (Revision operation). *Let $\mathcal{O}$ and $\mathcal{O}'$ be two consistent $\mathcal{SHIQ}$ ontologies. A set of tree models $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ of the revision of $\mathcal{O}$ by $\mathcal{O}'$ is defined as follows:*

$$\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \{T \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \mid \exists T_0 \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')), \forall T' \in \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')), T'' \in \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) : T \triangle T_0 \leq T' \triangle T''\}$$

Intuitively, $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ includes completion trees from $\mathsf{MT}(\mathcal{O}')$ each node of which is consistently filled by an arbitrary set of concepts imported from $\mathsf{sub}(\mathcal{O})$ such that each axiom of $\mathcal{O}'$ remains satisfied. Among these completion trees, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ retains only those which are closest to completion trees from $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ thanks to the operator $T \triangle T'$ that characterizes the difference between T and T'. We consider the following example. Let $\mathcal{O} = \{\top \sqsubseteq A \sqcap \exists R.(\neg B) \sqcap \neg B\}$ and $\mathcal{O}' = \{\neg A \sqsubseteq \forall R.B, \neg B \sqsubseteq A \sqcap \forall R.B\}$. By running the algorithm $\mathbb{TA}$ for $\mathcal{O}$, we build the set $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ which contains a unique tree model T_1 with nodes $\{a, b\}$ and labels $L(a) = \{A, \exists R.(\neg B), \neg B\}, L(b) = \{A, \exists R.(\neg B), \neg B\}, E = \{R(a, b)\}$. In the same way, $\mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ has 4 tree models one of which is T_1' with nodes $\{a', b'\}$ and labels $L(a') = \{A, \exists R.(\neg B), B\}$, $L(b') = \{\neg B, A, \forall R.B\}, \{R(a', b')\}$. According to Definition 2, we have $T_1' \triangle T_1 = 2$ that is minimal. Thus, $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ contains a unique tree model T_1'.

We obtain a strong result which states that the all AGM postulates rephrased (Qi *et al.*, 2006) for DL ontologies in our setting hold. This result relies on a total pre-order over a set of all completion trees that can be devised from the distance according to

Definition 1. The main difference between the postulates presented by Qi et al. and those reformulated in our setting is that the set of models $\mathsf{Mod}(\mathcal{O})$ of an ontology $\mathcal{O}$ is replaced with $\mathsf{MT}(\mathcal{O})$. To illustrate this point, we consider a postulate by Qi et al. **(G2)**: *If* $\mathsf{Mod}(\mathcal{O}) \cap \mathsf{Mod}(\mathcal{O}') \neq \emptyset$ *then* $\mathsf{Mod}(\mathcal{O}, \mathcal{O}') = \mathsf{Mod}(\mathcal{O}) \cap \mathsf{Mod}(\mathcal{O}')$; and our corresponding postulate: **(P2)** *If* $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$ *then* $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$. A proof of **(P2)** can be obtained straightforwardly from the definition of $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}'))$ and $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

By soundness and completeness of the tableau algorithm, we can show that $\mathsf{Mod}(\mathcal{O})$ is semantically equivalent to $\mathsf{MT}(\mathcal{O})$, i.e., $\mathsf{MT}(\mathcal{O}) \models \alpha$ iff $\mathsf{Mod}(\mathcal{O}) \models \alpha$ for some axiom α. Moreover, it holds that $\mathsf{Mod}(\mathcal{O}) \cap \mathsf{Mod}(\mathcal{O}') \neq \emptyset$ iff $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O})) \neq \emptyset$. Therefore, as **(G2)** our postulate **(P2)** captures the fact that if $\mathcal{O} \cup \mathcal{O}'$ is consistent, then the revision ontology of $\mathcal{O}$ by $\mathcal{O}'$ should admit exactly shared models of $\mathcal{O}$ and $\mathcal{O}'$. Such models are encapsulated in $\mathsf{MT}(\mathcal{O}, \mathsf{sub}(\mathcal{O}')) \cap \mathsf{MT}(\mathcal{O}', \mathsf{sub}(\mathcal{O}))$ by our setting.

Finally, our goal is to build from $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ a revision ontology $\widehat{\mathcal{O}}$ that admits exactly $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ as tree models. However, we can show that there may not exist such an ontology $\widehat{\mathcal{O}}$ by reconsidering the example above with $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \{T_1'\}$. Assume that there exists an ontology $\widehat{\mathcal{O}}$ with $\mathsf{sub}(\widehat{\mathcal{O}}) = \{A, \neg A, B, \neg B, \exists R.(\neg B), \forall R.B\}$ which admits the unique T_1' as tree model. Due to the specific behavior of the sat-rule with $\mathsf{sub}(\widehat{\mathcal{O}})$, if we apply $\mathbb{TA}$ to $\widehat{\mathcal{O}}$ for building $\mathsf{MT}(\widehat{\mathcal{O}})$, we must obtain T_1 and another tree model T_2' with one node $\{x\}$, $L(x) = \{A, \forall R.B, B\}$, which is a contradiction.

For this reason, we use the notion of maximal approximation (De Giacomo *et al.*, 2007) to define an ontology $\mathcal{O}^*$ which satisfies the following conditions: (i) $\mathcal{O}^*$ is expressible in $\mathcal{SHIQ}$, (ii) it admits tree models in $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$, and (iii) it is a "smallest" ontology admitting $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Such an ontology $\mathcal{O}^*$, namely *maximal approximation*, can be built from the node labels of all tree models in $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$.

Definition 3 (Revision ontology). *Let $\mathcal{O}$ and $\mathcal{O}'$ be two consistent $\mathcal{SHIQ}$ ontologies with revision operation $\mathsf{MT}(\mathcal{O}, \mathcal{O}') = \{T_1, \cdots, T_n\}$ where $T_i = \langle V_i, L_i, E_i, \widehat{x_i} \rangle$ for $1 \leq i \leq n$. A revision ontology $\mathcal{O}^* = (\mathcal{T}, \mathcal{R})$ of $\mathcal{O}$ by $\mathcal{O}'$ can be built from completion trees in $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$ as follows: $\mathcal{R}$ includes the role hierarchy of $\mathcal{O}'$ and the one of $\mathcal{O}$; $\mathcal{T}$ contains all axioms of $\mathcal{O}'$ and the following axiom :* $\top \sqsubseteq \bigsqcup_{1 \leq i \leq n} \left(\bigsqcup_{x \in V_i} \left(\bigsqcap_{C \in L_i(x)} C \right) \right)$.

Theorem 1. *Let $\mathcal{O}$ and $\mathcal{O}'$ be two consistent $\mathcal{SHIQ}$ ontologies. The revision ontology $\mathcal{O}^*$ of $\mathcal{O}$ by $\mathcal{O}'$ is a maximal approximation from $\mathsf{MT}(\mathcal{O}, \mathcal{O}')$. Additionally, the size of $\mathcal{O}^*$ is bounded by a doubly exponential function in the size of $\mathcal{O}$ and $\mathcal{O}'$.*

Conclusion The main limitation of our approach is to omit individuals in ontologies. However, our approach can be extended in order to deal with individuals by extending the distance defined for completion trees to graphs. Another limitation is that the obtained revision ontology is very large. This exponential blow-up in size arises from doubly exponential size of completion trees. We believe that our procedure can be improved by using a method for compressing completion trees generated from tableau algorithms. Such a method has been proposed by Le Duc et al. (Le Duc *et al.*, 2013).

Acknowledgements This work was partially supported by FUI project "Learning Café".

References

[Alchourrón *et al.*, 1985] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change : Partial meet contraction and revision functions. *Journal of symbolic Logic*, 50:510–530, 1985.

[De Giacomo *et al.*, 2007] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On the approximation of instance level update and erasure in description logics. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 403–408, 2007.

[Flouris *et al.*, 2005] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the agm theory to dls and owl. In *In 4th International Semantic Web Conference (ISWC*, pages 216–231, 2005.

[Le Duc *et al.*, 2013] Chan Le Duc, Myriam Lamolle, and Olivier Curé. A decision procedure for SHOIQ with transitive closure of roles. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 264–279, 2013.

[Qi and Yang, 2008] Guilin Qi and Fangkai Yang. A survey of revision approaches in description logics. In Diego Calvanese and Georg Lausen, editors, *Web Reasoning and Rule Systems*, volume 5341 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin Heidelberg, 2008.

[Qi *et al.*, 2006] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *European Conference on Logics in Artificial Inteligence*, pages 386–398, 2006.

Extending the Combined Approach Beyond Lightweight Description Logics*

Cristina Feier[1], David Carral[2], Giorgio Stefanoni[1], Bernardo Cuenca Grau[1], Ian Horrocks[1]

[1] Department of Computer Science, University of Oxford, Oxford UK
[2] Department of Computer Science, Wright State University, Dayton US

Abstract. Combined approaches have become a successful technique for CQ answering over ontologies. Existing algorithms, however, are restricted to the logics underpinning the OWL 2 profiles. Our goal is to make combined approaches applicable to a wider range of ontologies. We focus on RSA: a class of Horn ontologies that extends the profiles while ensuring tractability of standard reasoning. We show that CQ answering over RSA ontologies without role composition is feasible in NP. Our reasoning procedure generalises the combined approach for $\mathcal{ELHO}$ and DL-Lite$_\mathcal{R}$ using an encoding of CQ answering into fact entailment w.r.t. a Logic Program with function symbols and stratified negation. Our results are significant in practice since many out-of-profile Horn ontologies are RSA.

1 Introduction

Answering conjunctive queries (CQs) over ontology-enriched datasets is a core reasoning task for many applications. CQ answering is computationally expensive: for expressive description logics it is at least doubly exponential in combined complexity [10], and it remains single exponential even when restricted to Horn ontologies [15].

Recently, there has been a growing interest in ontology languages with favourable computational properties, such as $\mathcal{EL}$ [1], DL-Lite [2] or the rule language datalog, which provide the foundation for the EL, QL and RL profiles of OWL 2, resp. [13]. Standard reasoning tasks (e.g., satisfiability checking) are tractable for all three profiles. CQ answering is NP-complete (in combined complexity) for the QL and RL profiles, and PSPACE-complete for OWL 2 EL [18]; PSPACE-hardness of CQ answering in EL is due to role composition axioms and the complexity further drops to NP if these are restricted to express role transitivity and reflexivity [16]. Furthermore, in all these cases CQ answering is tractable in data complexity. Such complexity bounds are rather benign, and this has spurred the development of a wide range of practical algorithms.

A technique that is receiving increasing attention is the *combined approach* [12, 7, 8, 11, 17]. Data is augmented in a query-independent way to build (in polynomial time) a canonical interpretation that might not be a model, but that can be exploited for CQ answering in two alternative ways: either the query is rewritten and then evaluated against

* Work supported by the Royal Society, the EPSRC grants Score!, DBOnto and MaSI^3, the NSF award 1017255 "III: Small: TROn: Tractable Reasoning with Ontologies" and "La Caixa" Foundation.

the interpretation [7] or the query is first evaluated over the interpretation and unsound answers are discarded by means of a *filtration* process [17, 11]. With the exception of [5] and [19] who focus on decidable classes of existential rules, algorithms based on the combined approach are restricted to (fragments of) the OWL 2 profiles.

Our goal is to push the boundaries of the logics underpinning the OWL 2 profiles while retaining their nice complexity for CQ answering. Furthermore, we aim to devise algorithms that seamlessly extend the combined approach and which can be applied to a wide range of ontologies.

Recently, a class of Horn ontologies, called *role safety acyclic* (RSA), has been proposed [3, 4]. RSA extends the profiles while ensuring tractability of standard reasoning tasks: it allows the use of all language constructs in the profiles, while establishing polynomially checkable conditions that preclude their harmful interaction. Roles in an RSA ontology are partitioned into *safe* and *unsafe* depending on the way they are used, where the latter ones are involved in potentially harmful interactions which could increase complexity; an *acyclicity* condition is imposed on unsafe roles to ensure tractability. A recent evaluation revealed that over 60% of out-of-profile Horn ontologies are RSA [4].

In this paper, we investigate CQ answering over RSA ontologies and show its feasibility in NP. This result has significant implications in practice as it shows that CQ answering over a wide range of out-of-profile ontologies is no harder (in combined complexity) than over a database. Our procedure generalises the combined approach for $\mathcal{ELHO}$ [17] and DL-Lite$_\mathcal{R}$ [11] in a seamless way by means of a declarative encoding of CQ answering into fact entailment w.r.t. a logic program (LP) with function symbols and stratified negation. The least Herbrand model of this program can be computed in time polynomial in the ontology size and exponential in query size. We have implemented our encoding using the LP engine DLV [9] and tested its feasibility with encouraging results. Proofs can be found in a TR (http://tinyurl.com/pqmxa5u).

2 Preliminaries

Logic Programs We use the standard notions of constants, terms and atoms in first-order logic (FO). A *literal* is an atom a or its negation $not\ a$. A rule r is an expression of the form $\varphi(\vec{x}, \vec{z}) \rightarrow \psi(\vec{x})$ with $\varphi(\vec{x}, \vec{z})$ a conjunction of literals with variables $\vec{x} \cup \vec{z}$, and $\psi(\vec{x})$ a non-empty conjunction of atoms over $\vec{x}$.[3] We denote with $vars(r)$ the set $\vec{x} \cup \vec{z}$. With $head(r)$ we denote the set of atoms in ψ, $body^+(r)$ is the set of atoms in φ, and $body^-(r)$ is the set of atoms which occur negated in r. Rule r is safe iff $vars(r)$ all occur in $body^+(r)$. We consider only safe rules. Rule r is *definite* if $body^-(r)$ is empty and it is *datalog* if it is definite and function-free. A *fact* is a rule with empty body and head consisting of a single function-free atom.

A program $\mathcal{P}$ is a finite set of rules. Let $preds(X)$ denote the predicates in X, with X a (set of) atoms or a program. A *stratification* of $\mathcal{P}$ is a function $str : preds(\mathcal{P}) \rightarrow \{1, \ldots, k\}$, where $k \leq |preds(\mathcal{P})|$, s.t. for every $r \in \mathcal{P}$ and $P \in preds(head(r))$ it holds that: *(i)* for every $Q \in preds(body^+(r))$: $str(Q) \leq str(P)$, and *(ii)* for every $Q \in preds(body^-(r))$: $str(Q) < str(P)$. The *stratification partition* of $\mathcal{P}$ induced

[3] We assume rule heads non-empty, and allow multiple atoms.

by str is the sequence $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$, with $\mathcal{P}_i$ consisting of all rules $r \in \mathcal{P}$ such that $max_{a \in head(r)}(str(pred(a))) = i$. The programs $\mathcal{P}_i$ are the *strata* of P. A program is *stratified* if it admits a stratification. All definite programs are stratified.

Stratified programs have a Least Herbrand Model (LHM), which is constructed using the immediate consequence operator $T_\mathcal{P}$. Let $\mathbf{U}$ and $\mathbf{B}$ be the Herbrand Universe and Base of $\mathcal{P}$, and let $S \subseteq \mathbf{B}$. Then, $T_\mathcal{P}(S)$ consists of all facts in $head(r)\sigma$ with $r \in \mathcal{P}$ and σ a substitution from $vars(r)$ to $\mathbf{U}$ satisfying $body^+(r)\sigma \subseteq S$ and $body^-(r)\sigma \cap S = \emptyset$. The powers of $T_\mathcal{P}$ are as follows: $T^0_\mathcal{P}(S) = S$, $T^{i+1}_\mathcal{P}(S) = T_\mathcal{P}(T^n_\mathcal{P}(S))$, and $T^\omega_\mathcal{P}(S) = \bigcup_{i=0}^\infty T^n_\mathcal{P}(S)$. Let str be a stratification of $\mathcal{P}$, and let $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ be its stratification partition. Also, let $U_1 = T^\omega_{\mathcal{P}_1}(\emptyset)$ and for each $1 \leq i \leq k$ let $U_{i+1} = T^\omega_{\mathcal{P}_{i+1}}(U_i)$. Then, the LHM of $\mathcal{P}$ is U_k and is denoted $M[\mathcal{P}]$. A program $\mathcal{P}$ entails a positive existential sentence α ($\mathcal{P} \models \alpha$) if $M[\mathcal{P}]$ seen as a FO structure satisfies α.

We use LPs to encode FO theories. For this, we introduce rules axiomatising the built-in semantics of the equality ($\approx$) and truth ($\top$) predicates. For a finite signature Σ, we denote with $\mathcal{F}^\top_\Sigma$ the smallest set with a rule $p(x_1, x_2, \ldots, x_n) \rightarrow \top(x_1) \wedge \top(x_2) \wedge \ldots \wedge \top(x_n)$ for each n-ary predicate p in Σ, and with $\mathcal{F}^\approx_\Sigma$ the usual axiomatisation of $\approx$ as a congruence over Σ. For an LP $\mathcal{P}$, we denote with $\mathcal{P}^{\approx,\top}$ the extension of $\mathcal{P}$ to $\mathcal{P} \cup \mathcal{F}^\top_\Sigma \cup \mathcal{F}^\approx_\Sigma$ with Σ the signature of $\mathcal{P}$.

Ontologies and Queries We define Horn-$\mathcal{ALCHOIQ}$ and specify its semantics via translation to definite programs. W.l.o.g. we consider a normal form close to that in [14]. Let N_C, N_R and N_I be countable pairwise disjoint sets of concept names, role names and individuals. We assume $\{\top, \bot\} \subseteq N_\mathsf{C}$. A *role* is an element of $N_\mathsf{R} \cup \{R^- | R \in N_\mathsf{R}\}$, where the roles in the latter set are called *inverse roles*. The function $\mathsf{Inv}(\cdot)$ is defined as follows, where $R \in N_\mathsf{R}$: $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. An *RBox* $\mathcal{R}$ is a finite set of axioms (R2) in Table 1, where R and S are roles and $\sqsubseteq^*_\mathcal{R}$ is the minimal reflexive-transitive relation over roles s.t. $\mathsf{Inv}(R) \sqsubseteq^*_\mathcal{R} \mathsf{Inv}(S)$ and $R \sqsubseteq^*_\mathcal{R} S$ hold if $R \sqsubseteq S \in \mathcal{R}$. A *TBox* $\mathcal{T}$ is a finite set of axioms (T1)-(T5) where $A, B \in N_\mathsf{C}$ and R is a role.[4] An *ABox* $\mathcal{A}$ is a finite set of axioms of the form (A1) and (A2), with $A \in N_\mathsf{C}$ and $R \in N_\mathsf{R}$. An *ontology* is a finite set of axioms $\mathcal{O} = \mathcal{R} \cup \mathcal{T} \cup \mathcal{A}$.

OWL 2 specifies the EL, QL, and RL profiles, which are all fragments of Horn-$\mathcal{ALCHOIQ}$ with the exception of property chain axioms and transitivity, which we do not consider here. An ontology is: *(i) EL* if it does not contain inverse roles or axioms (T4); *(ii) RL* if it does not contain axioms (T5); and *(iii) QL* if it does not contain axioms (T2) or (T4), each axiom (T1) satisfies $n = 1$, and each axiom (T3) satisfies $A = \top$.

A *conjunctive query (CQ)* Q is a formula $\exists \vec{y}.\psi(\vec{x}, \vec{y})$ with $\psi(\vec{x}, \vec{y})$ a conjunction of function-free atoms over $\vec{x} \cup \vec{y}$, where $\vec{x}$ are the *answer variables*. We denote with $terms(Q)$ the set of terms in Q. Queries with no answer variables are *Boolean* (BCQs) and for convenience are written as a set of atoms.

We define the semantics by a mapping π into definite rules as in Table 1: $\pi(\mathcal{O}) = \{\pi(\alpha) \mid \alpha \in \mathcal{O}\}$ [5]. An ontology $\mathcal{O}$ is satisfiable if $\pi(\mathcal{O})^{\approx,\top} \not\models \exists y.\bot(y)$. A tuple of constants $\vec{c}$ is an *answer* to Q if $\mathcal{O}$ is unsatisfiable, or $\pi(\mathcal{O})^{\approx,\top} \models \exists \vec{y}.\psi(\vec{c}, \vec{y})$. The set of answers is written $cert(Q, \mathcal{O})$. This semantics is equivalent to the usual one.

[4] Axioms $A \sqsubseteq \geq n\, R.B$ can be simulated by (T1) and (T5).

[5] By abuse of notation we say that $R^- \in \mathcal{O}$ whenever R^- occurs in $\mathcal{O}$.

Axioms α		Definite LP rules $\pi(\alpha)$
(R1)	R^-	$R(x,y) \to R^-(y,x); R^-(y,x) \to R(x,y)$
(R2)	$R \sqsubseteq S$	$R(x,y) \to S(x,y)$
(T1)	$\prod_{i=1}^n A_i \sqsubseteq B$	$\bigwedge_{i=1}^n A_i(x) \to B(x)$
(T2)	$A \sqsubseteq \{a\}$	$A(x) \to x \approx a$
(T3)	$\exists R.A \sqsubseteq B$	$R(x,y) \wedge A(y) \to B(x)$
(T4)	$A \sqsubseteq \leq 1R.B$	$A(x) \wedge R(x,y) \wedge B(y) \wedge R(x,z) \wedge B(z) \to y \approx z$
(T5)	$A \sqsubseteq \exists R.B$	$A(x) \to R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$
(A1)	$A(a)$	$\to A(a)$
(A2)	$R(a,b)$	$\to R(a,b)$

Table 1: Translation from Horn ontologies into rules.

3 Reasoning over RSA Ontologies

CQ answering is ExpTime-complete for Horn-$\mathcal{ALCHOIQ}$ ontologies [14], and the ExpTime lower bound holds already for satisfiability checking. Intractability is due to *and-branching*: owing to the interaction between axioms in Table 1 of type (T5) with either axioms (T3) and (R1), or axioms (T4) an ontology may only be satisfied by large (possibly infinite) models which cannot be succinctly represented.

RSA is a class of ontologies where all axioms in Table 1 are allowed, but their interaction is restricted s.t. model size can be polynomially bounded [4]. We recapitulate RSA ontologies and their properties; let $\mathcal{O}$ be an arbitrary Horn-$\mathcal{ALCHOIQ}$ ontology.

Roles in $\mathcal{O}$ are divided into *safe* and *unsafe*. The intuition is that unsafe roles may participate in harmful interactions.

Definition 1. *A role R is* unsafe *if it occurs in an axiom of the form $A \sqsubseteq \exists R.B$, and there is a role S s. t. either: 1. $R \sqsubseteq^*_{\mathcal{R}}$ $\mathsf{Inv}(S)$ and S occurs in an axiom of the form $\exists S.A \sqsubseteq B$ with $A \neq \top$, or 2. $R \sqsubseteq^*_{\mathcal{R}} S$ or $R \sqsubseteq^*_{\mathcal{R}}$ $\mathsf{Inv}(S)$ and S occurs in an axiom of the form $A \sqsubseteq \leq 1S.B$. A role R in $\mathcal{O}$ is* safe*, if it is not unsafe.*

It follows from Definition 1 that RL, QL, and EL ontologies contain only safe roles.

Example 1. Let $\mathcal{O}_{\mathsf{Ex}}$ be the (out-of-profile) ontology with the following axioms:

$$A(a) \quad (1) \qquad A \sqsubseteq \exists S^-.C \quad (3) \qquad D \sqsubseteq \exists R.B \quad (5) \qquad R \sqsubseteq T^- \quad (7)$$
$$A \sqsubseteq D \quad (2) \qquad \exists S.A \sqsubseteq D \quad (4) \qquad B \sqsubseteq \exists S.D \quad (6) \qquad S \sqsubseteq T \quad (8)$$

Roles R, S, T, and T^- are safe; however, S^- is unsafe as it occurs in an axiom (T5) while S occurs in an axiom (T3). We will $\mathcal{O}_{\mathsf{Ex}}$ use as a running example.

The distinction between safe and unsafe roles makes it possible to strengthen the translation π in Table 1 while preserving satisfiability and entailment of unary facts. The translation of axioms (T5) with R safe can be realised by replacing the functional term $f^A_{R,B}(x)$ with a Skolem constant $v^A_{R,B}$ unique to A, R and B. The modified transformation generally leads to a smaller LHM: if all roles are safe then $\mathcal{O}$ is mapped into a Datalog program whose LHM is polynomial in the size of $\mathcal{O}$.

Definition 2. *Let $v^A_{R,B}$ be a fresh constant for each concept A, B, and each safe role R in $\mathcal{O}$. Then π_{safe} maps each $\alpha \in \mathcal{O}$ to (i) $A(x) \rightarrow R(x, v^A_{R,B}) \wedge B(v^A_{R,B})$ if α is of type (T5) with R safe;(ii) $\pi(\alpha)$, otherwise. Let $\mathcal{P} = \{\pi_{\mathsf{safe}}(\alpha) \mid \alpha \in \mathcal{O}\}$ and $\mathcal{P}_\mathcal{O} = \mathcal{P}^{\approx,\top}$.*

Example 2. Mapping π_{safe} differs from π on ax. (5) and (6). For instance, (5) yields $D(x) \rightarrow R(x, v^D_{R,B}) \wedge B(v^D_{R,B})$.

Theorem 1. *[4, Theorem 2] Ontology $\mathcal{O}$ is satisfiable iff $\mathcal{P}_\mathcal{O} \not\models \exists y.\bot(y)$. If $\mathcal{O}$ is satisfiable, then $\mathcal{O} \models A(c)$ iff $A(c) \in M[\mathcal{P}_\mathcal{O}]$ for each unary predicate A and individual c in $\mathcal{O}$.*

If $\mathcal{O}$ has unsafe roles the model $M[\mathcal{P}_\mathcal{O}]$ might be infinite. We next define a Datalog program $\mathcal{P}_{\mathsf{RSA}}$ by introducing Skolem constants for all axioms (T5) in $\mathcal{O}$. $\mathcal{P}_{\mathsf{RSA}}$ introduces also a predicate PE which 'tracks' all binary facts generated by the application of Skolemised rules over unsafe roles. A unary predicate U is initialised with the constants associated to unsafe roles and a rule $\mathsf{U}(x) \wedge \mathsf{PE}(x, y) \wedge \mathsf{U}(y) \rightarrow \mathsf{E}(x, y)$ stores the PE-facts originating from unsafe roles using a predicate E. Then, $M[\mathcal{P}_\mathcal{O}]$ is of polynomial size when the graph induced by the extension of E is an oriented forest (i.e., a DAG whose underlying undirected graph is a forest). When this condition is fulfilled together with some additional conditions which preclude harmful interactions between equality-generating axioms and inverse roles, we say that $\mathcal{O}$ is RSA.

Definition 3. *Let PE and E be fresh binary predicates, U be a fresh unary predicate, and $u^A_{R,B}$ be a fresh constant for each concept A, B and each role R in $\mathcal{O}$. Function π_{RSA} maps each (i) $\alpha \in \mathcal{O}$ to $A(x) \rightarrow R(x, u^A_{R,B}) \wedge B(u^A_{R,B}) \wedge \mathsf{PE}(x, u^A_{R,B})$, if α is of type (T5), and to (ii) $\pi(\alpha)$, otherwise. The program $\mathcal{P}_{\mathsf{RSA}}$ consists of $\pi_{\mathsf{RSA}}(\alpha)$, for each $\alpha \in \mathcal{O}$, a rule $\mathsf{U}(x) \wedge \mathsf{PE}(x, y) \wedge \mathsf{U}(y) \rightarrow \mathsf{E}(x, y)$, and a fact $\mathsf{U}(u^A_{R,B})$ for each $u^A_{R,B}$ with R unsafe.*

Let M_{RSA} be the LHM of $\mathcal{P}_{\mathsf{RSA}}{}^{\approx,\top}$. Then, $G_\mathcal{O}$ is the digraph with an edge (c, d) for each $\mathsf{E}(c, d)$ in M_{RSA}. Ontology $\mathcal{O}$ is equality-safe if: 1. for each pair of atoms $w \approx t$ (with w and t distinct) and $R(t, u^A_{R,B})$ in M_{RSA} and each role S s.t. $R \sqsubseteq \mathsf{Inv}(S)$, it holds that S does not occur in an axiom (T4); and 2. for each pair of atoms $R(a, u^A_{R,B}), S(u^A_{R,B}, a)$ in M_{RSA}, with $a \in N_I$, there does not exist a role T such that both $R \sqsubseteq^*_\mathcal{R} T$ and $S \sqsubseteq^*_\mathcal{R} \mathsf{Inv}(T)$ hold.

We say that $\mathcal{O}$ is RSA if it is equality-safe and $G_\mathcal{O}$ is an oriented forest.

The fact that $G_\mathcal{O}$ is a DAG ensures that the LHM $M[\mathcal{P}_\mathcal{O}]$ is finite, whereas the lack of 'diamond-shaped' subgraphs in $G_\mathcal{O}$ guarantees polynomiality of $M[\mathcal{P}_\mathcal{O}]$. The safety condition on $\approx$ will ensure that RSA ontologies enjoy a special form of forest-model property that we exploit for CQ answering. Every ontology in QL (which is equality-free), RL (where $\mathcal{P}_{\mathsf{RSA}}$ has no Skolem constants) and EL (no inverse roles) is RSA.

Theorem 2. *[4, Theorem 3] If $\mathcal{O}$ is RSA, then $|M[\mathcal{P}_\mathcal{O}]|$ is polynomial in $|\mathcal{O}|$.*

Tractability of standard reasoning for RSA ontologies follows from Theorems 1, 2. It can be checked that $\mathcal{O}_{\mathsf{Ex}}$ is RSA.

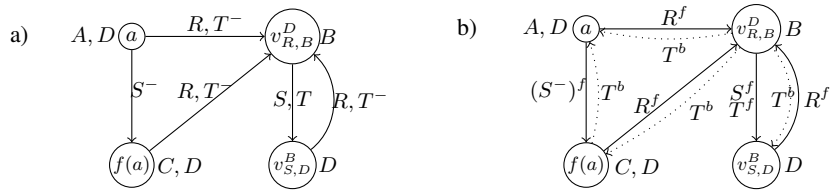


Fig. 1: Original (a) and annotated (b) model for $\mathcal{O}_{\mathsf{Ex}}$

4 Answering Queries over RSA Ontologies

We next present our combined approach with filtration to CQ answering over RSA ontologies, which generalises existing techniques for DL-Lite$_\mathcal{R}$ and $\mathcal{ELHO}$.

In Section 4.1 we take the LHM for RSA ontologies given in Section 3 as a starting point and extend it to a more convenient canonical model over an extended signature. In order to deal with the presence of inverse roles in RSA ontologies, the extended model captures the "directionality" of binary atoms; this will allow us to subsequently extend the filtration approach from [17] in a seamless way. The canonical model is captured declaratively as the LHM of an LP program over the extended signature.

As usual in combined approaches, this model is not universal and the evaluation of CQs may lead to spurious, i.e. unsound answers. In Section 4.2, we specify our filtration approach for RSA ontologies as the LHM of a stratified program. In the following, we fix an arbitrary RSA ontology $\mathcal{O} = \mathcal{R} \cup \mathcal{T} \cup \mathcal{A}$ and an input CQ Q, which we use to parameterise all our technical results.

4.1 Constructing the Canonical Model

The LHM $M[\mathcal{P}_\mathcal{O}]$ in Sec. 3 is a model of $\mathcal{O}$ that preserves entailment of unary facts. It generalises the canonical model in [17], which is specified as the LHM of a datalog program obtained by Skolemising all axioms (T5) into constants and hence coincides with $M[\mathcal{P}_\mathcal{O}]$ when $\mathcal{O}$ is EL. However, RSA ontologies allow for unsafe roles and hence $M[\mathcal{P}_\mathcal{O}]$ may contain also functional terms.

A main source for spurious matches when evaluating Q over the canonical model of an EL ontology is the presence of 'forks' — confluent chains of binary atoms — in the query which map to 'forks' in the model over Skolem constants. This is also problematical in our setting since RSA ontologies have the forest-model property.

Example 3. Fig. 1 a) depicts the LHM $M[\mathcal{P}_{\mathcal{O}_{\mathsf{Ex}}}]$ of $\mathcal{O}_{\mathsf{Ex}}$ (the function $f_{S,C}$ is abbreviated with f). We see models as digraphs where the direction of edges reflects the satisfaction of axioms (T5). Consider $Q_1 = \{A(y_1), R(y_1, y_2), R(y_3, y_2)\}$. Substitution $(y_1 \mapsto a, y_2 \mapsto v^D_{R,B}, y_3 \mapsto v^B_{S,D})$ is a spurious match of Q_1 as it relies on edges $(a, v^D_{R,B})$ and $(v^B_{S,D}, v^D_{R,B})$ in $M[\mathcal{P}_{\mathcal{O}_{\mathsf{Ex}}}]$, which form a fork over $v^D_{R,B}$.

In EL, only queries which contain forks can be mapped to forks in the model. This is no longer the case for RSA ontologies, where forks in the model can lead to spurious answers even for linearly-shaped queries due to the presence of inverse roles.

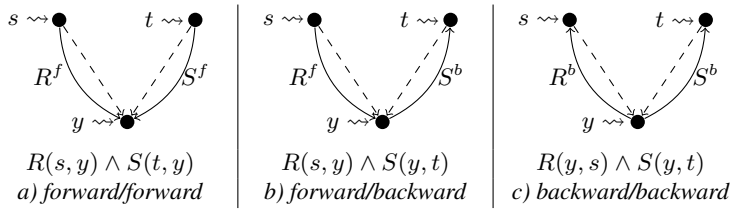


Fig. 2: Forks in the presence of inverse roles

Example 4. Let $Q_2 = \{A(y_1), R(y_1, y_2), T(y_2, y_3)\}$. Then $(y_1 \mapsto a, y_2 \mapsto v^D_{R,B}, y_3 \mapsto f(a))$ is a spurious match for Q_2 as it relies on the fork $(a, v^D_{R,B})$, $(f(a), v^D_{R,B})$. Axiom $R \sqsubseteq T^-$ causes a linear match over R and T to become a fork over R and T^-.

To identify such situations, we compute a canonical model over an extended signature that contains fresh roles R^f and R^b for each role R. Annotations f (forward) and b (backwards) are intended to reflect the directionality of binary atoms in the model, where binary atoms created to satisfy an axiom (T5) are annotated with f. To realise this intuition declaratively, we modify the rules in $\mathcal{P}_\mathcal{O}$ for axioms (T5) as follows. If R is safe, then we introduce the rule $A(x) \rightarrow R^f(x, v^A_{R,B}) \wedge B(v^A_{R,B})$; if it is unsafe, we introduce rule $A(x) \rightarrow R^f(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B})$ instead.

Superroles inherit the direction of the subrole, while roles and their inverses have opposite directions. To reflect this we include the following rules where $* \in \{f, b\}$: *(i)* $R^*(x, y) \rightarrow S^*(x, y)$ for each axiom $R \sqsubseteq S$ in $\mathcal{O}$; *(ii)* $R^f(x, y) \rightarrow \mathsf{Inv}(R)^b(y, x)$ and $R^b(x, y) \rightarrow \mathsf{Inv}(R)^f(y, x)$ for each role R; and *(iii)* $R^*(x, y) \rightarrow R(x, y)$ for each role R. Rules *(ii)* are included only if $\mathcal{O}$ has inverse roles, and rules *(iii)* 'copy' annotated atoms to atoms over the original predicate. Fig. 1 b) depicts the annotated model for $P_{\mathcal{O}_{\mathsf{Ex}}}$: solid (resp. dotted) lines represent 'forward' (resp. 'backward') atoms.

Fig. 2 depicts the ways in which query matches may spuriously rely on a fork in an annotated model. Nodes represent the images in the model of the query terms; solid lines indicate the annotated atoms responsible for the match; and dashed lines depict the underpinning fork. The images of s and t must not be equal; additionally, y cannot be mapped to (a term identified to) a constant in $\mathcal{O}$. For instance, the match in Ex. 4 is spurious as it corresponds to pattern (b) in Fig. 2. Unfortunately, the annotated model can present ambiguity: it is possible for both atoms $R^f(s, t)$ and $R^b(s, t)$ to hold.

Example 5. Consider Q_2 from Ex. 4. $(y_1 \mapsto a, y_2 \mapsto v^D_{R,B}, y_3 \mapsto v^B_{S,D})$ is also a match, where both $T^f(v^D_{R,B}, v^B_{S,D})$ and $T^b(v^D_{R,B}, v^B_{S,D})$ hold in the annotated model.

Such ambiguity is problematic for the subsequent filtration step. To disambiguate, we use a technique similar to the one in [11] for DL-Lite$_\mathcal{R}$, where the idea is to unfold certain cycles of length one and two in the canonical model. We unfold self-loops to cycles of length three while cycles of length two are unfolded to cycles of length four.

Example 6. Fig. 3 a) shows the model expansion for $\mathcal{O}_{\mathsf{Ex}}$. Ambiguities are resolved. Fig. 3 b) shows the unfolding of a generic self-loop over a safe role R for which T exists s.t. both $R \sqsubseteq^*_\mathcal{R} T$ and $R \sqsubseteq^*_\mathcal{R} \mathsf{Inv}(T)$ hold.

We now specify a program that yields the required model.

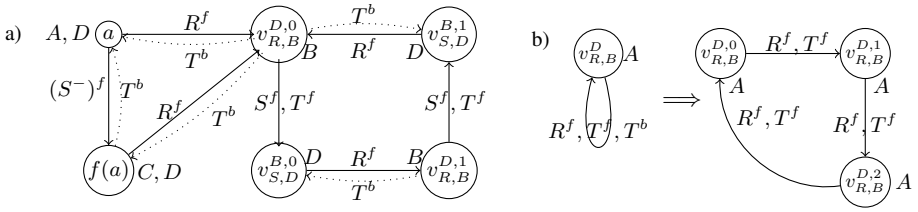


Fig. 3: Model expansion in the presence of loops/cycles

Definition 4. *Let* $\mathsf{confl}(R)$ *be the set of roles* S *s.t.* $R \sqsubseteq^*_{\mathcal{R}} T$ *and* $S \sqsubseteq^*_{\mathcal{R}} \mathsf{Inv}(T)$ *for some* T*. Let* $\prec$ *be a strict total order on triples* (A, R, B)*, with* R *safe and* A *and* B *concept names* B *in* $\mathcal{O}$*. For each* (A, R, B)*, let:*

- $v^{A,0}_{R,B}$*,* $v^{A,1}_{R,B}$*, and* $v^{A,2}_{R,B}$ *be fresh constants;*
- $\mathsf{self}(A, R, B)$ *be the smallest set containing* $v^{A,0}_{R,B}$ *and* $v^{A,1}_{R,B}$ *if* $R \in \mathsf{confl}(R)$*;*
- $\mathsf{cycle}(A, R, B)$ *be the smallest set containing, for each* $S \in \mathsf{confl}(R)$*,* $v^{D,0}_{S,C}$ *if* $(A, R, B) \prec (D, S, C)$*;* $v^{D,1}_{S,C}$ *if* $(D, S, C) \prec (A, R, B)$*;* $f^D_{S,C}(v^{A,0}_{R,B})$ *and every* $f^F_{T,E}(v^{A,0}_{R,B})$ *s. t.* $u^D_{S,C} \approx u^F_{T,E}$ *is in* M_{RSA}*, if* S *is unsafe.*
- $\mathsf{unfold}(A, R, B) = \mathsf{self}(A, R, B) \cup \mathsf{cycle}(A, R, B)$.

Let R^f *and* R^b *be fresh binary predicates for each role* R *in* $\mathcal{O}$*,* NI *be a fresh unary predicate, and* notIn *be a built-in predicate which holds when the first argument is an element of second argument. Let* $\mathcal{P}$ *be the smallest program with a rule* $\rightarrow \mathsf{NI}(a)$ *for each constant* a *and all rules in Fig. 4 and* $E_{\mathcal{O}} = \mathcal{P}^{\approx,\top}$.

symbols/axioms in $\mathcal{O}$	Logic Programming Rules
ax. α not of type (T5)	$\pi(\alpha)$
$R \sqsubseteq S, * \in \{f, b\}$	$R^*(x, y) \rightarrow S^*(x, y)$
R role, $* \in \{f, b\}$	$R^*(x, y) \rightarrow R(x, y)$ $R^f(x, y) \rightarrow \mathsf{Inv}(R)^b(y, x)$ $R^b(x, y) \rightarrow \mathsf{Inv}(R)^f(y, x)$
ax. (T5), R unsafe	$A(x) \rightarrow R^f(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$
ax. (T5), R safe	$A(x) \wedge \mathsf{notIn}(x, \mathsf{unfold}(A, R, B)) \rightarrow R^f(x, v^{A,0}_{R,B}) \wedge B(v^{A,0}_{R,B})$
	if $R \in \mathsf{confl}(R)$, for every $i = 0, 1$: $A(v^{A,i}_{R,B}) \rightarrow R^f(v^{A,i}_{R,B}, v^{A,i+1}_{R,B}) \wedge B(v^{A,i+1}_{R,B})$
	for every $x \in \mathsf{cycle}(A, R, B)$: $A(x) \rightarrow R^f(x, v^{A,1}_{R,B}) \wedge B(v^{A,1}_{R,B})$

Fig. 4: Rules in the program $E_{\mathcal{O}}$

The set $\mathsf{confl}(R)$ contains roles that may cause ambiguity in conjunction with R. The ordering $\prec$ determines how cycles are unfolded using auxiliary constants. Each axiom $A \sqsubseteq \exists R.B$ with R safe is Skolemised by default using $v^{A,0}_{R,B}$, except when the axiom applies to a term in $\mathsf{unfold}(A, R, B)$ where we use $v^{A,1}_{R,B}$ or $v^{A,2}_{R,B}$ instead.

Theorem 3. *The following holds: (i)* $M[E_{\mathcal{O}}]$ *is polynomial in* $|\mathcal{O}|$ *(ii)* $\mathcal{O}$ *is satisfiable iff* $E_{\mathcal{O}} \not\models \exists y.\bot(y)$ *(iii) if* $\mathcal{O}$ *is satisfiable,* $\mathcal{O} \models A(c)$ *iff* $A(c) \in M[E_{\mathcal{O}}]$ *and (iv) there are no terms* s, t *and role* R *s.t.* $E_{\mathcal{O}} \models R^f(s, t) \wedge R^b(s, t)$.

(1) $\psi(\vec{x},\vec{y}) \to \mathsf{QM}(\vec{x},\vec{y})$
(2) $\to named(a)$ for each constant a in $\mathcal{O}$
(3a) $\mathsf{QM}(\vec{x},\vec{y}), not\ \mathsf{NI}(y_i) \to \mathsf{id}(\vec{x},\vec{y},i,i)$, for each $1 \leq i \leq \lvert\vec{y}\rvert$ **(3b)** $\mathsf{id}(\vec{x},\vec{y},u,v) \to \mathsf{id}(\vec{x},\vec{y},v,u)$ **(3c)** $\mathsf{id}(\vec{x},\vec{y},u,v) \wedge \mathsf{id}(\vec{x},\vec{y},v,w) \to \mathsf{id}(\vec{x},\vec{y},u,w)$

for all $R(s,y_i)$, $S(t,y_j)$ in Q with $y_i, y_j \in \vec{y}$
(4a) $R^f(s,y_i) \wedge S^f(t,y_j) \wedge \mathsf{id}(\vec{x},\vec{y},i,j) \wedge not\ s \approx t \to \mathit{fk}(\vec{x},\vec{y})$
for all $R(s,y_i)$, $S(y_j,t)$ in Q with $y_i, y_j \in \vec{y}$:
(4b) $R^f(s,y_i) \wedge S^b(y_j,t) \wedge \mathsf{id}(\vec{x},\vec{y},i,j) \wedge not\ s \approx t \to \mathit{fk}(\vec{x},\vec{y})$
for all $R(y_i,s)$, $S(y_j,t)$ in Q with $y_i, y_j \in \vec{y}$:
(4c) $R^b(y_i,s) \wedge S^b(y_j,t) \wedge \mathsf{id}(\vec{x},\vec{y},i,j) \wedge not\ s \approx t \to \mathit{fk}(\vec{x},\vec{y})$

for all $R(y_i,y_j)$, $S(y_k,y_l)$ in Q with $y_i, y_j, y_k, y_l \in \vec{y}$: **(5a)** $R^f(y_i,y_j) \wedge S^f(y_k,y_l) \wedge \mathsf{id}(\vec{x},\vec{y},j,l) \wedge y_i \approx y_k \wedge not\ \mathsf{NI}(y_i) \to \mathsf{id}(\vec{x},\vec{y},i,k)$ **(5b)** $R^f(y_i,y_j) \wedge S^b(y_k,y_l) \wedge \mathsf{id}(\vec{x},\vec{y},j,k) \wedge y_i \approx y_l \wedge not\ \mathsf{NI}(y_i) \to \mathsf{id}(\vec{x},\vec{y},i,l)$ **(5c)** $R^b(y_i,y_j) \wedge S^b(y_l,y_k) \wedge \mathsf{id}(\vec{x},\vec{y},i,l) \wedge y_j \approx y_k \wedge not\ \mathsf{NI}(y_j) \to \mathsf{id}(\vec{x},\vec{y},j,k)$
for each $R(y_i,y_j)$ in Q with $y_i, y_j \in \vec{y}$, and $* \in \{f,b\}$: **(6)** $R^*(y_i,y_j) \wedge \mathsf{id}(\vec{x},\vec{y},i,v) \wedge \mathsf{id}(\vec{x},\vec{y},j,w) \to AQ^*(\vec{x},\vec{y},v,w)$
(7a) $AQ^*(\vec{x},\vec{y},u,v) \to TQ^*(\vec{x},\vec{y},u,v)$, for each $* \in \{f,b\}$ **(7b)** $AQ^*(\vec{x},\vec{y},u,v) \wedge TQ^*(\vec{x},\vec{y},v,w) \to TQ^*(\vec{x},\vec{y},u,w)$, for each $* \in \{f,b\}$
(8a) $\mathsf{QM}(\vec{x},\vec{y}) \wedge not\ named(x) \to \mathsf{sp}(\vec{x},\vec{y})$, for each $x \in \vec{x}$ **(8b)** $\mathit{fk}(\vec{x},\vec{y}) \to \mathsf{sp}(\vec{x},\vec{y})$ **(8c)** $TQ^*(\vec{x},\vec{y},v,v) \to \mathsf{sp}(\vec{x},\vec{y})$, for each $* \in \{f,b\}$
(9) $\mathsf{QM}(\vec{x},\vec{y}) \wedge not\ \mathsf{sp}(\vec{x},\vec{y}) \to \mathsf{Ans}(\vec{x})$

Table 2: Rules in $\mathcal{P}_Q$. Variables u, v, w from U are distinct.

4.2 Filtering Unsound Answers

We now define a program $\mathcal{P}_Q$ that can be used to eliminate all spurious matches of Q over the annotated model of $\mathcal{O}$. The rules of the program are summarised in Table 2. We will refer to all terms in the model that are not equal to a constant in $\mathcal{O}$ as *anonymous*.

Matches where an answer variable is not mapped to a constant in $\mathcal{O}$ are spurious. We introduce a predicate $named$ and populate it with such constants (rules **(2)**); then, we flag answers as spurious using a rule with negation (rules **(8a)**).

To detect forks we introduce a predicate fk, whose definition in datalog encodes the patterns in Fig. 2 (rules **(4)**). If terms s and t in Fig. 2 are existential variables mapping to the same anonymous term, further forks might be recursively induced.

Example 7. Let $Q_3 = \{A(y_1), R(y_1,y_2), T(y_2,y_3), C(y_4),\ \ R(y_4,y_5), S(y_5,y_3)\}$ be a BCQ over $\mathcal{O}_{\mathsf{Ex}}$, with $(y_1 \mapsto a, y_2 \mapsto v^{D,0}_{R,B}, y_3 \mapsto v^{B,0}_{S,D}, y_4 \mapsto f(a), y_5 \mapsto v^{D,0}_{R,B})$ being its only match over the model in Fig. 3a). The identity of y_2, y_5 induces a fork on the match of $R(y_1,y_2)$ and $R(y_4,y_5)$.

We track identities in the model relative to a match using a fresh predicate id. It is initialised as the minimal congruence relation over the positions of the existential variables in the query which are mapped to anonymous terms (rules **(3)**). Identity is recursively propagated (rules **(5)**). Matches involving forks are marked as spurious by rule **(8b)**.

Spurious matches can also be caused by cycles in the model and query satisfying certain requirements. First, the positions of existential variables of the query must

be cyclic when considering also the id relation. Second, the match must involve only anonymous terms. Finally, all binary atoms must have the same directionality.

Example 8. Consider the following BCQs over $\mathcal{O}_{\mathsf{Ex}}$: $Q_4 = \{S(y_1, y_2), R(y_2, y_3), S(y_3, y_4), R(y_4, y_1)\}$, $Q_5 = \{T(y_1, y_2), S(y_2, y_3), R(y_3, y_1)\}$, and $Q_6 = \{S(y_1, y_2), R(y_2, y_3), S(y_3, y_4), R(y_4, y_5)\}$. Then, $(y_1 \mapsto v^{D,0}_{R,B}, y_2 \mapsto v^{B,0}_{S,D}, y_3 \mapsto v^{D,1}_{R,B}, y_4 \mapsto v^{B,1}_{S,D})$ is a match of Q_4 inducing a cycle: all binary atoms are mapped 'forward' and the cycle involves only anonymous terms. In contrast, match $(y_1 \mapsto v^{D,0}_{R,B}, y_2 \mapsto f(a), y_3 \mapsto a)$ over Q_5 does not satisfy the requirements as it involves constant a. Note that Q_4 and Q_5 are cyclic. Q_6 is not cyclic; thus, although the match $(y_1 \mapsto v^{D,0}_{R,B}, y_2 \mapsto v^{B,0}_{S,D}, y_3 \mapsto v^{D,1}_{R,B}, y_4 \mapsto v^{B,1}_{S,D}, y_5 \mapsto v^{D,0}_{R,B})$ involves a cycle in the model, it is not spurious.

Such cycles are recognised by rules **(6)** and **(7)**. Rule **(6)** defines potential individual arcs in the cycle with their directionality using fresh predicates AQ^* with $* \in \{f, b\}$. Rules **(7)** detect the cycles recursively using predicates TQ^*. Matches involving cycles are marked as spurious by rules **(8c)**. All correct answers are collected by rule **(9)** using predicate Ans. We next define program $\mathcal{P}_Q$ and its extension $\mathcal{P}_{\mathcal{O},Q}$ with $E_\mathcal{O}$ in Def. 4, which can be exploited to answer Q w.r.t. $\mathcal{O}$.

Definition 5. *Let $Q = \exists \vec{y}.\psi(\vec{x}, \vec{y})$ be a CQ, let QM, sp, and fk be fresh predicates of arity $|\vec{x}| + |\vec{y}|$, let id, AQ^*, and TQ^*, with $* \in \{f, b\}$, be fresh predicates of arity $|\vec{x}| + |\vec{y}| + 2$, let Ans be a fresh predicate of arity $|\vec{x}|$, let $named$ be a fresh unary predicate, and let U be a set of fresh variables s.t. $|U| \geq |\vec{y}|$. Then, $\mathcal{P}_Q$ is the smallest program with all rules in Table 2, and $\mathcal{P}_{\mathcal{O},Q}$ is defined as $E_\mathcal{O} \cup \mathcal{P}_Q$.*

Note that, to distinguish between constants in $\mathcal{O}$ (recorded by $named$ in $\mathcal{P}_Q$) and their closure under equality (recorded by NI in $E_\mathcal{O}$), we do not axiomatise equality w.r.t. $\mathcal{P}_Q$.

Theorem 4. *(i) $\mathcal{P}_{\mathcal{O},Q}$ is stratified; (ii) $M[\mathcal{P}_{\mathcal{O},Q}]$ is polynomial in $|\mathcal{O}|$ and exponential in $|Q|$; and (iii) if $\mathcal{O}$ is satisfiable, $\vec{x} \in cert(Q, \mathcal{O})$ iff $\mathcal{P}_{\mathcal{O},Q} \models \mathsf{Ans}(\vec{x})$.*

Theorem 4 suggests a worst-case exponential algorithm that, given $\mathcal{O}$ and Q, materialises $\mathcal{P}_{\mathcal{O},Q}$ and returns the extension of predicate Ans. This procedure can be modified to obtain a 'guess and check' algorithm applicable to BCQs. This algorithm first materialises $E_\mathcal{O}$ in polynomial time; then, it guesses a match σ to Q over the materialisation; finally, it materialises $(\mathcal{P}_{\mathcal{O},Q})\sigma$, where variables $\vec{x}$ and $\vec{y}$ are grounded by σ. The latter step can also be shown to be tractable.

Theorem 5. *Checking whether $\mathcal{O} \models Q$ is* NP*-complete in combined complexity.*

5 Proof of Concept

We implemented our approach using the DLVsystem,[6] which supports function symbols and stratified negation. For testing, we used the LUBM ontology [6] (which contains only safe roles) and the Horn fragments of the Reactome and Uniprot (which are RSA, but contain also unsafe roles).[7] LUBM comes with a data generator; Reactome

[6] http://www.dlvsystem.com/dlv/

[7] http://www.ebi.ac.uk/rdf/platform

Ontology	Facts (M1)	Model M2/M3	q_1(M4/M5/M6)	q_2(M4/M5/M6)	q_3(M4/M5/M6)	q_4(M4/M5/M6)
Reactome	$54{\cdot}10^3$	8s / $242{\cdot}10^3$	6s / 10 / 0%	5s / 11 / 0%	6s / 50 / 48%	
	$107{\cdot}10^3$	16s / $485{\cdot}10^3$	14s / 11 / 0%	14s / 17 / 0%	12s / 122 / 38%	
	$159{\cdot}10^3$	21s / $728{\cdot}10^3$	42s / 17 / 0%	44s / 23 / 0%	36s/ 216 / 35%	
	$212{\cdot}10^3$	19s / $970{\cdot}10^3$	19s / 21 / 0%	15s/ 24 / 0%	14s/ 299 / 34%	
LUBM	$37{\cdot}10^3$	4s / $213{\cdot}10^3$	11s / 2350 / 86%	4s / 650/ 96%	4s / 1580/ 0%	5s / 1743/ 0%
	$75{\cdot}10^3$	6s / $395{\cdot}10^3$	45s / 9340/ 85%	8s / 1640/ 97%	9s / 7925/ 0%	8s / 5969/ 0%
	$113{\cdot}10^3$	8s / $550{\cdot}10^3$	108s / 24901/ 83%	13s / 2352/ 98%	13s / 18661/ 0%	13s / 10870/ 0%
	$150{\cdot}10^3$	11s / $682{\cdot}10^3$	188s / 52196/ 83%	17s / 2550/ 98%	18s / 32370/ 0%	24s / 15076/ 0%
	$188{\cdot}10^3$	12s / $795{\cdot}10^3$	305s / 91366/ 82%	31s / 2550/ 98%	40s / 49555/ 0%	38s / 18517/ 0%
	$226{\cdot}10^3$	14s / $894{\cdot}10^3$	390s / 148340/ 80%	39s / 2550/ 98%	46s / 72438/ 0%	40s / 20404/ 0%
Uniprot	$10{\cdot}10^3$	1s / $51{\cdot}10^3$	1s / 2 / 0%	1s / 0 / 0%	1s / 18 / 28%	
	$49{\cdot}10^3$	4s / $246{\cdot}10^3$	3s / 7 / 0%	3s / 0 / 0%	3s / 89 / 26%	
	$98{\cdot}10^3$	9s / $487{\cdot}10^3$	7s / 9 / 0%	6s / 1 / 0%	6s / 193 / 23%	
	$146{\cdot}10^3$	11s / $726{\cdot}10^3$	13s / 14 / 0%	12s / 1 / 0%	10s / 273 / 22%	

Table 3: Evaluation Results

and Uniprot come with large datasets, which we sampled. Test queries are given in the appendix. We measured (M1) number of facts of the given data; (M2) materialisation times for the canonical model; (M3) model size; (M4) materialisation times for $\mathcal{P}_Q$; (M5) number of candidate query answers; and (M6) percentage of spurious answers. Experiments were performed on a MacBook Pro laptop with 8GB RAM and an Intel Core 2.4 GHz processor.

Table 3 summarises our results. Computation times for the models scale linearly in data size. Model size is at most 6 times larger than the original data, which is a reasonable growth factor in practice. As usual in combined approaches (e.g. see [17]), query processing times depend on the number of candidate answers; thus, the applicability of the approach largely depends on the ratio between spurious and correct answers. Queries q_1-q_2 in Reactome and Uniprot are realistic queries given as examples in the EBI website. Neither of these queries lead to spurious answers, and processing times scale linearly with data size. No query in the LUBM benchmark leads to spurious answers (e.g., LUBM queries q_3 and q_4 in Table 3). We manually crafted one additional query for Reactome and Uniprot (q_3 in both cases) and two for LUBM (queries q_1 and q_2), which lead to a high percentage of spurious answers. Although these queries are challenging, we can observe that the proportion of spurious answers remains constant with increasing data size. Finally, note that query q_1 in LUBM retrieves the highest number of candidate answers and is thus the most challenging query. Our prototype and all test data, ontologies and queries are available at http://tinyurl.com/qcolx3w.

6 Conclusions and Future Work

We presented an extension to the combined approaches to CQ answering that can be applied to a wide range of out-of-profile Horn ontologies. Our theoretical results unify and extend existing techniques for $\mathcal{ELHO}$ and DL-Lite$_\mathcal{R}$ in a seamless and elegant way. Our preliminary experiments indicate the feasibility of our approach in practice.

We anticipate several directions for future work. First, we have not considered logics with transitive roles. Recently, it was shown that CQ answering over EL ontologies with transitive roles is feasible in NP [16]. We believe that our techniques can be extended in a similar way. Finally, we would like to optimise our encoding into LP and conduct a more extensive evaluation.

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369, 2005.
2. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Automated Reasoning (JAR)*, 39(3):385–429, 2007.
3. David Carral, Cristina Feier, Bernardo Cuenca Grau, Pascal Hitzler, and Ian Horrocks. $\mathcal{EL}$-ifying ontologies. In *IJCAR*, 2014.
4. David Carral, Cristina Feier, Bernardo Cuenca Grau, Pascal Hitzler, and Ian Horrocks. Pushing the boundaries of tractable ontology reasoning. In *ISWC*, 2014.
5. Georg Gottlob, Marco Manna, and Andreas Pieris. Polynomial combined rewritings for existential rules. In *KR*, 2014.
6. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Semantics*, 3(2-3):158–182, 2005.
7. Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in DL-Lite. In *KR*, 2010.
8. Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *IJCAI*, pages 2656–2661, 2011.
9. Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.
10. Carsten Lutz. Inverse roles make conjunctive queries hard. In *DL*, 2007.
11. Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. The combined approach to OBDA: Taming role hierarchies using filters. In *ISWC*, pages 314–330, 2013.
12. Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *IJCAI*, pages 2070–2075, 2009.
13. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, editors. *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation, 2009.
14. Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In *KR*, 2010.
15. Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *IJCAI*, pages 1039–1044, 2011.
16. Giorgio Stefanoni and Boris Motik. Answering conjunctive queries over $\mathcal{EL}$ knowledge bases with transitive and reflexive roles. In *AAAI*, 2015.
17. Giorgio Stefanoni, Boris Motik, and Ian Horrocks. Introducing nominals to the combined query answering approaches for $\mathcal{EL}$. In *AAAI*, 2013.
18. Giorgio Stefanoni, Boris Motik, Markus Krötzsch, and Sebastian Rudolph. The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. *J. Artif. Intell. Res. (JAIR)*, 51:645–705, 2014.
19. Michaël Thomazo and Sebastian Rudolph. Mixing materialization and query rewriting for existential rules. In *ECAI*, pages 897–902, 2014.

Adding Threshold Concepts to the Description Logic $\mathcal{EL}$

Franz Baader[1], Gerhard Brewka[2], and Oliver Fernández Gil[2⋆]
baader@tcs.inf.tu-dresden.de
{brewka,fernandez}@informatik.uni-leipzig.de

[1] Theoretical Computer Science, TU Dresden, Germany
[2] Department of Computer Science, University of Leipzig, Germany

The description logic (DL) $\mathcal{EL}$, in which concepts can be built using concept names as well as the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has drawn considerable attention in the last decade since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even with respect to expressive terminological axioms [6]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[3] In $\mathcal{EL}$ we can, for example, define the concept of a *happy man* as a male human that is healthy and handsome, has a rich and intelligent wife, a son and a daughter, and a friend:

$$\begin{aligned} &\mathsf{Human} \sqcap \mathsf{Male} \sqcap \mathsf{Healthy} \sqcap \mathsf{Handsome} \sqcap \\ &\exists \mathsf{spouse}.(\mathsf{Rich} \sqcap \mathsf{Intelligent} \sqcap \mathsf{Female}) \sqcap \\ &\exists \mathsf{child}.\mathsf{Male} \sqcap \exists \mathsf{child}.\mathsf{Female} \sqcap \exists \mathsf{friend}.\top \end{aligned} \tag{1}$$

For an individual to belong to this concept, all the stated properties need to be satisfied. However, maybe we would still want to call a man happy if most, though not all, of the properties hold. It might be sufficient to have just a daughter without a son, or a wife that is only intelligent but not rich, or maybe an intelligent and rich spouse of a different gender. But still, not too many of the properties should be violated.

In this paper, we introduce a DL extending $\mathcal{EL}$ that allows us to define concepts in such an approximate way. The main idea is to use a *graded membership function*, which instead of a Boolean membership value 0 or 1 yields a membership degree from the interval $[0, 1]$. We can then require a happy man to belong to the $\mathcal{EL}$ concept (1) with degree at least .8. More generally, if C is an $\mathcal{EL}$ concept, then the *threshold concept* $C_{\geq t}$ for $t \in [0, 1]$ collects all the individuals that belong to C with degree at least t. In addition to such upper threshold concepts, we will also consider lower threshold concepts $C_{\leq t}$ and allow the use of strict inequalities in both. For example, an unhappy man could be required to belong to the $\mathcal{EL}$ concept (1) with a degree less than .2.

⋆ Supported by DFG Graduiertenkolleg 1763 (QuantLA).
[3] see http://www.ihtsdo.org/snomed-ct/

The use of membership degree functions with values in the interval $[0,1]$ may remind the reader of fuzzy logics. However, there is no strong relationship between this work and the work on fuzzy DLs [5] for two reasons. First, in fuzzy DLs the semantics is extended to fuzzy interpretations where concept and role names are interpreted as fuzzy sets and relations, respectively. The membership degree of an individual to belong to a complex concept is then computed using fuzzy interpretations of the concept constructors. In our setting, we consider crisp interpretations of concept and role names, and directly define membership degrees for complex concepts based on them. Second, we use membership degrees to obtain new concept constructors, but the threshold concepts obtained by applying these constructors are again crisp rather than fuzzy.

We name our new logic $\tau\mathcal{EL}(m)$, where the membership degree function m is a parameter in defining the logic. In [2], we propose one specific such function *deg*, but we do not claim this is the only reasonable way to define such a function. Nevertheless, membership functions are not arbitrary. There are two properties we require such functions to satisfy:

Definition 1. *A graded membership function m is a family of functions that contains for every interpretation $\mathcal{I}$ a function $m^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$ satisfying the following conditions:*

$$M1 \ : d \in C^{\mathcal{I}} \Leftrightarrow m^{\mathcal{I}}(d, C) = 1$$
$$M2 \ : C \equiv D \Leftrightarrow \ \textit{for all } d \in \Delta^{\mathcal{I}} : m^{\mathcal{I}}(d, C) = m^{\mathcal{I}}(d, D).$$

Property *M2* expresses the intuition that the membership value should not depend on the syntactic form of a concept, but only on its semantics.

The set of $\tau\mathcal{EL}(m)$ concept descriptions is defined inductively, starting from finite sets of concept names $\mathsf{N_C}$ and role names $\mathsf{N_R}$, as follows:

$$\widehat{C}, \widehat{D} ::= \top \mid A \mid \widehat{C} \sqcap \widehat{D} \mid \exists r.\widehat{C} \mid E_{\sim q}$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $\sim \in \{<, \leq, >, \geq\}$, $q \in [0,1] \cap \mathbb{Q}$, E is an $\mathcal{EL}$ concept description, and $\widehat{C}, \widehat{D}$ are $\tau\mathcal{EL}(m)$ concept descriptions. For a given interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, the semantics of the new threshold concepts is defined as follows:

$$[E_{\sim q}]^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid m^{\mathcal{I}}(d, E) \sim q\}.$$

The extension of $.^{\mathcal{I}}$ to more complex concepts is defined as for $\mathcal{EL}$ by additionally considering the semantics of the newly introduced threshold concepts.

To make things more concrete, we introduce in [2] a specific membership function, denoted *deg*, which satisfies properties *M1* and *M2*. Given an interpretation $\mathcal{I}$, an element $d \in \Delta^{\mathcal{I}}$, and an $\mathcal{EL}$ concept description C, this function measures to which degree d satisfies the conditions for membership expressed by C. To come up with such a function, we use the homomorphism characterization of crisp membership in $\mathcal{EL}$. In $\mathcal{EL}$, concept descriptions and interpretations can be translated into $\mathcal{EL}$ description trees and $\mathcal{EL}$ description graphs, respectively (see [4,1]). Then, homomorphisms between $\mathcal{EL}$ description trees can be used to characterized subsumption in $\mathcal{EL}$ [4]. The proof of this result can be easily adapted to obtain the following characterization of *element-hood* in $\mathcal{EL}$.

Theorem 1. *Let $\mathcal{I}$ be an interpretation, $d \in \Delta^{\mathcal{I}}$ and C an $\mathcal{EL}$ concept description. Then, $d \in C^{\mathcal{I}}$ iff there exists a homomorphism φ from T_C to $G_{\mathcal{I}}$ such that $\varphi(v_0) = d$.*

Using Theorem 1 as a starting point, we consider all partial mappings h from T_C to $G_{\mathcal{I}}$ that map the root of T_C to d and respect the edge structure of T_C. For each of these mappings we then calculate to which degree it satisfies the homomorphism conditions, and take the degree of the best such mapping as the membership degree $deg^{\mathcal{I}}(d, C)$. Intuitively, to compute the degree associated to a partial mapping h, we define the *weighted* homomorphism induced by h as a function $h_w : \mathsf{dom}(h) \to [0, 1]$. Basically, in the definition of this function, the individual d is punished (in the sense that its membership degree is lowered) for each missing property (i.e., required element-hood in a concept name or an existential restriction) in a uniform way (see [2] for the precise definition).

In [2], we describe an algorithm that, given a finite interpretation $\mathcal{I}$, computes $deg^{\mathcal{I}}(d, C)$ in polynomial time. This polynomial time algorithm is inspired by the polynomial time algorithm for checking the existence of a homomorphism between $\mathcal{EL}$ description trees [3,4], and similar to the algorithm for computing the similarity degree between $\mathcal{EL}$ concept descriptions introduced in [9].

The main technical contribution of this work is, however, the investigation of the complexity of terminological (subsumption, satisfiability) and assertional (consistency, instance) reasoning in $\tau\mathcal{EL}(deg)$. To provide lower bounds, we show NP-hardness of the satisfiability problem by a simple reduction from the well-known NP-complete problem ALL-POS ONE-IN-THREE 3SAT [8]. The corresponding NP upper bound for satisfiability is an immediate consequence of the following polynomial bounded model property.

Lemma 1. *Let $\widehat{C}$ be a $\tau\mathcal{EL}(deg)$ concept description of size m. If $\widehat{C}$ is satisfiable, then there exists an interpretation $\mathcal{J}$ such that $\widehat{C}^{\mathcal{J}} \neq \emptyset$ and $|\Delta^{\mathcal{J}}| \leq m$.*

A coNP-upper bound for subsumption cannot directly be obtained from the fact that satisfiability is in NP. In fact, though we have $\widehat{C} \sqsubseteq \widehat{D}$ iff $\widehat{C} \sqcap \neg\widehat{D}$ is unsatisfiable, this equivalence cannot be used directly since $\neg\widehat{D}$ need not be a $\tau\mathcal{EL}(deg)$ concept description. Nevertheless, we can extend the ideas used in the proof of Lemma 1 to obtain a polynomial bounded model property for satisfiability of concepts of the form $\widehat{C} \sqcap \neg\widehat{D}$. The same is true for ABox consistency. Regarding instance checking, the bound on the size of counter models is exponential w.r.t. combined complexity, but fortunately still polynomial w.r.t. data complexity (in the sense of [7]).

Overall, we thus obtain the following complexity results for reasoning in $\tau\mathcal{EL}(deg)$.

Theorem 2. *In the DL $\tau\mathcal{EL}(deg)$, satisfiability is NP-complete, subsumption is coNP-complete, and ABox consistency is NP-complete. Moreover, instance checking is coNP-complete w.r.t. data complexity.*

Due to the space constraints, we could not provide technical details and proofs in this extended abstract. They can be found in the technical report [2].

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Gottlob, G., Walsh, T. (eds.) IJCAI. pp. 325–330. Morgan Kaufmann (2003)
2. Baader, F., Brewka, G., Fernández Gil, O.: Adding threshold concepts to the description logic $\mathcal{EL}$. LTCS-Report 15-09, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2015), see http://lat.inf.tu-dresden.de/research/reports.html.
3. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. LTCS-Report LTCS-98-09, LuFG Theoretical Computer Science, RWTH Aachen, Germany (1998), see http://lat.inf.tu-dresden.de/research/reports.html.
4. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Dean, T. (ed.) IJCAI. pp. 96–103. Morgan Kaufmann (1999)
5. Borgwardt, S., Distel, F., Peñaloza, R.: The limits of decidability in fuzzy description logics with general concept inclusions. Artificial Intelligence 218, 23–55 (2015)
6. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004. pp. 298–302. IOS Press (2004)
7. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Deduction in concept languages: From subsumption to instance checking. J. Log. Comput. 4(4), 423–452 (1994), http://dx.doi.org/10.1093/logcom/4.4.423
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
9. Suntisrivaraporn, B.: A similarity measure for the description logic EL with unfoldable terminologies. In: 2013 5th International Conference on Intelligent Networking and Collaborative Systems, Xi'an city, Shaanxi province, China, September 9-11, 2013. pp. 408–413. IEEE (2013), http://dx.doi.org/10.1109/INCoS.2013.77

Lower and Upper Bounds for SPARQL Queries over OWL Ontologies

Birte Glimm[1] Yevgeny Kazakov[1] Ilianna Kollia[2] and Giorgos Stamou[2]

[1] Ulm University, Germany, <firstname.surname>@uni-ulm.de
[2] National Technical University of Athens, Greece, ilianna2@mail.ntua.gr, gstam@cs.ntua.gr

Introduction

The recent standardization of the SPARQL 1.1 Entailment Regimes [2], which extend the SPARQL Query Language [3] with the capability of querying also for implicit knowledge makes the need for an efficient evaluation of complex queries over OWL ontologies urgent. We present an approach for optimizing the evaluation of SPARQL queries over OWL ontologies using SPARQL's OWL Direct Semantics entailment regime. Such queries consist of *axiom templates*, i.e., Description Logic (DL) axioms with variables in place of concept, role and individual names. Answers to such queries are mappings of query (concept, role or individual) variables to corresponding (concept, role or individual) names that instantiate the axiom templates to axioms entailed by the queried knowledge base (KB).

Since computing query answers over an expressive KB is computationally very costly, approximation techniques have been proposed that use a weakened version of the KB to compute a lower bound (yields sound but potentially incomplete results) and a strengthened version to compute an upper bound (yields complete but potentially unsound results) for the results [9, 7, 8]. Another well-known technique is to compute the bounds from a complete and clash-free tableau generated by a DL reasoner [4, 5]. Deterministically derived facts are used as lower bound, while also non-deterministically derived ones are considered for the upper bound. Answers in the "gap", i.e., potential answers in the upper but not the lower bound, usually have to be checked individually by performing a consistency check with a fully fledged OWL 2 DL reasoner.

While we also use bounds, we allow for much more expressive queries than related approaches. To optimize the evaluation of possible query answers in the gap, we present a query extension approach that uses the TBox of the queried KB to extend the query with additional parts. We show that the resulting query is equivalent to the original one and we use the additional parts that are simple to evaluate for restricting the bounds of subqueries of the initial query. In an empirical evaluation we show that the proposed query extension approach can lead to a significant decrease in the query execution time of up to four orders of magnitude. More details about our method as well as more evaluation results can be found in the extended version of our paper [1].

Improving Bounds via Query Extension

We will show the intuition of the proposed query extension method through an example. Let $\mathcal{K}$ be a KB, A, B, C be concept names, r a role name, a, b, c, d individual names, x

an individual variable, Y a concept variable and

$$\mathcal{T} = \{\, B \sqsubseteq A \sqcup C,\ \exists r.B \sqsubseteq C\} \quad \mathcal{A} = \{A(a),\ B(b),\ r(a,b),\ A(d)\}$$
$$q = \{A(x), \exists r.Y(x), Y \sqsubseteq B\}$$

Note that the only answer for q over $\mathcal{K}$ is the mapping $\{\mu \mid \mu(x) = a, \mu(Y) = B\}$.

First we compute bounds for axiom templates of q over $\mathcal{K}$. Since $\{A(a),\ A(d)\} \subseteq \mathcal{K}$ we have $\mathcal{K} \models A(a)$ and $\mathcal{K} \models A(d)$. That is, we can find the *lower bound* $\mathcal{L} = \{\mu \mid \mu(x) \in \{a, d\}\}$ for the query $\{A(x)\}$ over $\mathcal{K}$ without performing any tests. To find an *upper bound* for $\{A(x)\}$, we can use a model $\mathcal{I}$ of $\mathcal{K}$. It is easy to check that $\mathcal{K}$ has a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $\Delta^{\mathcal{I}} = \{d_1, d_2, d_3, d_4\}$, $a^{\mathcal{I}} = d_1$, $b^{\mathcal{I}} = d_2$, $c^{\mathcal{I}} = d_3$, $d^{\mathcal{I}} = d_4$, $A^{\mathcal{I}} = \{d_1, d_2, d_4\}$, $B^{\mathcal{I}} = \{d_2\}$, $C^{\mathcal{I}} = \{d_1\}$ and $r^{\mathcal{I}} = \{\langle d_1, d_2\rangle\}$. Note that $\mathcal{I} \not\models A(c)$. Thus, from this model alone one can conclude that $\mathcal{K} \not\models A(c)$ and hence that the set $\mathcal{U} = \{\mu \mid \mu(x) \in \{a, b, d\}\}$ provides an upper bound for the query $\{A(x)\}$ over $\mathcal{K}$. Although the model $\mathcal{I}$ can be similarly used for finding an upper bound for complex templates, such as $\{\exists r.Y(x)\}$, in general it can only be found by iterating over all possible mappings for x and Y and checking which instances of this template are entailed by the model. Therefore, in practice, one does not compute the bounds for complex templates. The bounds for the query $\{Y \sqsubseteq B\}$ can be computed by classifying the KB and retrieving subsumption relationships for B. Since for classification one usually needs to consider just the (relatively small) TBox $\mathcal{T}$, the bounds for this query can be computed exactly, i.e., in our example we have $\mathcal{L} = \mathcal{U} = \{\mu \mid \mu(Y) \in \{\bot, B\}\}$.

Our query extension method uses additionally the notion of a *subquery bound*,which provides a range for those answers of a subquery (subset of templates) of q that are sufficient to evaluate q. The intuition of our method is that subquery bounds can be improved using bounds of other subqueries of this query. Thus, if a query q can be extended to an *equivalent* query $q \cup q'$, the number of reasoner calls performed for evaluating q can be reduced using q'. The proposed algorithm can be summarized as follows: First, we replace every (concept, role, individual) variable in q with a fresh distinct (concept, role, individual) name. For our example, consider a mapping μ such that $\mu(x) = a_x$, $\mu(Y) = A_Y$ with a_x an individual and A_Y a concept name. Then we materialize and classify $\mathcal{K}$ plus $\mu(q)$, where $\mu(q)$ denotes the result of replacing each variable x in q with $\mu(x)$, i.e., we compute all concept assertions $A(a)$, role assertions $r(a, b)$, and subsumptions $A \sqsubseteq B$ with atomic concepts and roles entailed by $\mathcal{K} \cup \mu(q)$. Afterwards, we replace names back with their corresponding variables in the extended KB. In our example $\mathcal{T} \cup \mu(q) = \{B \sqsubseteq A \sqcup C, \exists r.B \sqsubseteq C, A(a_x), \exists r.A_Y(a_x), A_Y \sqsubseteq B\} \models C(a_x)$. Thus, for $q' = \{C(x)\}$, we have $\mathcal{K} \cup \mu(q) \models C(a_x) = \mu(q')$, and it holds that the query q has the same answers for $\mathcal{K}$ as the extended query $q \cup q' = \{A(x),\ \exists r.Y(x),\ Y \sqsubseteq B,\ C(x)\}$. In the end, we compute query bounds for the templates in q' and use them to improve the subquery bounds for templates in q. Using again the model $\mathcal{I}$ for $\mathcal{K}$, since $\mathcal{I} \models C(a)$, but $\mathcal{I} \not\models C(b)$, $\mathcal{I} \not\models C(c)$ and $\mathcal{I} \not\models C(d)$, we can derive the upper bound $\mathcal{U} = \{\mu \mid \mu(x) = a\}$ for the query $\{C(x)\}$. Using this upper bound, it is now possible to reduce the upper bound for the subquery $\{A(x)\}$ of q to $\mathcal{U}$. Since $\mathcal{U}$ is a subset of the lower bound for $\{A(x)\}$ (computed in the beginning of the section), this subquery can be evaluated without performing any further entailment tests. The new upper bound $\mathcal{U}$ can also be used to further reduce the upper bound for the subquery $\{\exists r.Y(x)\}$ of q to

Table 1. Query answering times in seconds (n/a indicates a timeout, > 30 min) and number of performed entailment checks for UOBM with the first department using evalStatic and evalExt

UOBM	evalStatic time	evalStatic #entail	evalExt time	evalExt #entail
$q_1 = \{isAdvisedBy(x, y), GraduateStudent(x), Woman(y)\}$, $q'_1 = \{Professor(y)\}$	20.84	47	10.54	19
$q_2 = \{isTaughtBy(x, y), GraduateCourse(x), Woman(y)\}$, $q'_2 = \{Faculty(y)\}$	21.63	51	12.06	26
$q_3 = \{teachingAssistantOf(x, y), GraduateCourse(y), Woman(x)\}$, $q'_3 = \{TeachingAssistant(x)\}$	12.78	32	5.60	12
$q_4 = \{\exists worksFor.Organization(x), Woman(x)\}$, $q'_4 = \{Employee(x)\}$	n/a		18.36	135
$q_5 = \{X \sqsubseteq \exists isHeadOf.Department, \exists isTaughtBy.X(y)\}$, $q'_5 = \{X \sqsubseteq Chair, CourseTaughtByChair(y)\}$	n/a		1.99	4
$q_6 = \{X \sqsubseteq \exists isHeadOf.College, \exists isAdvisedBy.X(y)$, $q'_6 = \{X \sqsubseteq Dean, PersonAdvisedByDean(y)\}$	n/a		0.21	0

$\{\mu \mid \mu(x) = a, \mu(Y) \in \{\bot, B\}\}$. After this reduction, this subquery can be evaluated using just two entailment tests.

Evaluation

The proposed method has been implemented and evaluated over a set of well-known benchmark ontologies and relevant datasets and for several forms of queries. Although it can be used, in general, for improving the performance of most query answering systems based on query bounds, here the evaluation is based on the system described in Kollia et al. [5], which, to the best of our knowledge, is the only system that supports the evaluation of complex queries over OWL 2 DL ontologies under the OWL Direct Semantics entailment regime of SPARQL 1.1. In our implemented method (referred to as evalExt) we improve the subquery bounds computed in evalStatic [5] using the method described in the previous section. Afterwards, we perform the ordering and evaluation methods of Kollia et al. using the improved subquery bounds.

In Table 1 we show the results of the evaluation on the University Ontology Benchmark (UOBM) [6] using a range of custom queries since the queries provided for UOBM are only simple conjunctive instance queries. Column 1 shows the query q_i and extension templates q'_i ($1 \leq i \leq 6$), columns 2 and 3 show the query answering times and the number of performed entailment checks for evalStatic, respectively, and columns 4 and 5 show the respective numbers for evalExt. In all queries the time spent for query extension is negligible compared to the time spent for query evaluation. We observe that for all queries, additional extension templates were derived, which have significantly better query bounds than the complex templates of the queries. This directly translates to a significantly lower number of entailment checks for evalExt and, hence, a reduction in execution times. The reduction in query answering times is up to four orders of magnitude.

References

1. Glimm, B., Kazakov, Y., Kollia, I., Stamou, G.: Lower and upper bounds for SPARQL queries over OWL ontologies. In: Proceedings of the 29th Conference on Artificial Intelligence (AAAI'15) (2015)
2. Glimm, B., Ogbuji, C. (eds.): SPARQL 1.1 Entailment Regimes. W3C Recommendation (2013), available at http://www.w3.org/TR/sparql11-entailment/
3. Harris, S., Seaborne, A. (eds.): SPARQL 1.1 Query Language. W3C Recommendation (2013), available at http://www.w3.org/TR/sparql11-query/
4. Kollia, I., Glimm, B.: Cost based query ordering over OWL ontologies. In: Proceedings of the 11th International Semantic Web Conference (ISWC 2012). Lecture Notes in Computer Science, vol. 7649, pp. 231–246. Springer (2012)
5. Kollia, I., Glimm, B.: Optimizing SPARQL Query Answering over OWL Ontologies. Journal of Artificial Intelligence Research 48, 253–303 (2013)
6. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: The Semantic Web: Research and Applications, pp. 125–139. Lecture Notes in Computer Science, Springer (2006)
7. Pan, J.Z., Thomas, E., Zhao, Y.: Completeness guaranteed approximation for OWL-DL query answering. In: Proceedings of the 22nd International Workshop on Description Logics (DL'09). CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009), http://dblp.uni-trier.de/db/conf/dlog/dlog2009.html#PanTZ09
8. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for TBox reasoning. In: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI'10). AAAI Press (2010)
9. Zhou, Y., Nenov, Y., Grau, B.C., Horrocks, I.: Pay-as-you-go OWL query answering using a triple store. In: Proceedings of the 28th Conference on Artificial Intelligence (AAAI'14). pp. 1142–1148 (2014)

Polynomial Combined Rewritings for Linear Existential Rules and DL-Lite with n-ary Relations

Georg Gottlob[1], Marco Manna[2], and Andreas Pieris[3]

[1] Department of Computer Science, University of Oxford, UK
georg.gottlob@cs.ox.ac.uk
[2] Department of Mathematics and Computer Science, University of Calabria, Italy
manna@mat.unical.it
[3] Institute of Information Systems, Vienna University of Technology, Austria
pieris@dbai.tuwien.ac.at

1 Introduction

This paper considers the setting of ontology-based query answering (OBQA). In this setting, Description Logics (DLs) and existential rules (a.k.a. tuple-generating dependencies or Datalog$^{\pm}$ rules) are popular ontology languages, while conjunctive queries (CQs) is the main querying tool. Among KR researchers there is a clear consensus that the required level of scalability in OBQA can only be achieved via query rewriting, which attempts to reduce OBQA into the problem of evaluating a query over a relational database. Two query languages are usually considered: *first-order queries* (FO) and *non-recursive Datalog queries* (NDL).

An interesting approach to query rewriting is the *polynomial combined approach* [7], which can be described as follows: an ontology Σ can be incorporated together with a CQ q into a database query q_Σ in polynomial time, and also with a database D into a database D_Σ in polynomial time, such that q_Σ over D_Σ yields the same result as q evaluated over the knowledge base consisting of D and Σ. The polynomial combined approach has been applied to $\mathcal{ELH}^{dr}_{\bot}$ [7], an extension of the well-known DL $\mathcal{EL}$, to DL-Lite$^{\mathcal{N}}_{horn}$ [5, 6], one of the most expressive logics of the DL-Lite family, and only recently to the main guarded- and sticky-based classes of existential rules [3].

Research Challenges. The problem of applying the polynomial combined approach to existing DLs and classes of existential rules is relatively understood. Nevertheless, there are still basic open questions that deserve our attention. Regarding DLs, little is known about formalisms with n-ary relations such as DLR-Lite$_{\mathcal{R}}$, that is, the extension of DL-Lite$_{\mathcal{R}}$ with n-ary roles. Regarding existential rules, it is open whether the polynomial combined approach can be applied to the class of *linear existential rules* (or simply *linear rules*), that is, assertions of the form $\forall\mathbf{X}\forall\mathbf{Y}(s(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\, p(\mathbf{X},\mathbf{Z}))$, where $s(\mathbf{X},\mathbf{Y})$ and $p(\mathbf{X},\mathbf{Z})$ are atomic formulas [1].

It is not difficult to show that, if linear rules are polynomially combined rewritable, then also DLR-Lite$_{\mathcal{R}}$ is polynomially combined rewritable — this follows from the fact that query answering under DLR-Lite$_{\mathcal{R}}$ can be easily reduced to query answering under linear rules [1]. Thus, the key question that we need to answer, which has been explicitly stated as an open problem in [3], is the following:

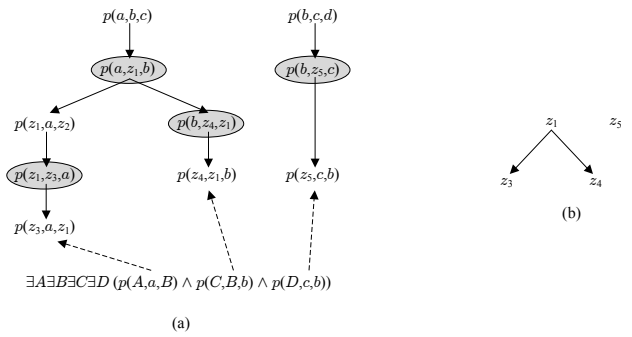


Fig. 1. Illustration of a proof generator.

Given a (Boolean) CQ q, a database D, and a set Σ of linear rules, can we rewrite in polynomial time: (i) q and Σ, independently of D, into a $\mathcal{Q}$-query q_Σ, where $\mathcal{Q} \in \{\mathsf{FO}, \mathsf{NDL}\}$, and (ii) D and Σ, independently of q, into a database D_Σ, such that $(D \cup \Sigma) \models q$ iff $D_\Sigma \models q_\Sigma$?

The answer to the above question is affirmative under the assumption that the arity of the underlying schema is bounded; implicit in [2]. However, little is known for arbitrary linear rules, without any assumption on the underlying schema. We give a positive answer even for linear rules that use predicates of unbounded arity. For more details, we refer the reader to [4].

2 Proof Generator

We assume the reader is familiar with the *chase procedure*. Recall that the chase for a database D and a set Σ of rules, denoted $chase(D, \Sigma)$, is a *universal model* of D and Σ, and thus $(D \cup \Sigma) \models q$ iff $chase(D, \Sigma) \models q$, for each CQ q. The instance $chase(D, \Sigma)$ can be naturally seen as a directed labeled graph, called *chase graph*, denoted $CG(D, \Sigma)$. It is also easy to verify that for linear rules, $CG(D, \Sigma)$ is a directed forest; for details on the chase, see, e.g., [1]. Our main technical tool is called proof generator, and it formalizes the intuitive idea that (Boolean) CQ answering under linear rules can be realized as a reachability problem on the chase graph. Let us illustrate the key ideas underlying the proof generator via a simple example.

Example 1. Let $D = \{p(a, b, c), p(b, c, d)\}$, and let Σ be the set of linear rules (for brevity, the universal quantifiers are omitted):

$$p(X, Y, Z) \rightarrow \exists W\, p(X, W, Y) \qquad p(X, Y, Z) \rightarrow \exists W\, p(Z, W, Y)$$
$$p(X, Y, Z) \rightarrow \exists W\, p(Y, X, W) \qquad p(X, Y, Z) \rightarrow p(Y, Z, X).$$

Given the BCQ $q = \exists A \exists B \exists C \exists D\, (p(A, a, B) \wedge p(C, B, b) \wedge p(D, c, b))$, as shown in Figure 1(a), there exists a homomorphism h (dashed arrows in the figure) that maps q to an initial segment of the chase graph of D and Σ, and thus $(D \cup \Sigma) \models q$. It is interesting to observe that the nulls occurring in $h(q)$, i.e., z_1, z_3, z_4 and z_5, form a rooted forest

F, depicted in Figure 1(b), with the following properties; for brevity, let $\nu(z)$ be the atom in $CG(D, \Sigma)$ where the null z is invented (see shaded atoms in Figure 1(a) for $\nu(z_1)$, $\nu(z_3)$, $\nu(z_4)$ and $\nu(z_5)$): (i) for every root node z, $\nu(z)$ is reachable from D; (ii) for every edge (z, w), $\nu(w)$ is reachable from $\nu(z)$; and (iii) for every atom $\underline{a} \in h(q)$, there exists a unique path π in F that contains all the nulls in $\underline{a}$, and, assuming that the leaf node of π is z, $\underline{a}$ is reachable from $\nu(z)$. Indeed, it is easy to verify that $\nu(z_1)$ and $\nu(z_5)$ are reachable from D, $\nu(z_3)$ and $\nu(z_4)$ are reachable from $\nu(z_1)$, and finally, $h(p(A, a, B)) = p(z_3, a, z_1)$ is reachable from $\nu(z_3)$, $h(p(C, B, b)) = p(z_4, z_1, b)$ is reachable from $\nu(z_4)$, and $h(p(D, c, b)) = p(z_5, c, b)$ is reachable from $\nu(z_5)$. ∎

Roughly speaking, the triple consisting of: (1) the homomorphism h, that maps q to the chase; (2) the function ν, that gives the atoms in the chase where the nulls occurring in $h(q)$ were invented; and (3) the forest F, that encodes how the nulls of $h(q)$ depend on each other, as well as the order of their generation, is what we call a proof generator for q w.r.t. D and Σ. The existence of such a triple, allows us to generate the part of the chase due to which the query is entailed, i.e., the proof of the query (hence the name "proof generator"). Therefore, for query answering purposes under linear rules, we simply need to check if such a proof generator exists.

Lemma 1. $(D \cup \Sigma) \models q$ *iff there exists a proof generator for* q *w.r.t.* D *and* Σ.

3 The Combined Rewriting

We give a positive answer to our research question regarding linear rules and the polynomial combined approach. More precisely, we show that:

Theorem 1. *The class of linear rules is polynomially combined* $\mathcal{Q}$*-rewritable, where* $\mathcal{Q} \in \{\mathsf{FO}, \mathsf{NDL}\}$.

To establish the above theorem, we heavily rely on the notion of the proof generator. Fix a (Boolean) CQ q, a database D, and a set Σ of linear rules. By Lemma 1, it suffices to construct in polynomial time a query $q_\Sigma \in \mathcal{Q}$ (independently of D), and a database D_Σ (independently of q), such that $D_\Sigma \models q_\Sigma$ iff a proof generator for q w.r.t. D and Σ exists. Roughly, the query q_Σ guesses a triple (h, ν, F) (as described in Example 1), and then verifies that the guessed triple is a proof generator for q w.r.t. D and Σ. The interesting part of q_Σ is the component that applies the crucial reachability checks required by the definition of the proof generator. Although the path among two atoms in the chase graph may be of exponential size, its existence can be checked via $\mathcal{Q}$-queries of polynomial size. An immediate consequence of Theorem 1 is that:

Corollary 1. *The description logic DLR-Lite*$_\mathcal{R}$ *is polynomially combined* $\mathcal{Q}$*-rewritable, where* $\mathcal{Q} \in \{\mathsf{FO}, \mathsf{NDL}\}$.

The results of this work are, for the moment, of theoretical nature and we do not claim that they will directly lead to better practical algorithms. We believe that a smart implementation of the proposed techniques can lead to more efficient rewriting procedures; this will be the subject of future research. We are also planning to optimize the proposed rewriting algorithm, with the aim of constructing queries of optimal size.

Acknowledgements. G. Gottlob was supported by the EPSRC Programme Grant EP/M025268/ "VADA: Value Added Data Systems – Principles and Architecture". M. Manna was supported by the MIUR project "SI-LAB BA2KNOW – Business Analitycs to Know", and by Regione Calabria, programme POR Calabria FESR 2007-2013, projects "ITravel PLUS" and "KnowRex: Un sistema per il riconoscimento e l'estrazione di conoscenza". A. Pieris was supported by the Austrian Science Fund (FWF): P25207-N23 and Y698.

References

1. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
2. Gottlob, G., Kikot, S., Kontchakov, R., Podolskii, V.V., Schwentick, T., Zakharyaschev, M.: The price of query rewriting in ontology-based data access. Artif. Intell. 213, 42–59 (2014)
3. Gottlob, G., Manna, M., Pieris, A.: Polynomial combined rewritings for existential rules. In: KR (2014)
4. Gottlob, G., Manna, M., Pieris, A.: Polynomial rewritings for linear existential rules. In: IJCAI (2015)
5. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR (2010)
6. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI. pp. 2656–2661 (2011)
7. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: IJCAI. pp. 2070–2075 (2009)

The Complexity of Temporal Description Logics with Rigid Roles and Restricted TBoxes: In Quest of Saving a Troublesome Marriage

Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Thomas Schneider

Department of Computer Science, Universität Bremen
{victor,jeanjung,ts}@cs.uni-bremen.de

1 Introduction

Temporal description logics (TDLs) extend classical DLs, providing built-in means to represent and reason about temporal aspects of knowledge. The importance of TDLs stems from the need of relevant applications to capture temporal and dynamic aspects of knowledge, e.g., in medical and life science ontologies, which are very large but still demand efficient reasoning, such as SNOMED CT and FMA [9], and the gene ontology (GO) [20]. A natural task is to model *dynamic* knowledge about patient histories against *static* medical knowledge (e.g., about diseases): e.g., the temporal concept $C := \mathbf{E}\Diamond\exists\,\mathsf{requiresTransfusion}.\top$ describes a patient who may need a blood transfusion in the future, and the axiom $\mathsf{Anemic} \sqsubseteq C$ says that this applies to anemic people. In contrast, $\mathsf{Anemia} \sqsubseteq \mathsf{Disorder}$ represents static knowledge.

A notable approach to designing TDLs is to combine DLs with temporal logics commonly used in software/hardware verification such as LTL, CTL$^{(*)}$, and to provide a two-dimensional product-like semantics [19, 11, 17]. The combination allows various design choices, e.g., we can restrict the scope of temporal operators to certain types of entities (such as concepts, roles, axioms), or declare some DL concepts or roles as rigid, meaning that their interpretation will not change over time. The need for rigid roles in TDL applications, e.g., in biomedical ontologies to accurately capture life-time relations, has been identified [7]. For example, the role $\mathsf{hasBloodType}$ should be rigid since a human's blood type does not change during their lifetime.

Alas, TDLs based on the Boolean-complete DL $\mathcal{ALC}$ with rigid roles cannot be effectively used since they become undecidable when temporal operators are applied to concepts and a general TBox is allowed [11, 15]. This is the case even if we severely restrict the temporal operators available and use the sub-Boolean DL $\mathcal{EL}$, whose standard reasoning problems are tractable, instead of $\mathcal{ALC}$ [1, 15]. In the light of these results, several efforts have been devoted to design decidable TDLs with rigid roles [3, 2]; e.g., decidability can be recovered by using a lightweight DL component based on *DL-Lite*. Both the $\mathcal{EL}$ and *DL-Lite* families underlie prominent profiles of the OWL standard.

Interestingly, no research has been yet devoted to TDLs based on $\mathcal{EL}$ in the presence of restricted TBoxes, such as classical TBoxes, which consist solely of definitions of the form $A \equiv C$ with A atomic and unique, or acyclic TBoxes, which additionally forbid syntactic cycles in definitions. This is surprising since in the presence of general TBoxes TDLs based on $\mathcal{EL}$ tend to be as complex as the $\mathcal{ALC}$ variant [3, 13, 15].

These considerations lead us to investigating TDLs with rigid roles based on $\mathcal{EL}$ and the (branching-time) CTL allowing for temporal concepts and acyclic TBoxes. We are convinced that TDLs designed in this way are suitable for temporal extensions of biomedical ontologies: large parts of SNOMED CT and GO are acyclic $\mathcal{EL}$-TBoxes.

Our main contributions are algorithms for standard reasoning problems and (mostly tight) complexity bounds. We begin by showing that the combination of CTL and $\mathcal{ALC}$ with empty and acyclic TBoxes is decidable. Our nonelementary upper bound is optimal even when the set of temporal operators is restricted to $\mathbf{E}\Diamond$ ("possibly eventually") or $\mathbf{E}\bigcirc$ ("possibly next"). We then replace $\mathcal{ALC}$ with $\mathcal{EL}$ and maintain the restriction to $\mathbf{E}\Diamond, \mathbf{E}\bigcirc$ and empty TBoxes. We particularly show that the resulting TDLs are decidable in PTIME with one of the two operators, and CONP-complete with both. To this aim, we employ canonical models, together with expansion vectors [16] in the case with both $\mathbf{E}\Diamond, \mathbf{E}\bigcirc$. Next, we lift the PTIME upper bound to the case of acyclic TBoxes, employing a completion algorithm in the style of those for $\mathcal{EL}$ and extensions, [5]. Finally, we show that the combination of $\mathbf{E}\Diamond$ with $\mathbf{A}\Box$ ("always globally") and acyclic TBoxes leads to a PSPACE-complete TDL, again employing a completion algorithm. An overview of existing and new results is given in Table 1, where CTL_X^Y denotes the combination of the DL X with the fragment of CTL restricted to the temporal operators Y. In particular, all the new results hold even if rigid concepts are also included.

Rigid roles? TBoxes	no general	yes general	yes acyclic	yes empty
$\text{CTL}_{\mathcal{ALC}}$	=EXPTIME [13]	undecidable [15]	nonelementary, decidable (1)	nonelem., decidable (1)
$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$	$\leq$PTIME [13]	nonelementary [15]	$\leq$PTIME (6)	$\leq$PTIME (6)
$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$	$\leq$PTIME [13]	undecidable [15]	$\leq$PTIME (6)	$\leq$PTIME (6)
$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$	=EXPTIME [13, 15]	undecidable [15]	$\geq$CONP, (2) $\leq$CONEXPTIME (5)	=CONP (2)
$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond,\mathbf{A}\Box}$	=PSPACE [13]	nonelementary [15]	=PSPACE (9)	$\leq$PSPACE (9)

Table 1. Previous and new complexity results. $\geq$ hardness, $\leq$ membership, $=$ completeness. (n) refers to our Theorem or Corollary n.

The relatively low complexity that we obtain for $\mathcal{EL}$-based TDLs over restricted TBoxes are in sharp contrast with the undecidability and nonelementary lower bounds known for the same logics over general TBoxes [15]. With the restriction to acyclic TBoxes, we will thus identify the first computationally well-behaved TDLs with rigid roles based on $\mathcal{EL}$ and classical temporal logics.

Additional technical notions and proofs are in a report: http://tinyurl.com/ijcai15tdl

2 Preliminaries

We introduce $\text{CTL}_{\mathcal{ALC}}$, a TDL based on the classical DL $\mathcal{ALC}$. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of *concept names* and *role names*, respectively. We assume that $\mathsf{N_C}$ and $\mathsf{N_R}$ are partitioned into two countably infinite sets: $\mathsf{N_C^{rig}}$ and $\mathsf{N_C^{loc}}$ of *rigid concept*

names and *local concept names*, respectively; and, $\mathsf{N}_\mathsf{R}^\mathsf{rig}$ and $\mathsf{N}_\mathsf{R}^\mathsf{loc}$ of *rigid role names* and *local role names*, respectively. CTL$_{\mathcal{ALC}}$*-concepts* C are defined by the grammar

$$C := \top \mid A \mid \neg C \mid C \sqcap D \mid \exists r.C \mid \mathbf{E}{\bigcirc}C \mid \mathbf{E}\Box C \mid \mathbf{E}(C\,\mathcal{U}\,D)$$

where A ranges over N_C, r over N_R. We use standard DL abbreviations [6] and temporal abbreviations $\mathbf{E}{\Diamond}C$, $\mathbf{A}\Box C$, $\mathbf{A}{\Diamond}C$ and $\mathbf{A}(C\,\mathcal{U}\,D)$ [10].

The semantics of classical DLs, such as $\mathcal{ALC}$, is given in terms of *interpretations* of the form $\mathcal{I} = (\Delta, \cdot^\mathcal{I})$, where Δ is a non-empty set called the *domain* and $\cdot^\mathcal{I}$ is an *interpretation function* that maps each $A \in \mathsf{N}_\mathsf{C}$ to a subset $A^\mathcal{I} \subseteq \Delta$ and each $r \in \mathsf{N}_\mathsf{R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta \times \Delta$. The semantics of CTL$_{\mathcal{ALC}}$ is given in terms of temporal interpretations based on infinite trees [15]: A *temporal interpretation* based on an infinite tree $T = (W, E)$ is a structure $\mathfrak{I} = (T, (\mathcal{I}_w)_{w \in W})$ such that, for each $w \in W$, $\mathcal{I}_w$ is a DL interpretation with domain Δ; and, $r^{\mathcal{I}_w} = r^{\mathcal{I}_{w'}}$ and $A^{\mathcal{I}_w} = A^{\mathcal{I}_{w'}}$ for all $r \in \mathsf{N}_\mathsf{R}^\mathsf{rig}$, $A \in \mathsf{N}_\mathsf{C}^\mathsf{rig}$ and $w, w' \in W$. We usually write $A^{\mathfrak{I},w}$ instead of $A^{\mathcal{I}_w}$. The stipulation that all worlds share the same domain is called the *constant domain assumption (CDA)*. For Boolean-complete TDLs, CDA is the most general: increasing, decreasing and varying domains can all be reduced to it [11, Prop. 3.32]. For the sub-Boolean logics studied here, CDA is not w.l.o.g. Indeed, we identify a logic in which reasoning with increasing domains cannot be reduced to the constant domain case.

We now define the semantics of CTL$_{\mathcal{ALC}}$-concepts. A *path* in $T = (W, E)$ starting at a node w is an infinite sequence $\pi = w_0 w_1 w_2 \cdots$ with $w_0 = w$ and $(w_i, w_{i+1}) \in E$. We write $\pi[i]$ for w_i, and use $\mathsf{Paths}(w)$ to denote the set of all paths starting at the node w. The mapping $\cdot^{\mathfrak{I},w}$ is extended from concept names to CTL$_{\mathcal{ALC}}$-concepts as follows.

$$\top^{\mathfrak{I},w} = \Delta \qquad (C \sqcap D)^{\mathfrak{I},w} = C^{\mathfrak{I},w} \cap D^{\mathfrak{I},w}$$

$$\begin{aligned}
(\exists r.C)^{\mathfrak{I},w} &= \{d \in \Delta \mid \exists e\,.\,(d,e) \in r^{\mathfrak{I},w} \wedge e \in C^{\mathfrak{I},w}\}\\
(\mathbf{E}{\bigcirc}C)^{\mathfrak{I},w} &= \{d \mid \exists \pi \in \mathsf{Paths}(w)\,.\,d \in C^{\mathfrak{I},\pi[1]}\}\\
(\mathbf{E}\Box C)^{\mathfrak{I},w} &= \{d \mid \exists \pi \in \mathsf{Paths}(w)\,.\,\forall j \geq 0\,.\,d \in C^{\mathfrak{I},\pi[j]}\}\\
(\mathbf{E}(C\mathcal{U}D))^{\mathfrak{I},w} &= \{d \mid \exists \pi \in \mathsf{Paths}(w)\,.\,\exists j \geq 0\,.\,(d \in D^{\mathfrak{I},\pi[j]} \wedge (\forall k < j\,.\,d \in C^{\mathfrak{I},\pi[k]}))\}
\end{aligned}$$

An *acyclic CTL$_{\mathcal{ALC}}$-TBox* $\mathcal{T}$ is a finite set of *concept definitions (CDs)* $A \equiv D$ with $A \in \mathsf{N}_\mathsf{C}$ and D a CTL$_{\mathcal{ALC}}$ concept, such that (1) no two CDs have the same left-hand side, and (2) there are no CDs $A_1 \equiv C_1, \ldots, A_k \equiv C_k$ in $\mathcal{T}$ such that A_{i+1} occurs in C_i for $1 \leq i \leq k$, where $A_{k+1} = A_1$.

A temporal interpretation $\mathfrak{I}$ is a *model* of a concept C if $C^{\mathfrak{I},\varepsilon} \neq \emptyset$; it is a model of an acyclic TBox $\mathcal{T}$, written $\mathfrak{I} \models \mathcal{T}$, if $A^{\mathfrak{I},w} = C^{\mathfrak{I},w}$ for all $A \equiv C \in \mathcal{T}$ and $w \in W$; it is a model of a *concept inclusion* $C \sqsubseteq D$, written $\mathfrak{I} \models C \sqsubseteq D$, if $C^{\mathfrak{I},w} \subseteq D^{\mathfrak{I},w}$ for all $w \in W$.

The two main reasoning tasks we consider are concept satisfiability and subsumption. A concept C is *satisfiable* relative to an acyclic TBox $\mathcal{T}$ if there is a common model of C and $\mathcal{T}$. A concept D *subsumes* a concept C relative to an acyclic TBox $\mathcal{T}$, written $\mathcal{T} \models C \sqsubseteq D$, if $\mathfrak{I} \models C \sqsubseteq D$ for all models $\mathfrak{I}$ of $\mathcal{T}$. If $\mathcal{T}$ is empty, we write $\models C \sqsubseteq D$.

3 First Observations

We start by observing that the combination of CTL and $\mathcal{ALC}$ with rigid roles relative to empty and acyclic TBoxes is decidable and inherently nonelementary. In a nutshell, we

show the upper bounds using a variant of the quasimodel technique [11, Thm. 13.6]; the lower bound follows from the fact that satisfiability for the product modal logics $\mathsf{S4}\times\mathsf{K}$ and $\mathsf{K}\times\mathsf{K}$ is inherently nonelementary [12]. Indeed, the fragment of $\text{CTL}_{\mathcal{ALC}}$ allowing $\mathbf{E}\Diamond$ ($\mathbf{E}\bigcirc$) as the only temporal operator is a notational variant of $\mathsf{S4}\times\mathsf{K}$ ($\mathsf{K}\times\mathsf{K}$) [15].

Theorem 1. *Concept satisfiability relative to acyclic and empty TBoxes for* $\text{CTL}_{\mathcal{ALC}}$ *with rigid roles is decidable and inherently nonelementary.*

With Theorem 1 and the third column of Table 1 in mind, we particularly set as our goal the identification of elementary (ideally tractable) TDLs. To this aim, we study combinations of (fragments of) CTL with the lightweight DL $\mathcal{EL}$. $\text{CTL}_{\mathcal{EL}}$ is the fragment of $\text{CTL}_{\mathcal{ALC}}$ that disallows the constructor $\neg$ (and thus the abbreviations $C \sqcup D$, $\forall r.C$, $\mathbf{A}\Box$, ...). The standard reasoning problem for $\text{CTL}_{\mathcal{EL}}$, as for $\mathcal{EL}$, is concept subsumption since each concept and TBox are trivially satisfiable. In what follows we consider various fragments of $\text{CTL}_{\mathcal{EL}}$ obtained by restricting the available temporal operators. We denote the fragments by putting the allowed operators as a superscript. In this context, we view each of the operators $\mathbf{E}\Diamond$, $\mathbf{A}\Box$ as primitive instead of as an abbreviation.

In order to keep the presentation of our main results accessible, in Sections 5-6, we concentrate on the case where only rigid role names and local concept names are present. Later on, in Section 7, we explain how to deal with the general case.

4 $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$ relative to the Empty TBox

We begin by investigating the complexity of subsumption relative to the empty TBox for a TDL whose subsumption relative to general TBoxes is undecidable: $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$.

Theorem 2. *Concept subsumption relative to the empty TBox is* coNP*-complete for* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$ *with rigid roles and in* PTIME *for* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ *and* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ *with rigid roles.*

coNP-hardness is obtained by embedding $\mathcal{EL}$ plus transitive closure into $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$; the jump in complexity comes from the ability to express disjunctions, e.g., $\models \mathbf{E}\Diamond C \sqsubseteq C \sqcup \mathbf{E}\bigcirc\mathbf{E}\Diamond C$. We next explain coNP-membership; the PTIME results are a byproduct and improved later.

We proceed in two steps: first we provide a characterization of $\models C \sqsubseteq D$ where C is an $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-concept and D an $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$-concept. Next we generalize this characterization to $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$-concepts C.

Given a $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-concept C, the *description tree* $t_C = (V_C, L_C, E_C)$ *for* C is a labeled graph corresponding to C's syntax tree; we denote its *root* by x_C. For example, if $C = \mathbf{E}\bigcirc(\exists r.A \sqcap \exists s.B)$, then t_C is given in Figure 1, left.

For plain $\mathcal{EL}$, we have $\models C \sqsubseteq D$ if and only if there is a homomorphism from t_D to t_C, which can be tested in polynomial time [8]. This criterion cannot directly be transferred to $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ because t_C does not explicitly represent all pairs of worlds and domain elements whose existence is implied by t_C, e.g., for $\models \mathbf{E}\bigcirc\exists r.A \sqsubseteq \exists r.\mathbf{E}\bigcirc A$ with r rigid, there is no homomorphism from t_D to t_C. We overcome this problem by transforming t_C into a *canonical model* $\mathfrak{I}_C$ of C, i.e., (1) its distinguished root is an instance of C and (2) $\mathfrak{I}_C$ homomorphically embeds into every model of C. The

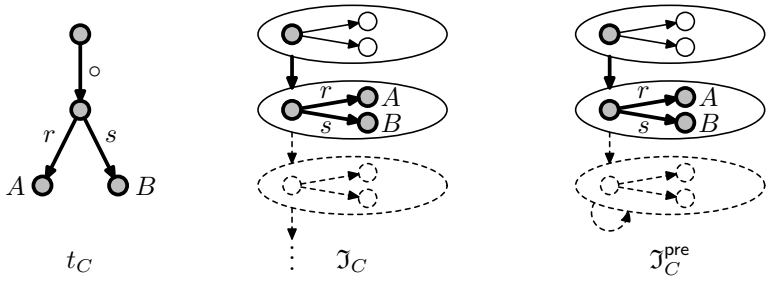


Fig. 1. Description tree t_C, canonical model $\mathfrak{I}_C$, finite representation $\mathfrak{I}_C^{\mathsf{pre}}$ for $C = \mathbf{E}\bigcirc(\exists r.A \sqcap \exists s.B)$

construction of $\mathfrak{I}_C$ from t_C is straightforward: for every node with an incoming $\bigcirc$-edge (r-edge, r being a role) create a fresh world (domain element); for the root x_C create *both* a world and domain element. The temporal relation and the interpretation of r and concept names is read off E_C and L_C. To transform (W, R) into an infinite tree, we add an infinite path of fresh worlds to every world without R-successor. The canonical model for the above concept C is shown in Fig. 1, center; the infinite path of worlds is dashed.

From (1), (2), and the preservation properties of homomorphisms, we obtain:

Lemma 3. *For all* $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$*-concepts* C *and all* $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$*-concepts* D*, we have* $\models C \sqsubseteq D$ *if and only if* $x_C \in D^{\mathfrak{I}_C, x_C}$.

Now $x_C \in D^{\mathfrak{I}_C, x_C}$ can be verified by model-checking D in world x_C and element x_C of $\mathfrak{I}_C^{\mathsf{pre}}$, which is the polynomial-sized modification of $\mathfrak{I}$ where the lastly added infinite path of worlds is replaced by a single loop, see Fig. 1, right. Since $\mathfrak{I}_C$ is the unraveling of $\mathfrak{I}_C^{\mathsf{pre}}$ into the temporal dimension, $\mathfrak{I}_C$ and $\mathfrak{I}_C^{\mathsf{pre}}$ satisfy the same concepts in their roots. Theorem 2 for $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ thus follows. The $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ part can be obtained by representing every $\mathbf{E}\Diamond$ in C by a $\bigcirc$-edge in t_C and adapting the notion of a homomorphism.

For $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$, we use expansion vectors introduced in [16], applied to the temporal dimension. Let C be a $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$-concept with n occurrences of $\mathbf{E}\Diamond$. An *expansion vector for* C is an n-tuple $U = (u_1, \dots, u_n)$ of integers $u_i \geqslant 0$. Intuitively, U fixes a specific number of temporal steps taken for each $\mathbf{E}\Diamond$ in C when constructing t_C and $\mathfrak{I}_C$. More precisely, we denote with $C[U]$ the $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-concept obtained from C by replacing the i-th occurrence of $\mathbf{E}\Diamond$ with $(\mathbf{E}\bigcirc)^{u_i}$, i.e., i times $\mathbf{E}\bigcirc$. For example, if $C = \mathbf{E}\Diamond\exists r.\mathbf{E}\Diamond(A \sqcap \mathbf{E}\bigcirc B)$ and $U = (2, 0)$, then $C[U] = \mathbf{E}\bigcirc\mathbf{E}\bigcirc\exists r.(A \sqcap \mathbf{E}\bigcirc B)$.

Let $\mathbb{U}_C^m = \{(u_1, \dots, u_n) \mid u_i \leqslant m \text{ for all } i\}$. We denote with $\mathsf{tdepth}(D)$ the nesting depth of temporal operators in D. We use expansion vectors with entries bounded by $\mathsf{tdepth}(D)$ to reduce $\not\models C \sqsubseteq D$ for $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$ to the case where C is from $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$.

Lemma 4. *For all* $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$*-concepts* C, D*, we have* $\models C \sqsubseteq D$ *if and only if* $\models C[\overline{U}] \sqsubseteq D$ *for all* $\overline{U} \in \mathbb{U}_C^{\mathsf{tdepth}(D)+1}$.

Together with Lemma 3, this yields the desired polynomial-time guess-and-check procedure for deciding $\models C \sqsubseteq D$.

5 $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ relative to Acyclic TBoxes

The results of Theorem 2 transfer to acyclic TBoxes with an exponential blowup due to unfolding [18], that is:

Corollary 5. *Concept subsumption relative to acyclic* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$*-TBoxes with rigid roles is in* coNEXPTIME.

For the subfragments $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$, we can even show polynomial complexity:

Theorem 6. *Concept subsumption relative to acyclic* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$*- and* $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$*-TBoxes with rigid roles is in* PTIME.

We first concentrate on the $\mathbf{E}\Diamond$ case and explain below how to deal with the $\mathbf{E}\bigcirc$ one. We focus w.l.o.g. on subsumption between concept *names* and assume that the input TBox is in normal form (NF), i.e., each axiom is of the shape $A \equiv A_1 \sqcap A_2$, $A \equiv \mathbf{E}\Diamond A_1$, or $A \equiv \exists r.A_1$, where $A_i \in \mathsf{N_C} \cup \{\top\}$ and $r \in \mathsf{N_R}$. As usual, a subsumption-equivalent TBox in NF can be computed in polynomial time [4]. We use CN and ROL to denote the sets of concept names and roles occurring in $\mathcal{T}$.

To prove a PTIME upper bound, we devise a completion algorithm in the style of those known for $\mathcal{EL}$ and (two-dimensional) extensions, cf. [5, 14], which build an abstract representation of the 'minimal' model of the input TBox $\mathcal{T}$ (in the sense of Horn logic). The main difficulty is that different occurrences of the same concept name in the TBox cannot all be treated uniformly (as it is the case for, say, $\mathcal{EL}$), due to the two-dimensional semantics. Instead, we have to carefully choose witnesses for $\mathbf{E}\Diamond A$ and $\exists r.A$, respectively. Our algorithm constructs a graph $G = (W, E, Q, R)$ based on a set W, a binary relation E, a mapping Q that associates with each $A \in \mathsf{CN}$ and each $w \in W$ a subset $Q(A, w) \subseteq \mathsf{CN}$, and a mapping R that associates with each rigid role $r \in \mathsf{ROL}$ a relation $R(r) \subseteq \mathsf{CN} \times W \times \mathsf{CN} \times W$. For brevity, we write $(A, w) \xrightarrow{r} (B, w')$ instead of $(A, w, B, w') \in R(r)$ and denote with E^* the reflexive, transitive closure of E.

The algorithm for deciding subsumption initializes G as follows. For all $r \in \mathsf{ROL}$, set $R(r) = \emptyset$. Set $W = \mathsf{CN} \times \mathsf{CN} \cup \{\mathbf{E}\Diamond A \mid A \in \mathsf{CN}\}$. Set $E = \{(\mathbf{E}\Diamond A, AA), (AB, A\top) \mid A, B \in \mathsf{CN}\}$. For all $A \in \mathsf{CN}$, set $Q(A, w) = \{\top, B\}$ if $w = AB$ and $Q(A, w) = \{\top\}$ otherwise.

Intuitively, the unraveling of (W, E) is the temporal tree underlying the minimal model and the mappings Q and R contain condensed information on how to interpret concepts and roles, respectively. More specifically, the data stored in $Q(A, \cdot)$ describes the temporal evolution of an instance of A. For example, $Q(A, AA)$ collects all concept names B such that $\mathcal{T} \models A \sqsubseteq B$; likewise, $Q(A, \mathbf{E}\Diamond A)$ captures everything that follows from $\mathbf{E}\Diamond A$. Finally, $Q(A, AB)$ contains concept names that are implied by B given that B appears in the temporal evolution of an instance of A, i.e., $B' \in Q(A, AB)$ implies $\mathcal{T} \models A \sqcap \mathbf{E}\Diamond B \sqsubseteq \mathbf{E}\Diamond(B \sqcap B')$.

After initialization, the algorithm extends G by applying the completion rules depicted in Figure 2 in three phases. In the first phase – also called FORWARD-phase, since definitions $A \equiv C \in \mathcal{T}$ are read as $A \sqsubseteq C$ – rules **F1**-**F3** are exhaustively applied in order to generate a fusion-like representation by adding witness-worlds and witness-existentials as demanded. Most notably, rule **F2** introduces a pointer to the structure representing the temporal evolution of an instance of B'.

F1 If $B \in Q(A, AA')$ & $B \equiv \mathbf{E}\Diamond B' \in \mathcal{T}$, then add (AA', AB') to E

F2 If $B \in Q(A, w)$ and $B \equiv \exists r.B' \in \mathcal{T}$, then set $(A, w) \xrightarrow{r} (B', B'B')$

F3 If $B \in Q(A, w)$ & $B \equiv A_1 \sqcap A_2 \in \mathcal{T}$, then add A_1, A_2 to $Q(A, w)$

C1 If $(BB, w) \in E$ and $(A, w') \xrightarrow{r} (B, BB)$, then add (w', w) to E

C2 If $(A, w) \xrightarrow{r} (B, BB)$, then

a) $(A, w') \xrightarrow{r} (B, \mathbf{E}\Diamond B)$ for all $w' \neq w$ with $(w', w) \in E^*$

b) $(A, w') \xrightarrow{r} (B, w')$ for all w' with $(w', w) \notin E^*$

B1 If $B \in Q(A, w)$, $(w', w) \in E^*$, and $A' \equiv \mathbf{E}\Diamond B \in \mathcal{T}$, then add A' to $Q(A, w')$

B2 If $A \in Q(B, w)$, $(A', w') \xrightarrow{r} (B, w)$, and $A'' \equiv \exists r.A \in \mathcal{T}$ then add A'' to $Q(A', w')$

B3 If $A_1, A_2 \in Q(B, w)$ & $A \equiv A_1 \sqcap A_2 \in \mathcal{T}$ then add A to $Q(B, w)$

Fig. 2. Completion rules

Subsequently, G is extended to conform with the constant domain assumption and reflect rigidity of roles by exhaustively applying rules **C1**, **C2**. Here read **C2** as '*if two points are connected via r in some world, then they should be connected in all worlds*.' Note that $Q(B, \mathbf{E}\Diamond B)$ is used as a representative for the entire "past" of B in part **a)**.

In the final phase, BACKWARD-completion rules **B1**-**B3** are exhaustively applied in order to respect the 'backwards'-direction of definitions, i.e., definitions $A \equiv C \in \mathcal{T}$ are read as $A \sqsupseteq C$. This separation into a FORWARD and BACKWARD phase is sanctioned by acyclicity of the TBox. In fact, one run through each phase is enough; note that no new tuples are added to E or R in the BACKWARD-phase.

The following lemma shows correctness of our algorithm.

Lemma 7. *Let $\mathcal{T}$ be an acyclic $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$-TBox in normal form. Then for all $A, B \in \mathsf{CN}$, we have $\mathcal{T} \models A \sqsubseteq B$ iff, after exhaustive rule application, $B \in Q(A, AA)$.*

To prove "$\Leftarrow$", we show that (a certain unraveling of) G "embeds" into every model of A and $\mathcal{T}$. For this purpose, we need to adapt the notion of a homomorphism to temporal interpretations and rigid roles. For "$\Rightarrow$", we construct from G a model $\mathfrak{I}$ of $\mathcal{T}$ with $d \in A^{\mathfrak{I},w} \setminus B^{\mathfrak{I},w}$ for some d, w. The algorithm runs in polynomial time: the size of the data structures W, E, and R is clearly polynomial and the mapping $Q(\cdot, \cdot)$ is extended in every rule application, so the algorithm stops after polynomially many steps.

Finally, we sketch two modifications of the algorithm such that it works for $\mathbf{E}\bigcirc$ instead of $\mathbf{E}\Diamond$. First, we have to use a non-transitive version of **B1**. Second, and a bit more subtly, we have to replace $\mathbf{E}\Diamond A \in W$ with $\mathbf{E}\bigcirc^k A$, $1 \leq k \leq |\mathcal{T}|$ to capture what is implied by $\mathbf{E}\bigcirc^k A$; more precisely, $B' \in Q(A, \mathbf{E}\bigcirc^k A)$ implies $\mathcal{T} \models \mathbf{E}\bigcirc^k A \sqsubseteq B'$, where $\mathbf{E}\bigcirc^k$ denotes $\mathbf{E}\bigcirc \cdots \mathbf{E}\bigcirc$ k times.

We next show that there is a jump in the complexity if increasing domains are considered instead of constant ones. Intuitively, this can be explained by the fact that increasing domains allow rigid roles to mimic the behaviour of the $\mathbf{A}\Box$-operator. In the next section, we show that adding $\mathbf{A}\Box$ to $\{\mathbf{E}\Diamond\}$ indeed leads to PSPACE-hardness.

Theorem 8. *Concept subsumption relative to acyclic $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$- and $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$-TBoxes with rigid roles and increasing domains is* PSPACE*-hard.*

6 $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond,\mathbf{A}\Box}$ relative to Acyclic TBoxes

We now add $\mathbf{A}\Box$ and observe an increase in complexity over acyclic TBoxes.

Theorem 9. *Concept subsumption relative to acyclic* $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond,\mathbf{A}\Box}$*-TBoxes with rigid roles is* PSPACE*-complete.*

The lower bound is obtained via a reduction from QBF validity. For the upper bound, we again consider w.l.o.g. subsumption between concept *names* and assume that the acyclic TBox is in normal form, i.e., axioms are of the shape $A \equiv A_1 \sqcap A_2$, $A \equiv \mathbf{E}\Diamond A_1$, $A \equiv \mathbf{A}\Box A_1$, or $A \equiv \exists r.A_1$, where $A_i \in \mathsf{N_C} \cup \{\top\}$ and $r \in \mathsf{N_R}$. We also restrict ourselves again to only rigid roles. CN and ROL are used as before.

In contrast to the previous section, we cannot maintain the entire minimal model in memory since the added operator $\mathbf{A}\Box$ can be used to enforce models of exponential size. Instead, we will compute all concepts implied by the input concept A (the left-hand side of the subsumption to be checked) by iteratively visiting relevant parts of the minimal model. Our main tool for doing so are *traces*.

Definition 10. *A* trace *is a tuple* (σ, E, R) *where* σ *is a sequence* $(d_0, w_0) \cdots (d_n, w_n)$ *such that for all* $0 \leq i < n$ *one of the following is true. (1)* $d_i = d_{i+1}$ *and* $(w_i, w_{i+1}) \in E$*. (2)* $w_i = w_{i+1}$ *and* $(d_i, d_{i+1}) \in R(r)$ *for some* $r \in \mathsf{ROL}$*.*

Intuitively, traces represent paths through temporal interpretations, which in each step follow either the temporal relation (Def. 10, 1) or a DL relation r (2); so, in a pair (d, w), d can be thought of as a domain element and w as a world.

Our algorithm, whose basic structure is given by Alg. 1, enumerates on input $\mathcal{T}, A, B$, in a systematic tableau-like way, all traces that *must* appear in every model of A and $\mathcal{T}$. Note that in the context of Algorithm 1 a trace is used as the basis for inducing a richer structure that conforms with the constant domain assumption and captures rigidity; see Example 11 below. The algorithm also maintains an additional mapping Q that labels each point (d, w) of the trace (and all the induced points) with a set $Q(d, w) \subseteq \mathsf{CN}$. The set $Q(d, w)$ captures all concept names that are satisfied in the minimal model at points represented by (d, w).

Algorithm 1: Subsumption in $\mathrm{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond,\mathbf{A}\Box}$

Input: Acyclic TBox $\mathcal{T}$, concept names A, B
Output: `true` if $\mathcal{T} \models A \sqsubseteq B$, `false` otherwise

$\sigma := (d_0, w_0)$; $Q(d_0, w_0) := \{A, \top\}$;
$E := \emptyset$; $R(r) := \emptyset$ for all $r \in \mathsf{ROL}$;
`expand`(σ, E, R);
return `true` if $B \in Q(d_0, w_0)$, `false` otherwise;
procedure `expand` (σ, E, R) **:**
  `complete` (σ, E, R, Q);
  if (σ, Q) *is periodic at* (i, j) **then**
    add (w_{j-1}, w_i) to E;
    `truncate`;
    `complete` (σ, E, R, Q);
    return;
  $(d, w) :=$ last element of σ;
  foreach $A \in Q(d, w)$ *with* $A \equiv \exists r.B \in \mathcal{T}$ **do**
    $Q(d', w) = \{B, \top\}$ for a fresh d';
    add (d, d') to $R(r)$;
    `expand` $(\sigma \cdot (d', w), E, R)$;
  foreach $A \in Q(d, w)$ *with* $A \equiv \mathbf{E}\Diamond B \in \mathcal{T}$ **do**
    $Q(d, w') = \{B, \top\}$ for a fresh w';
    add (w, w') to E;
    `expand` $(\sigma \cdot (d, w'), E, R)$;

R1 If $A \equiv A_1 \sqcap A_2 \in \mathcal{T}$ and $A \in Q_*(\cdot)$, then add A_1, A_2 to $Q_*(\cdot)$

R2 If $A \equiv A_1 \sqcap A_2 \in \mathcal{T}$ and $A_1, A_2 \in Q_*(\cdot)$, then add A to $Q_*(\cdot)$

R3 If $(d, d') \in R(r)$, $B \in Q(d', w)$, $A \equiv \exists r.B \in \mathcal{T}$, then add A to $Q(d, w)$

R4 If $B \in Q(d, w)$, $(w', w) \in E^*$, $A \equiv \mathbf{E}\Diamond B \in \mathcal{T}$, then add A to $Q(d, w')$

R5 If $B \in Q(d, w)$, $(w, w') \in E^*$, $B \equiv \mathbf{A}\Box A \in \mathcal{T}$, then add B, A to $Q(d, w')$

R6 If $(d, d') \in R(r)$, $B \in Q_{\mathsf{cert}}(d')$, $A \equiv \exists r.B \in \mathcal{T}$, then add A to $Q_{\mathsf{cert}}(d)$

R7 If $B \in Q_{\mathsf{cert}}(d)$, $A \equiv \mathbf{A}\Box B \in \mathcal{T}$, then add A to $Q_{\mathsf{cert}}(d)$

R8 If $B \in Q_{\mathsf{cert}}(d)$, then add B to $Q(d, w)$ for all w

---

R9 If $B \in Q_{\mathbf{A}\Box}(d, w)$, $A \equiv \mathbf{A}\Box B \in \mathcal{T}$, add A to $Q(d, w)$

R10 If $A \in Q(d, w)$, $A \equiv \mathbf{A}\Box B \in \mathcal{T}$, add A, B to $Q_{\mathbf{A}\Box}(d, w)$

R11 If $(d, d') \in R(r)$, $B \in Q_{\mathbf{A}\Box}(d', w)$, $A \equiv \exists r.B \in \mathcal{T}$, then add A to $Q_{\mathbf{A}\Box}(d, w)$

R12 If $A \in Q_{\mathbf{A}\Box}(d, w)$, $A \equiv \mathbf{E}\Diamond B \in \mathcal{T}$, w' added due to $A \in Q(d, w)$ in Line 18, $B' \in Q(d, w')$, $A' \equiv \mathbf{E}\Diamond B' \in \mathcal{T}$, then add A' to $Q_{\mathbf{A}\Box}(d, w)$

Fig. 3. Saturation rules. In **R1**, **R2** the set $Q_*(\cdot)$ ranges over all $Q(d, w)$, $Q_{\mathsf{cert}}(d)$, and $Q_{\mathbf{A}\Box}(d, w)$.

The basics of Algorithm 1 are the following. In Lines 1 and 2, it creates a trace consisting of a single point representing A and initializes the necessary data structures. In Line 3, the systematic expansion is set off. When that is finished, the algorithm just returns whether or not B (the right-hand of the subsumption) has been added during the expansion. As for the `expand` procedure:

- in Line 6 and 10, the algorithm updates the mapping Q;
- Line 7 contains some termination condition; and finally,
- the loops in Lines 13 & 17 enumerate all $\exists r.B$ and $\mathbf{E}\Diamond B$ that appear in the set $Q(d, w)$ of the last trace element and expand the trace to witness these concepts.

This basic description of the algorithm leaves open several points: *(i)* the precise behavior of the subroutine `complete`, *(ii)* when a trace is *periodic*, and *(iii)* what happens inside the `truncate` command in Line 9. Let us start with describing the subroutine `complete`. It uses additional mappings $Q_{\mathsf{cert}}(d) \subseteq \mathsf{CN}$ and $Q_{\mathbf{A}\Box}(d, w) \subseteq \mathsf{CN}$, which intuitively contain all the concept names that d satisfies *certainly*, i.e., in all worlds, and starting from world w, respectively. It proceeds in two steps. (1) Initialize undefined $Q(d, w)$ and $Q_{\mathsf{cert}}(d)$ with $\{\top\}$, and undefined $Q_{\mathbf{A}\Box}(d, w)$ with $Q_{\mathsf{cert}}(d)$. (2) Apply rules **R1**-**R12** in Figure 3 to $Q(\cdot)$, $Q_{\mathsf{cert}}(\cdot)$ and $Q_{\mathbf{A}\Box}(\cdot)$.

The number of rules is indeed scarily high; however, they can be divided into four digestible groups: **R1** and **R2** are used to ensure that all sets Q_* are closed under conjunction; **R3**-**R5** are used to complete $Q(\cdot)$. Note that **R1**-**R4** are already known from the algorithm of the previous section. Furthermore, **R6**-**R8** are used to deal with $Q_{\mathsf{cert}}(\cdot)$; and **R9**-**R12** to update $Q_{\mathbf{A}\Box}(\cdot)$. As an example of the interplay between the different mappings take **R9**: If B is certain for d starting in world w and $A \equiv \mathbf{A}\Box B$, then we also know that d satisfies A in w; and **R11** for the interplay between temporal operators and rigid roles: indeed, for r rigid, $\models \exists r.\mathbf{A}\Box B \sqsubseteq \mathbf{A}\Box\exists r.B$.

Example 11. Let $\mathcal{T} = \{A \equiv \mathbf{E}\Diamond A_1,\ A_1 \equiv \exists r.B,\ B \equiv \mathbf{E}\Diamond A_1\}$ be the input TBox; and $\mathcal{T} \models A \sqsubseteq A_1$ is to be checked. Figure 4 (left) shows the trace initiated at (d_0, w_0) with $Q(d_0, w_0) = \{\top, A\}$, and further expanded in Lines 13 and 17. The trace, as mentioned above, induces a richer structure, reflecting rigid roles and the constant domain assumption; see Fig. 4 (center). This richer structure is then `completed` to properly enrich the types $Q(d, w)$ of each element. In particular, during completion, further concept names are added to the corresponding types (Fig. 4, right). One can now easily see that $\mathcal{T} \models A \sqsubseteq A_1$ indeed holds. Furthermore, note that $\mathcal{T} \not\models A \sqsubseteq A_1$, if r is local or increasing domains are assumed. This is the case since, in both cases, the r-connection is not necessarily present in the 'root world'.

For the termination condition in Line 7, we take the following definition of periodicity.

Definition 12. *A trace (σ, E, R) together with a mapping Q is called* periodic at (i, j) *if $\sigma = (d_0, w_0) \cdots (d_n, w_n)$, $i < j$, $d_i = d_j = d_n$, and $Q(d_i, w_i) = Q(d_j, w_j)$.*

This means that during the evolution of element $d = d_i = d_j$, we find two different worlds w_i, w_j such that d has the same type in w_i and w_j. We can stop expanding worlds appearing after w_j since their behavior is already captured by the successors of w_i. If a trace periodic at (i, j) is found, we add an edge (w_{j-1}, w_i) to E reflecting the periodic behavior, see Line 8. Then, in `truncate`, the trace is shortened to $(d_0, w_0) \cdots (d_{j-1}, w_{j-1})$ and the relations E and $R(r)$, $r \in \mathsf{ROL}$, and the mappings $Q, Q_{\mathsf{cert}}, Q_{\mathbf{A}\Box}$ are restricted to those d and w that appear in the shortened trace.

Lemma 13. *On every input $\mathcal{T}, A, B$, Alg. 1 terminates and returns* `true` *iff $\mathcal{T} \models A \sqsubseteq B$.*

For termination, consider a trace with suffix $(d, w_1) \cdots (d, w_n)$ and let $A_1, \ldots, A_n$ be the concept names such that $\mathbf{E}\Diamond A_i$ lead to w_i, see Line 17 of Alg. 1. It is not difficult to show that if $A_i = A_j$ for $i < j$, then $Q(d, w_i) \subseteq Q(d, w_j)$ after application of `complete`. Since $Q(d, w) \subseteq \mathsf{CN}$, there are no infinite (strictly) increasing sequences. Hence, the expansion in Lines 17ff. will not indefinitely be applied. Also, the expansion in Lines 13ff. stops due to acyclicity of the TBox. Together, this guarantees termination.

Correctness is shown similar to Lemma 7. For "$\Rightarrow$", we show that every trace together with the labeling so far computed in Q can be embedded into every model of A and $\mathcal{T}$. For "$\Leftarrow$", we present a model of $\mathcal{T}$ witnessing $\mathcal{T} \not\models A \sqsubseteq B$.

We finish the proof of Theorem 9 by noting that the termination argument indeed yields a polynomial bound on the length of the traces encountered by Alg. 1.

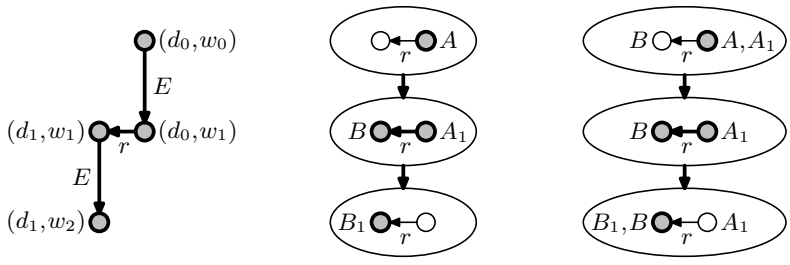


Fig. 4. An example trace and the induced structure

7 Local Roles and Rigid Concepts

One can easily extend the above algorithms so as to deal with local roles. In fact, e.g., in Section 5 only **B4** below needs to be added to the BACKWARD-rules in Figure 3. Note that **F2** is only applied to rigid roles and **C2** is therefore not applied to local ones. Clearly, the algorithm in Section 6 can be extended with a similar rule.

B4 If $A \in Q(B,w)$, $A \equiv \exists r.A'$, $B' \in Q(A', A'A')$, $B'' \equiv \exists r.B' \in \mathcal{T}$, add B'' to $Q(B,w)$
RC If $B \in Q(A,w)$, $B \in \mathsf{CN}_{\mathsf{rig}}$, then add B to $Q(A,w')$, $\forall w' \in W$
R13 If $B \in Q(d,w)$ or $B \in Q_{\mathbf{A}\Box}(d,w)$ & $B \in \mathsf{CN}_{\mathsf{rig}}$, then add B to $Q_{\mathsf{cert}}(d)$

A rigid concept has a constant interpretation over time. In the first example of Section 1, the concept Disorder should be rigid because we regard medical knowledge as static. PatientWithDisorder should be local because a disease history has begin and end.

In the presence of general TBoxes, rigid concepts can be simulated by rigid roles: replace each rigid concept name A with $\exists r_A.\top$, where r_A is a fresh rigid role. Alas, this simulation does not work in the context of acyclic TBoxes: the result of replacing A with $\exists r_A.\top$ in a CD $A \equiv D$ is no longer a CD. Still, our algorithms can be extended, without increasing the complexity, to consider rigid concepts: e.g., the algorithm in Section 5 can be extended by adding **RC** above to the FORWARD *and* BACKWARD rules – $\mathsf{CN}_{\mathsf{rig}}$ denotes the set of rigid concepts occurring in the input TBox. Note that the intermediate phase remains the same, i.e., rules **C1** and **C2** are neither extended nor modified.

Rigid concepts can analogously be included in Section 6 by adding a new rule **R13** above (recall: intuitively, $Q_{\mathsf{cert}}(d)$ contains the concepts that hold for d in any world).

In the empty TBox case rigid roles can again simulate rigid concepts as above.

8 Conclusions and Future Work

We have initiated the investigation of TDLs based on $\mathcal{EL}$ allowing for rigid roles and restricted TBoxes. We indeed achieved our main goal: we identified fragments of the combination of CTL and $\mathcal{EL}$ that have elementary, some even polynomial, complexity.

One important conclusion is that the use of acyclic TBoxes, instead of general ones, allows to design TDLs based on $\mathcal{EL}$ with dramatically better complexity than the $\mathcal{ALC}$ variant; e.g., for the fragment allowing only $\mathbf{E}\bigcirc$ the complexity drops from nonelementary to PTIME. As an important byproduct, the studied fragments of $\text{CTL}_{\mathcal{EL}}$ can be seen as *positive fragments* of product modal logics with elementary complexity, e.g., implication for the positive fragment of $\mathsf{K}\times\mathsf{K}$ is in PTIME.

Next, we plan to look at more expressive fragments of $\text{CTL}_{\mathcal{EL}}$ or at classical (cyclic) TBoxes, e.g., consider *non-convex* fragments, such as $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}^{\diamond}}$, with (a)cyclic TBoxes. We plan to incorporate temporal roles, too. It is also worth exploring how restricting TBoxes can help tame other TDLs with bad computational behavior over general TBoxes, such as TDLs based on LTL or the μ-calculus. We believe that the LTL case is technically easier than ours since it does not have the extra '$\frac{1}{2}$-dimension' introduced by branching.

Acknowledgements The first author was supported by the M8 PostDoc Initiative project TS-OBDA and the second one by the DFG project LU1417/1-1. We thank the anonymous reviewers for their detailed and constructive suggestions.

References

1. Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: Proc. TIME (2007)
2. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. ACM Trans. Comput. Log. 15(3), 25 (2014)
3. Artale, A., Lutz, C., Toman, D.: A description logic of change. In: Proc. IJCAI (2007)
4. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. IJCAI (2003)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI (2005)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
7. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. In: Proc. KR (2008)
8. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Proc. IJCAI (1999)
9. Bodenreider, O., Zhang, S.: Comparing the representation of anatomy in the FMA and SNOMED CT. In: Proc. AMIA (2006)
10. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
11. Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-dimensional modal logics: theory and applications, Studies in Logic, vol. 148. Elsevier (2003)
12. Göller, S., Jung, J.C., Lohrey, M.: The complexity of decomposing modal and first-order theories. ACM Trans. Comput. Log. 16(1), 9:1–9:43 (2015)
13. Gutiérrez-Basulto, V., Jung, J.C., Lutz, C.: Complexity of branching temporal description logics. In: Proc. ECAI (2012)
14. Gutiérrez-Basulto, V., Jung, J.C., Lutz, C., Schröder, L.: A closer look at the probabilistic description logic Prob-$\mathcal{EL}$. In: Proc. AAAI (2011)
15. Gutiérrez-Basulto, V., Jung, J.C., Schneider, T.: Lightweight description logics and branching time: a troublesome marriage. In: Proc. KR (2014)
16. Haase, C., Lutz, C.: Complexity of subsumption in the $\mathcal{EL}$ family of description logics: Acyclic and cyclic TBoxes. In: Proc. ECAI (2008)
17. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Proc. TIME (2008)
18. Nebel, B.: Terminological reasoning is inherently intractable. Artif. Intell. 43(2), 235–249 (1990)
19. Schild, K.: Combining terminological logics with tense logic. In: Proc. EPIA (1993)
20. The Gene Ontology Consortium: Gene ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)

Schema.org as a Description Logic

Andre Hernich[1], Carsten Lutz[2], Ana Ozaki[1] and Frank Wolter[1]

[1] University of Liverpool, UK
[2] University of Bremen, Germany

Abstract. Schema.org is an initiative by the major search engine providers Bing, Google, Yahoo!, and Yandex that provides a collection of ontologies which webmasters can use to mark up their pages. Schema.org comes without a formal language definition and without a clear semantics. We formalize the language of Schema.org as a Description Logic (DL) and study the complexity of querying data using (unions of) conjunctive queries in the presence of ontologies formulated in this DL. In particular, we consider rewritability into FO queries and into datalog programs and investigate the possibility of classifying the data complexity of ontology-mediated queries.

1 Introduction

The Schema.org initiative was launched in 2011 and is supported today by Bing, Google, Yahoo!, and Yandex. In the spirit of the Semantic Web, it provides a collection of ontologies that establish a standard vocabulary to mark up website content with metadata about itself (https://schema.org/). In particular, web content that is generated from structured data as found in relational databases is often difficult to recover for search engines and Schema.org markup elegantly solves this problem. The markup is used by search engines to more precisely identify relevant pages, to provide richer search results, and to enable new applications. Schema.org is experiencing very rapid adoption and is used today by more than 15 million webpages including all major ones Guha [2013].

Schema.org does neither formally specify the language in which its ontologies are formulated nor does it provide a formal semantics for the published ontologies. However, the provided ontologies are extended and updated frequently and follow an underlying language pattern. This pattern and its meaning is described informally in natural language. Schema.org adopts a class-centric representation enriched with binary relations and datatypes, similar in spirit to description logics (DLs) and to the OWL family of ontology languages; the current version includes 622 classes and 891 binary relations. Partial translations into RDF and into OWL are provided by the linked data community. Based on the informal descriptions at https://schema.org/ and on the mentioned translations, Patel-Schneider [2014] develops an ontology language for Schema.org with a formal syntax and semantics that, apart from some details, can be regarded as a fragment of OWL DL.

In this paper, we abstract slightly further and view the Schema.org ontology language as a description logic, in line with the formalization by Patel-Schneider. Thus, what Schema.org calls a *type* becomes a concept name and a *property* becomes a role name. The main characteristics of the resulting 'Schema.org DL' are that (i) the language is very

restricted, allowing only inclusions between concept and role names, domain and range restrictions, nominals, and datatypes; (ii) ranges and domains of roles can be restricted to *disjunctions* of concept names (possibly mixed with datatypes in range restrictions) and nominals are used in 'one-of enumerations' whose semantics also involves disjunction. While Point (i) suggests that the Schema.org DL is closely related to the tractable profiles of OWL2, because of Point (ii) it does actually not fall into any of them. There is also a close connection to the DL-Lite family of DLs Calvanese *et al.* [2007], and in particular to the DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ variant Artale *et al.* [2009]. However, DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ admits existential restriction, negation, conjunction, and free use of disjunction whereas the Schema.org DL allows no existential quantification and includes nominals and datatypes. We use the term *schema.org-ontology* to refer to ontologies formulated in the Schema.org language; in contrast, 'Schema.org 2015' refers to the concrete collection of ontologies provided at https://schema.org/ as of end of April, 2015.

Our main aim is to investigate the complexity of *querying data in the presence of schema.org-ontologies*, where the data is the markup that was extracted from web-pages. While answering queries over such data is the main reasoning task that arises in Schema.org applications and the Schema.org initiative specifies a format for the data in terms of so-called *items*, no information at all is given on how the data is queried (or used otherwise). We consider conjunctive queries (CQs) and unions of conjunctive queries (UCQ), a basic querying mechanism that is ubiquitous in relational database systems and research, and that also can be viewed as a core of the Semantic Web query language SPARQL. In particular, we also consider CQs and UCQs without quantified variables since these are not allowed in the relevant SPARQL entailment regimes Glimm and Krötzsch [2010]. We view a pair $(\mathcal{O}, q)$ that consists of a schema.org-ontology and an actual query as a compound query called an *ontology-mediated query (OMQ)*.

We start with the observation that evaluating OMQs is intractable in general, namely Π^p_2-complete in combined complexity and coNP-complete in data complexity. In the main part of the paper, we therefore aim (i) to identify large and practically useful classes of OMQs with lower computational complexity (both combined and data complexity), and (ii) to explore the situation in much more detail to see whether we can obtain a *full classification* of each schema.org ontology or each OMQ according to its data complexity. While the utility of aim (i) is obvious, we note that aim (ii) is also most useful from a user's perspective as it clarifies the complexity of every *concrete ontology or OMQ* that might be used in an actual application. Apart from classical tractability (that is, PTime), we are particularly interested in the rewritability of OMQs into first-order (FO) queries (actually: UCQs) and into datalog programs. One reason is that this allows to implement querying based on relational database systems and datalog engines, taking advantage of those systems' efficiency and maturity. Another reason is that there is significant research on how to efficiently answer UCQs and datalog queries in cluster computing models such as MapReduce Afrati and Ullman [2011, 2012], which is rather natural when processing web-scale data.

For both aims (i) and (ii) above, we start with analyzing *basic* schema.org ontologies in which enumeration definitions ('one of' expressions) and datatypes are disallowed. Regarding aim (i), we show that all OMQs which consist of a basic schema.org-ontology and a CQ of qvar-size two (the connected components that consist exclusively of quantified variables have size at most two) are datalog-rewritable in polynomial time and can

be evaluated in PTime in combined complexity. This result complements results about datalog-rewritability of OMQs for DLs with disjunction in Grau *et al.* [2013]; Kaminski *et al.* [2014b,a]. We establish the same results for OMQs that consist of an unrestricted schema.org-ontology and CQs without quantified variables.

Regarding aim (ii), we start with classifying each single schema.org-ontology $\mathcal{O}$ according to the data complexity of *all* OMQs $(\mathcal{O}, q)$ with q a UCQ. We establish a dichotomy between AC^0 and coNP in the sense that for each ontology $\mathcal{O}$ either all these OMQs are in AC^0 or there is one OMQ that is coNP-hard. The dichotomy comes with a transparent syntactic characterization and is decidable in PTIME. Though beautiful, the dichotomy is of limited use in practice since most interesting ontologies are of the intractable kind.

Therefore, we also consider an even more fine-grained classification on the level of OMQs, establishing a useful connection to constraint satisfaction problems (CSPs) in the spirit of Bienvenu *et al.* [2014b]. It turns out that even for basic schema.org-ontologies and for ontologies that consist exclusively of enumeration definitions, a complexity classification of OMQs implies a solution to the dichotomy conjecture for CSPs, which is a famous open problem Feder and Vardi [1998]; Bulatov [2011]. However, the CSP connection can also be used to obtain powerful positive results. In particular, we show that it is decidable in NEXPTIME whether an OMQ based on a schema.org-ontology and a restricted form of UCQ is FO-rewritable and, respectively, datalog-rewritable. We also establish a PSpace lower bound for this problem.

2 Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite and mutually disjoint sets of *concept names*, *role names*, and *individual names*. Throughout the paper, concepts names will be denoted by $A, B, C, \ldots$, role names by $r, s, t, \ldots$, and individual names by $a, b, c, \ldots$.

A schema.org-ontology consists of concept inclusions of different forms, role inclusions, and enumeration definitions. A *concept inclusion* takes the form $A \sqsubseteq B$ (*atomic concept inclusion*), $\mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ (*range restriction*), or $\mathsf{dom}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ (*domain restriction*). A *role inclusion* takes the form $r \sqsubseteq s$.

Example 1. The following are examples of concept inclusions and role inclusions (last line) in Schema.org 2015:

$$\begin{aligned} \mathsf{Movie} &\sqsubseteq \mathsf{CreativeWork} \\ \mathsf{ran}(\mathsf{musicBy}) &\sqsubseteq \mathsf{Person} \sqcup \mathsf{MusicGroup} \\ \mathsf{dom}(\mathsf{musicBy}) &\sqsubseteq \mathsf{Episode} \sqcup \mathsf{Movie} \sqcup \mathsf{RadioSeries} \sqcup \mathsf{TVSeries} \\ \mathsf{sibling} &\sqsubseteq \mathsf{relatedTo} \end{aligned}$$

We now define enumeration definitions. Fix a set $\mathsf{N_E} \subseteq \mathsf{N_I}$ of *enumeration individuals* such that both $\mathsf{N_E}$ and $\mathsf{N_I} \setminus \mathsf{N_E}$ are infinite. An *enumeration definition* takes the form $A \equiv \{a_1, \ldots, a_n\}$ with $A \in \mathsf{N_C}$ and $a_1, \ldots, a_n \in \mathsf{N_E}$.

Example 2. An example of an enumeration definition in Schema.org 2015 is $\mathsf{Booktype} \equiv \{\mathsf{ebook}, \mathsf{hardcover}, \mathsf{paperback}\}$.

A *datatype* $\mathcal{D} = (D, \Delta^{\mathcal{D}})$ consists of a *datatype name* D and a non-empty set of *data values* $\Delta^{\mathcal{D}}$. Examples of datatypes in Schema.org 2015 are Boolean, Integer, and Text. We assume that datatype names and data values are distinct from the symbols in $\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}$ and that there is an arbitrary but fixed set DT of datatypes such that $\Delta^{\mathcal{D}_1} \cap \Delta^{\mathcal{D}_2} = \emptyset$ for all $\mathcal{D}_1 \neq \mathcal{D}_2 \in \mathsf{DT}$.

To accommodate datatypes in ontologies, we generalize range restrictions to *range restrictions with datatypes*, which are inclusions of the form $\mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ with $A_1, \ldots, A_n$ concept names or datatype names from DT.

Example 3. An example of a range restriction with datatypes in Schema.org 2015 is

$$\mathsf{ran}(\mathsf{acceptsReservation}) \sqsubseteq \mathsf{Boolean} \sqcup \mathsf{Text}$$

A *schema.org-ontology* $\mathcal{O}$ is a finite set of concept inclusions (including range restrictions with datatypes), role inclusions, and enumeration definitions. We denote by $\mathsf{N_C}(\mathcal{O})$ the set of concept names in $\mathcal{O}$, by $\mathsf{N_R}(\mathcal{O})$ the set of role names in $\mathcal{O}$, and by $\mathsf{N_E}(\mathcal{O})$ the set of enumeration individuals in $\mathcal{O}$.

A *data instance* $\mathcal{A}$ is a finite set of *concept assertions* $A(a)$ where $A \in \mathsf{N_C}$ and $a \in \mathsf{N_I}$; and *role assertions* $r(a, b)$ where $r \in \mathsf{N_R}$, $a \in \mathsf{N_I}$ and $b \in \mathsf{N_I} \cup \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$. We say that $\mathcal{A}$ is a data instance *for the ontology* $\mathcal{O}$ if $\mathcal{A}$ contains no enumeration individuals except those in $\mathsf{N_E}(\mathcal{O})$. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of all individuals (including datatype elements) in $\mathcal{A}$.

Example 4. Examples for assertions are $\mathsf{Movie}(a)$, $\mathsf{name}(a, \text{'avatar'})$, $\mathsf{director}(a, b)$, $\mathsf{name}(b, \text{'Cam'})$.

Let $\mathcal{O}$ be a schema.org-ontology and $\mathcal{A}$ a data instance for $\mathcal{O}$. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *for* $\mathcal{O}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ disjoint from $\bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$ and with $\Delta^{\mathcal{I}} \cap \mathsf{N_E} = \mathsf{N_E}(\mathcal{O})$, and a function $\cdot^{\mathcal{I}}$ that maps

- every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$,
- every role name r to a subset $r^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I},\mathsf{DT}}$, where $\Delta^{\mathcal{I},\mathsf{DT}} = \Delta^{\mathcal{I}} \cup \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$;
- every individual name $a \in (\mathsf{N_I} \setminus \mathsf{N_E}) \cup \mathsf{N_E}(\mathcal{O})$ to some $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} = a$ for all $a \in \mathsf{N_E}(\mathcal{O})$.

Note that we make the standard name assumption (and, therefore, unique name assumption) for individuals in $\mathsf{N_E}$. Individual names from $\mathsf{N_E}$ that do not occur in $\mathcal{O}$ (and thus not in $\mathcal{A}$) are not interpreted by $\mathcal{I}$ to avoid enforcing infinite domains.

For an interpretation $\mathcal{I}$, set $\mathsf{dom}(r)^{\mathcal{I}} = \{d \mid (d, d') \in r^{\mathcal{I}}\}$ and $\mathsf{ran}(r)^{\mathcal{I}} = \{d' \mid (d, d') \in r^{\mathcal{I}}\}$. To achieve uniform notation, set $D^{\mathcal{I}} = \Delta^{\mathcal{D}}$ for every datatype $(D, \Delta^{\mathcal{D}})$ in DT and $d^{\mathcal{I}} = d$ for every $d \in \Delta^{\mathcal{D}}$, $\mathcal{D} \in \mathsf{DT}$. For concept or datatype names $A_1, \ldots, A_n$, set $(A_1 \sqcup \cdots \sqcup A_n)^{\mathcal{I}} = A_1^{\mathcal{I}} \cup \cdots \cup A_n^{\mathcal{I}}$. An interpretation $\mathcal{I}$ for an ontology $\mathcal{O}$ *satisfies* a (concept or role) inclusion $X_1 \sqsubseteq X_2 \in \mathcal{O}$ if $X_1^{\mathcal{I}} \subseteq X_2^{\mathcal{I}}$, an enumeration definition $A \equiv \{a_1, \ldots, a_n\}$ if $A^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$, a concept assertion $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and a role assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. Satisfaction of any of these objects is denoted with "$\models$", as in $\mathcal{I} \models X_1 \sqsubseteq X_2$ or $\mathcal{I} \models A(a)$.

An interpretation $\mathcal{I}$ for $\mathcal{O}$ is a *model* of $\mathcal{O}$ if it satisfies all inclusions and definitions in $\mathcal{O}$ and a *model* of a data instance $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$. We say that $\mathcal{A}$ is *satisfiable* w.r.t. $\mathcal{O}$ if $\mathcal{O}$ and $\mathcal{A}$ have a common model. Let α be a concept or role inclusion, or an enumeration definition. We say that α *follows from* $\mathcal{O}$, in symbols $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α.

We introduce the query languages considered in this paper. A *term* t is either a member of $\mathsf{N_I} \cup \bigcup_{\mathcal{D} \in \mathsf{DT}} \Delta^{\mathcal{D}}$ or an *individual variable* taken from an infinite set $\mathsf{N_V}$ of such variables. A *first-order query (FOQ)* consist of a (domain-independent) first-order formula $\varphi(\boldsymbol{x})$ that uses unary predicates from $\mathsf{N_C} \cup \{D \mid (D, \mathcal{D}) \in \mathsf{DT}\}$, binary predicates from $\mathsf{N_R}$, and only terms as introduced above. The unary datatype predicates are built-ins that identify the elements of the respective datatype. We call $\boldsymbol{x}$ the *answer variables* of $\varphi(\boldsymbol{x})$, the remaining variables are called *quantified*. A query without answer variables is *Boolean*. A *conjunctive query (CQ)* is a FOQ of the form $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ where $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms such that every answer variable x occurs in an atom that uses a symbol from $\mathsf{N_C} \cup \mathsf{N_R}$, that is, an answer variable x is not allowed to occur exclusively in atoms of the form $D(x)$ with D a datatype name (to ensure domain independence). A *union of conjunctive queries (UCQ)* is a disjunction of CQs. A CQ q can be regarded as a directed graph G^q with vertices $\{t \mid t \text{ term in } q\}$ and edges $\{(t, t') \mid r(t, t') \text{ in } q\}$. If G^q is acyclic and $r(t_1, t_2), s(t_1, t_2) \in q$ implies $r = s$, then q is an *acyclic CQ*. A UCQ is *acyclic* if all CQs in it are.

We are interested in querying data instances $\mathcal{A}$ using a UCQ $q(\boldsymbol{x})$ taking into account the knowledge provided by an ontology $\mathcal{O}$. A *certain answer to* $q(\boldsymbol{x})$ *in* $\mathcal{A}$ *under* $\mathcal{O}$ is a tuple $\boldsymbol{a}$ of elements of $\mathsf{Ind}(\mathcal{A})$ of the same length as $\boldsymbol{x}$ such that for every model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$, we have $\mathcal{I} \models q[\boldsymbol{a}]$. In this case, we write $\mathcal{O}, \mathcal{A} \models q(\boldsymbol{a})$.

Query evaluation is the problem to decide whether $\mathcal{O}, \mathcal{A} \models q(\boldsymbol{a})$. For the combined complexity of this problem, all of $\mathcal{O}, \mathcal{A}, q$, and $\boldsymbol{a}$ are the input. For the data complexity, only $\mathcal{A}$ and $\boldsymbol{a}$ are the input. It often makes sense to combine the ontology $\mathcal{O}$ and actual query $q(\boldsymbol{x})$ into an *ontology-mediated query (OMQ)* $Q = (\mathcal{O}, q(\boldsymbol{x}))$, which can be thought of as a compound overall query. The following can be shown using techniques similar to those in Eiter *et al.* [1997]; Bienvenu *et al.* [2014b].

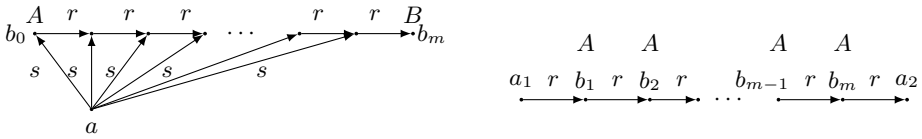
Theorem 1. *Query evaluation of CQs and UCQs under schema.org-ontologies is* Π_2^p*-complete in combined complexity. In data complexity, each OMQ* $(\mathcal{O}, q)$ *from this class can be evaluated in* coNP*; moreover, there is such a OMQ (with* q *a CQ) that is* coNP*-complete in data complexity.*

An OMQ $(\mathcal{O}, q(\boldsymbol{x}))$ is *FO-rewritable* if there exists a FOQ $Q(\boldsymbol{x})$ (called an *FO-rewriting* of $(\mathcal{O}, q(\boldsymbol{x}))$) such that for every data instance $\mathcal{A}$ for $\mathcal{O}$ and all $\boldsymbol{a} \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{O}, \mathcal{A} \models q(\boldsymbol{a})$ iff $\mathcal{I}_{\mathcal{A}} \models Q(\boldsymbol{a})$ where $\mathcal{I}_{\mathcal{A}}$ is the interpretation that corresponds to $\mathcal{A}$ (in the obvious way).

We also consider *datalog-rewritability*, defined in the same way as FO-rewritability, but using datalog programs in place of FOQs. Using Rossman's homomorphism preservation theorem Rossman [2008], one can show that an OMQ $(\mathcal{O}, q(\boldsymbol{x}))$ with $\mathcal{O}$ a schema.org-ontology and $q(\boldsymbol{x})$ a UCQ is FO-rewritable iff it has a UCQ-rewriting iff it has a non-recursive datalog rewriting, see Bienvenu *et al.* [2014b] for more details (in a slightly different context). Since non-recursive datalog-rewritings can be more succinct than UCQ-rewritings, we will generally prefer the former.

3 Basic schema.org-Ontologies

We start with considering *basic* schema.org-ontologies, which are not allowed to contain enumeration definitions and datatypes. The results obtained here can be easily extended



(a) Data instance $\mathcal{A}_m$.

(b) Data instance $\mathcal{A}'_m$.

Fig. 1: ABoxes used in Example 5 and the paragraph below Theorem 10.

to basic schema.org-ontologies with datatypes but do not hold for ontologies with enumeration definitions (as will be shown in the next section). In Schema.org 2015, 45 concept names from a total of 622 are defined using enumeration definitions, and hence are not covered by the results presented in this section.

We start with noting that the *entailment problem for basic schema.org-ontologies* is decidable in polynomial time. This problem is to check whether $\mathcal{O} \models \alpha$ for a given basic schema.org-ontology $\mathcal{O}$ and a given inclusion α of the form allowed in such ontologies. In fact, the algorithm is straightforward. For example, $\mathcal{O} \models \mathsf{ran}(r) \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ if there is a role name s and a range restriction $\mathsf{ran}(s) \sqsubseteq B_1 \sqcup \cdots \sqcup B_m \in \mathcal{O}$ such that $\mathcal{O}_R \models r \sqsubseteq s$ and $\mathcal{O}_C \models B_j \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ for all $1 \leq j \leq m$, where $\mathcal{O}_R$ and $\mathcal{O}_C$ denote the set of role inclusions and atomic concept inclusions in $\mathcal{O}$.

Theorem 2. *The entailment problem for basic schema.org-ontologies is in* PTIME.

The hardness results reported in Theorem 5 crucially rely on existential quantification in the actual query. In fact, it follows from results in Grau *et al.* [2013]; Kaminski *et al.* [2014b] that given an OMQ $Q = (\mathcal{O}, q(\boldsymbol{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\boldsymbol{x})$ a CQ without quantified variables, it is possible to construct a non-recursive datalog rewriting of Q in polynomial time, and that such OMQs can be evaluated in PTIME in combined complexity. We aim to push this bound further by admitting restricted forms of quantification.

A CQ q has *qvar-size* n if all connected components of quantified variables in the undirected graph underlying G^q have size at most n. For example, quantifier-free CQs have qvar-size 0 and the following query $q(x, y)$ has qvar-size 1:

$$\exists z_1 \exists z_2 \bigwedge_{v \in \{x,y\}} (\mathsf{producedBy}(z_1, v) \wedge \mathsf{musicBy}(v, z_2))$$

The above consequences of the work by Grau, Kaminski, et al. can easily be extended to OMQs where queries have qvar-size one. In what follows, we consider qvar-size two, which is more subtle and where, in contrast to qvar-size one, reasoning by case distinction is required. The following example shows that there are CQs of qvar-size two for which no non-recursive datalog rewriting exists.

Example 5. Let $\mathcal{O} = \{\mathsf{ran}(s) \sqsubseteq A \sqcup B\}$ and consider the following CQ of qvar-size two: $q(x) = \exists x_1 \exists x_2 (s(x, x_1) \wedge A(x_1) \wedge r(x_1, x_2) \wedge B(x_2))$. It is easy to see that $\mathcal{O}, \mathcal{A}_m \models q(a)$ for every data instance $\mathcal{A}_m$ with $m \geq 2$ as defined in Figure 1a.

By applying locality arguments and using the data instances $\mathcal{A}_m$, one can in fact show that $(\mathcal{O}, q(x))$ is not FO-rewritable (note that removing one $r(b_i, b_{i+1})$ from $\mathcal{A}_m$ results in $q(a)$ no longer being entailed).

Theorem 3. *For every OMQ $(\mathcal{O}, q(\boldsymbol{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\boldsymbol{x})$ a CQ of qvar-size at most two, one can construct a datalog-rewriting in polynomial time. Moreover, evaluating OMQs from this class is in* PTIME *in combined complexity.*

Applied to Example 5, the proof of Theorem 3 yields a datalog rewriting that consists of the rules

$$P(x_1, x_2, x) \leftarrow s(x, x_1) \wedge X_1(x_1) \wedge r(x_1, x_2) \wedge X_2(x_2)$$

where the X_i range over A, B, and $\exists y\ r(y, \cdot)$, plus

$$\begin{array}{ll} I_A(x_1, x) \leftarrow P(x_1, x_2, x) \wedge A(x_1) & I_B(x_2, x) \leftarrow P(x_1, x_2, x) \wedge B(x_2) \\ I_A(x_2, x) \leftarrow P(x_1, x_2, x) \wedge I_A(x_1, x) & I_B(x_1, x) \leftarrow P(x_1, x_2, x) \wedge I_B(x_2, x) \\ \mathsf{goal}(x) \leftarrow s(x, x_1) \wedge I_A(x_1, x) \wedge r(x_1, x_2) \wedge I_B(x_2, x). & \end{array}$$

The recursive rule for I_A (the one for I_B is dual) says that if the only option to possibly avoid a match for (x_1, x_2, x) is to color (x_1, x) with I_A, then the only way to possibly avoid a match for (x_1, x_2, x) is to color (x_2, x) with I_A (otherwise, since $\mathsf{ran}(s) \sqsubseteq A \sqcup B \in \mathcal{O}$, it would have to be colored with I_B which gives a match).

The rewriting presented in Theorem 3 can easily be extended to accommodate datatypes. For schema.org-ontologies $\mathcal{O}$ that are not basic, the rewriting is sound but not necessarily complete, and can thus be used to compute approximate query answers.

Interestingly, Theorem 3 cannot be generalized to UCQs. This follows from the result in the full version that for basic schema.org-ontologies $\mathcal{O}$ and quantifier-free UCQs $q(x)$ (even without role atoms), the problem $\mathcal{O}, \mathcal{A} \models q(a)$ is coNP-hard regarding combined complexity for data instances $\mathcal{A}$ with a single individual a. Since evaluating datalog programs in such data instances is in PTIME, datalog rewritings of UCQ-based OMQs can thus not be constructed in polynomial time (unless PTIME equals NP). We note that it is not difficult to show (and follows from FO-rewritability of instance queries in DL-Lite$^{\mathcal{H}}_{\mathsf{bool}}$ Artale *et al.* [2009]) that given an OMQ $(\mathcal{O}, q(\boldsymbol{x}))$ with $\mathcal{O}$ a basic schema.org-ontology and $q(\boldsymbol{x})$ a quantifier-free UCQ, one can construct an FO-rewriting in exponential time, and thus query evaluation is in AC^0 in data complexity.

We now classify basic schema.org-ontologies $\mathcal{O}$ according to the data complexity of evaluating OMQs $(\mathcal{O}, q)$ with q a UCQ (or CQ). It is convenient to work with *minimized* ontologies where for all inclusions $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ and all $i \leq n$, there is a model $\mathcal{I}$ of $\mathcal{O}$ and a $d \in \Delta^{\mathcal{I}}$ such that d *satisfies* $F \sqcap A_i \sqcap \bigsqcap_{j \neq i} \neg A_j$ (defined in the usual way). Every schema.org-ontology can be rewritten in polynomial time into an equivalent minimized one. We establish the following dichotomy theorem.

Theorem 4. *Let $\mathcal{O}$ be a minimized basic schema.org-ontology. If there exists $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$, then there is a Boolean CQ q that uses only concept and role names from $\mathcal{O}$ and such that $(\mathcal{O}, q)$ is* CONP*-hard in data complexity. Otherwise, a given OMQ $(\mathcal{O}, q)$ with q a UCQ can be rewritten into a non-recursive datalog-program in polynomial time (and is thus in AC^0 in data complexity).*

The proof of the second part of Theorem 4 is easy: if there are no $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n \in \mathcal{O}$ with $n \geq 2$, then $\mathcal{O}$ essentially is already a non-recursive datalog program and the construction is straightforward. The proof of the hardness part is obtained by extending

the corresponding part of a dichotomy theorem for $\mathcal{ALC}$-ontologies of depth one Lutz and Wolter [2012]. The main differences between the two theorems are that (i) for basic schema.org-ontologies, the dichotomy is decidable in PTIME (whereas decidability is open for $\mathcal{ALC}$), (ii) the CQs in CONP-hard OMQs use only concept and role names from $\mathcal{O}$ (this is not possible in $\mathcal{ALC}$), and (iii) the dichotomy is between AC^0 and CONP whereas for $\mathcal{ALC}$ OMQs can be complete for PTIME, NL, etc.

By Theorem 4, disjunctions in domain and range restrictions are the only reason that query answering is non-tractable for basic schema.org-ontologies. In Schema.org 2015, 14% of all range restrictions and 20% of all domain restrictions contain disjunctions.

In Theorem 4, we have classified the data complexity of *ontologies*, quantifying over the actual queries. In what follows, we aim to classify the data complexity of every OMQ. This problem turns out to be much harder and, in fact, we show that a classification of the data complexity of OMQs based on basic schema.org-ontologies and UCQs implies a classification of constraint satisfaction problems according to their complexity (up to FO-reductions), a famous open problem that is the subject of significant ongoing research Feder and Vardi [1998]; Bulatov [2011].

A *signature* is a set of concept and role names (also called *symbols*). Let $\mathcal{B}$ be a finite interpretation that interprets only the symbols from a finite signature Σ. The *constraint satisfaction problem CSP*($\mathcal{B}$) is to decide, given a data instance $\mathcal{A}$ over Σ, whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$. In this context, $\mathcal{B}$ is called the *template* of CSP($\mathcal{B}$).

Theorem 5. *For every template $\mathcal{B}$, one can construct in polynomial time an OMQ $(\mathcal{O}, q)$ where $\mathcal{O}$ is a basic schema.org-ontology and q a Boolean acyclic UCQ such that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.*

Theorem 13 below establishes the converse direction of Theorem 5 for unrestricted schema.org-ontologies and a large class of (acyclic) UCQs. From Theorem 13, we obtain a NEXPTIME-upper bound for deciding FO-rewritability and datalog-rewritability of a large class of OMQs. It remains open whether this bound is tight, but we can show a PSPACE lower bound for FO-rewritable using a reduction of the word problem of PSPACE Turing machines. The proof uses the ontology $\mathcal{O}$ and data instances $\mathcal{A}_m$ from Example 5 and is similar to a PSPACE lower bound proof for FO-rewritability in consistent query answering Lutz and Wolter [2015] which is, in turn, based on a construction from Cosmadakis *et al.* [1988].

Theorem 6. *It is* PSPACE*-hard to decide whether a given OMQ $(\mathcal{O}, q)$ with $\mathcal{O}$ a basic schema.org-ontology and q a Boolean acyclic UCQ is FO-rewritable.*

4 Incoherence and Unsatisfiability

In the subsequent section, we consider unrestricted schema.org ontologies instead of basic ones, that is, we add back enumeration definitions and datatypes. The purpose of this section is to deal with a complication that arises from this step, namely the potential presence of inconsistencies. We start with inconsistencies that concern the ontology alone and then consider inconsistencies that arise from combining an ontology with a data instance.

An ontology $\mathcal{O}$ is *incoherent* if there exists $X \in \mathsf{N_C} \cup \mathsf{N_R}$ such that $X^{\mathcal{I}} = \emptyset$ for all models $\mathcal{I}$ of $\mathcal{O}$. Incoherent ontologies can result from the UNA for enumeration

individuals such as in the ontology $\{A \equiv \{a\}, B \equiv \{b\}, A \sqsubseteq B\}$, which has no model if $a \neq b$; they can also arise from interactions between concept names and datatypes such as in the ontology $\{\mathsf{ran}(r) \sqsubseteq \mathsf{Integer}, \mathsf{ran}(s) \sqsubseteq A, r \sqsubseteq s\}$ with $A \in \mathsf{N_C}$ which has no model $\mathcal{I}$ with $r^{\mathcal{I}} \neq \emptyset$ since $\Delta^{\mathcal{I}} \cap \Delta^{\mathsf{Integer}} = \emptyset$. Using Theorem 2, one can show:

Theorem 7. *Incoherence of schema.org-ontologies can be decided in PTime.*

We now turn to inconsistencies that arise from combining an ontology $\mathcal{O}$ with a data instance $\mathcal{A}$ for $\mathcal{O}$. As an example, consider $\mathcal{O} = \{A \equiv \{a\}, B \equiv \{b\}\}$ and $\mathcal{A} = \{A(c), B(c)\}$. Although $\mathcal{O}$ is coherent, $\mathcal{A}$ is unsatisfiable w.r.t. $\mathcal{O}$. Like incoherence, unsatisfiability is decidable in polynomial time. In fact, we can even show the following stronger result.

Theorem 8. *Given a schema.org-ontology $\mathcal{O}$, one can compute in polynomial time a non-recursive datalog program Π such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\mathcal{A}$ is unsatisfiable w.r.t. $\mathcal{O}$ iff $\Pi(\mathcal{A}) \neq \emptyset$.*

In typical schema.org applications, the data is collected from the web and it is usually not acceptable to simply report back an inconsistency and stop processing the query. Instead, one would like to take maximum advantage of the data despite the presence of an inconsistency. There are many semantics for inconsistent query answering that can be used for this purpose. As efficiency is paramount in schema.org applications, our choice is the pragmatic intersection repair (IAR) semantics which avoids CONP-hardness in data complexity Lembo *et al.* [2010]; Rosati [2011]; Bienvenu *et al.* [2014a]. A *repair* of a data instance $\mathcal{A}$ w.r.t. an ontology $\mathcal{O}$ is a maximal subset $\mathcal{A}' \subseteq \mathcal{A}$ that is satisfiable w.r.t. $\mathcal{O}$. We use $\mathsf{rep}_{\mathcal{O}}(\mathcal{A})$ to denote the set of all repairs of $\mathcal{A}$ w.r.t. $\mathcal{O}$. The idea of IAR semantics is then to replace $\mathcal{A}$ with $\bigcap_{\mathcal{A}' \in \mathsf{rep}_{\mathcal{O}}(\mathcal{A})} \mathcal{A}'$. In other words, we have to remove from $\mathcal{A}$ all assertions that occur in some minimal subset $\mathcal{A}' \subseteq \mathcal{A}$ that is unsatisfiable w.r.t. $\mathcal{O}$. We call such an assertion a *conflict assertion*.

Theorem 9. *Given a schema.org-ontology $\mathcal{O}$ and concept name A (resp. role name r), one can compute a non-recursive datalog program Π such that for any data instance $\mathcal{A}$ for $\mathcal{O}$, $\Pi(\mathcal{A})$ is the set of all $a \in \mathsf{Ind}(\mathcal{A})$ (resp. $(a, b) \in \mathsf{Ind}(\mathcal{A})^2$) such that $A(a)$ (resp. $r(a, b)$) is a conflict assertion in $\mathcal{A}$.*

By Theorem 9, we can adopt the IAR semantics by simply removing all conflict assertions from the data instance before processing the query. Programs from Theorem 9 become exponential in the worst case, but can be expected to be very small in practical cases. In the remainder of the paper, we assume that ontologies are coherent and that $\mathcal{A}$ is satisfiable w.r.t. $\mathcal{O}$ if we query a data instance $\mathcal{A}$ using an ontology $\mathcal{O}$.

5 Unrestricted schema.org-Ontologies

We aim to lift the results from Section 3 to unrestricted schema.org-ontologies. Regarding Theorem 3, it turns out that quantified variables in CQs are computationally much more problematic when there are enumeration definitions in the ontology. In fact, one can expect positive results only for quantifier-free CQs, and even then the required constructions are quite subtle.

Theorem 10. *Given an OMQ $Q = (\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a quantifier-free CQ, one can construct in polynomial time a datalog-rewriting of Q. Moreover, evaluating OMQs from this class is in* PTIME *in combined complexity. The rewriting is non-recursive if $q = A(x)$.*

The following example illustrates the construction of the datalog program. Let $\mathcal{O} = \{A \equiv \{a_1, a_2\}\}$ and $q() = r(a_1, a_2)$. Observe that $\mathcal{O}, \mathcal{A}'_m \models q()$ for every data instance $\mathcal{A}'_m$ defined in Figure 1b. Similarly to Example 5, one can use the data instances $\mathcal{A}'_m$ to show that $(\mathcal{O}, q())$ is not FO-rewritable.

A datalog-rewriting of $(\mathcal{O}, q())$ is given by the program Π_{a_1,a_2} which contains

$$\begin{aligned}
\mathsf{goal}() &\leftarrow r(a_1, a_2)\\
\mathsf{goal}() &\leftarrow r(a_1, x) \wedge \mathsf{path}_A(x, y) \wedge r(y, a_2)\\
\mathsf{path}_A(x, y) &\leftarrow r(x, y) \wedge A(x) \wedge A(y)\\
\mathsf{path}_A(x, y) &\leftarrow \mathsf{path}_A(x, z) \wedge \mathsf{path}_A(z, y).
\end{aligned}$$

It is also instructive to check that $\mathcal{O}', \mathcal{A}'_m \not\models q()$ with $\mathcal{O}' = \{A \equiv \{a_1, a_2, a_3\}\}$ because in models of $\mathcal{O}'$, a_3 can be identified with some b_i, a_1 with $b_1, \ldots, b_{i-1}$ and a_2 with $b_{i+1}, \ldots, b_m$, $1 \leq i \leq m$.

We now modify the datalog program above to obtain a rewriting of the OMQ $(\mathcal{O}, q'(x, y))$ with $q(x, y) = r(x, y)$. First, we include in Π_r the rules $A(a_1) \leftarrow \mathsf{true}$ and $A(a_2) \leftarrow \mathsf{true}$. Then we add the following rules:

$$\mathsf{goal}(x, y) \leftarrow r(x, y), \qquad \mathsf{goal}(x, y) \leftarrow A(x) \wedge A(y) \wedge \bigwedge_{1 \leq i,j \leq 2} R_{a_i,a_j}(x, y).$$

We want to use the latter rule to check that x, y have to be mapped to $\{a_1, a_2\}$, and that for every possible assignment a_i, a_j to x, y that is consistent (i.e., we do not have $x \in \{a_1, a_2\}$ and $x \neq a_i$, and similarly for y), $r(a_i, a_j)$ is true. To this end, we add the rules:

$$\begin{aligned}
R_{a_i,a_j}(x, y) &\leftarrow \mathsf{neq}(x, a_i) \qquad R_{a_i,a_j}(x, y) \leftarrow \mathsf{neq}(y, a_j)\\
R_{a_i,a_j}(x, y) &\leftarrow \mathsf{goal}(a_i, a_j)\\
\mathsf{neq}(a_1, a_2) &\leftarrow \mathsf{true} \qquad\qquad \mathsf{neq}(a_2, a_1) \leftarrow \mathsf{true}.
\end{aligned}$$

It remains to add rules 3 and 4 from Π_{a_1,a_2} and

$$\mathsf{goal}(a_i, a_j) \leftarrow r(a_i, x) \wedge \mathsf{path}_A(x, y) \wedge r(y, a_j)$$

for $1 \leq i, j \leq 2$ and $i \neq j$.

Theorem 10 is tight in the sense that evaluating CQs with a single atom and a single existentially quantified variable, as well as quantifier-free UCQs, is coNP-hard in data complexity. For instance, let $\mathcal{O} = \{\mathsf{dom}(e) \sqsubseteq A,\ \mathsf{ran}(e) \sqsubseteq A,\ A \equiv \{r, g, b\}\}$. Then, an undirected graph $G = (V, E)$ is 3-colorable iff $\mathcal{O}, \{e(v, w) \mid (v, w) \in E\} \not\models \exists x\, e(x, x)$. Alternatively, one may replace the query by $r(r, r) \vee r(g, g) \vee r(b, b)$. In fact, one can prove the following variant of Theorem 5 which shows that classifying OMQs with ontologies using *only* enumeration definitions and quantifier-free UCQs according to their complexity is as hard as CSP.

Theorem 11. *Given a template $\mathcal{B}$, one can construct in polynomial time an OMQ $(\mathcal{O}, q)$ where $\mathcal{O}$ only contains enumeration definitions and q is a Boolean variable-free UCQ such that the complement of CSP($\mathcal{B}$) and $(\mathcal{O}, q)$ are mutually FO-reducible.*

We now turn to classifying the complexity of ontologies and of OMQs, starting with a generalization of Theorem 4 to unrestricted schema.org-ontologies.

Theorem 12. *Let $\mathcal{O}$ be a coherent and minimized schema.org-ontology. If $\mathcal{O}$ contains an enumeration definition $A \equiv \{a_1, \dots, a_n\}$ with $n \geq 2$ or contains an inclusion $F \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$ such that there are at least two concept names in $\{A_1, \dots, A_n\}$ and $\mathcal{O} \not\models F \sqsubseteq A \sqcup \bigsqcup_{(D, \Delta^{\mathcal{D}}) \in DT} D$ for any A with $A \equiv \{a\} \in \mathcal{O}$, then $(\mathcal{O}, q)$ is coNP-hard for some Boolean CQ q. Otherwise every $(\mathcal{O}, q)$ with q a UCQ is FO-rewritable (and thus in AC^0 in data complexity).*

Note that, in contrast to Theorem 4, being in AC^0 does not mean that no 'real disjunction' is available. For example, for $\mathcal{O} = \{\mathsf{ran}(r) \sqsubseteq A \sqcup B, A \sqsubseteq C, B \sqsubseteq C, C \equiv \{c\}\}$ and $\mathcal{A} = \{r(a, b)\}$ we have $\mathcal{O}, \mathcal{A} \models A(b) \vee B(b)$ and neither $A(b)$ nor $B(b)$ are entailed. This type of choice does not affect FO-rewritability, however, since it is restricted to individuals that must be identified with a unique individual in $\mathsf{N}_\mathsf{E}(\mathcal{O})$. Note that, for the hardness proof, we now need to use a role name that possibly does not occur in $\mathcal{O}$. For example, for $\mathcal{O} = \{A \equiv \{a_1, a_2\}\}$ there exists a Boolean CQ q such that $(\mathcal{O}, q)$ is NP-hard, but constructing q requires a fresh role name.

We now consider the complexity of single OMQs and show a converse of Theorems 5 and 11 for schema.org-ontologies and UCQs that are *qvar-acyclic*, that is, when all atoms $r(t, t')$ with neither of t, t' a quantified variable are dropped, then all CQs in it are acyclic. We use *generalized CSPs with marked elements* in which instead of a single template $\mathcal{B}$, one considers a finite set Γ of templates whose signature contains, in addition to concept and role names, a finite set of individual names. Homomorphisms have to respect also the individual names and the problem is to decide whether there is a homomorphism from the input interpretation to some $\mathcal{B} \in \Gamma$. Every such CSP is mutually FO-reducible with some standard CSP and FO-definability and datalog definability of the complement of generalized CSPs with marked elements are NP-complete Bienvenu *et al.* [2014b].

Theorem 13. *Given an OMQ $(\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a qvar-acyclic UCQ, one can compute in exponential time a generalized CSP with marked elements Γ such that $(\mathcal{O}, q)$ and the complement of CSP(Γ) are mutually FO-reducible.*

The proof uses an encoding of qvar-acyclic queries into concepts in the description logic $\mathcal{ALCIUO}$ that extends $\mathcal{ALC}$ by inverse roles, the universal role, and nominals. It extends the the template constructions of Bienvenu *et al.* [2014b] to description logics with nominals. As a particularly interesting consequence of Theorem 13, we obtain:

Theorem 14. *FO-rewritability and datalog-rewritability of OMQs $(\mathcal{O}, q)$ with $\mathcal{O}$ a schema.org-ontology and q a qvar-acyclic UCQ are decidable in* NExpTime.

6 Conclusion

The work presented in this paper lays a solid foundation for attacking many interesting and practically relevant questions that can be asked about querying in the presence of schema.org-ontologies. Topics of interest include different forms of queries such as SPARQL and regular path queries as well as uncertainty in the data that accounts for varying levels of trust in different data sources.

Bibliography

Foto N. Afrati and Jeffrey D. Ullman. Optimizing multiway joins in a map-reduce environment. *IEEE Trans. Knowl. Data Eng.*, 23(9):1282–1298, 2011.

Foto N. Afrati and Jeffrey D. Ullman. Transitive closure and recursive datalog implemented on clusters. In *EDBT*, pages 132–143, 2012.

Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. of Artifical Intelligence Research*, 36:1–69, 2009.

Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *AAAI*, pages 996–1002, 2014.

Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33, 2014.

Andrei A. Bulatov. On the CSP dichotomy conjecture. In *CSR*, pages 331–344, 2011.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 477–490, 1988.

Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

Birte Glimm and Markus Krötzsch. SPARQL beyond subgraph matching. In *ISWC*, volume 6496 of *LNCS*, pages 241–256. Springer, 2010.

Bernardo Cuenca Grau, Boris Motik, Giorgos Stoilos, and Ian Horrocks. Computing datalog rewritings beyond horn ontologies. In *IJCAI*, 2013.

Ramanathan V. Guha. Light at the end of the tunnel? Invited Talk, ISWC, https://www.youtube.com/watch?v=oFY-0QoxBi8, 2013.

Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *Web Reasoning and Rule Systems*, pages 76–91, 2014.

Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In *AAAI*, pages 1077–1083, 2014.

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Web Reasoning and Rule Systems*, pages 103–117, 2010.

Carsten Lutz and Frank Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of KR*, 2012.

Carsten Lutz and Frank Wolter. On the relationship between consistent query answering and constraint satisfaction problems. In *ICDT*, 2015.

Peter F. Patel-Schneider. Analyzing schema.org. In *ISWC, Part I*, pages 261–276, 2014.

Riccardo Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI*, pages 1057–1062, 2011.

Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3), 2008.

Polynomial Horn Rewritings for Description Logics Ontologies*

Mark Kaminski and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford, UK

Abstract. We study the problem of rewriting an ontology $\mathcal{O}_1$ in a DL $\mathcal{L}_1$ into an ontology $\mathcal{O}_2$ in a Horn DL $\mathcal{L}_2$ such that $\mathcal{O}_1$ and $\mathcal{O}_2$ are equisatisfiable when extended with any dataset. After showing undecidability whenever $\mathcal{L}_1$ extends $\mathcal{ALCF}$, we focus on devising efficiently checkable conditions that ensure existence of a Horn rewriting. By lifting existing Datalog rewriting techniques for Disjunctive Datalog programs to first-order programs with function symbols, we identify a class of ontologies that admit Horn rewritings of polynomial size. Our experiments indicate that many real-world ontologies admit such polynomial Horn rewritings.

1 Introduction

Reasoning over ontology-enriched datasets is a key requirement in many applications. Standard reasoning tasks are, however, of high worst-case complexity. Satisfiability checking is 2NExpTime-complete for the DL $\mathcal{SROIQ}$ underpinning OWL 2 and NExpTime-complete for $\mathcal{SHOIN}$, which underpins OWL DL [13]. Reasoning is also co-NP-hard in *data complexity*—a key measure of complexity for applications involving large amounts of instance data [9].

Tractability in data complexity is typically associated with *Horn* DLs, where ontologies correspond to first-order Horn clauses [18, 9]. The more favourable computational properties of Horn DLs make them a natural choice for data-intensive applications, but they also come at the expense of a loss in expressive power. In particular, Horn DLs cannot capture disjunctive axioms, i.e., statements such as "every X is either a Y or a Z". Disjunctive axioms are common in real-world ontologies, like the NCI Thesaurus or the ontologies underpinning the EBI linked data platform (see http://www.ebi.ac.uk/rdf/platform).

In this paper we are interested in *Horn rewritability* of description logic ontologies; that is, whether an ontology $\mathcal{O}_1$ in a DL $\mathcal{L}_1$ can be restated as an ontology $\mathcal{O}_2$ in a Horn DL $\mathcal{L}_2$ such that $\mathcal{O}_1$ and $\mathcal{O}_2$ are equisatisfiable when extended with an arbitrary dataset. Ontologies admitting such rewritings are amenable to more efficient reasoning techniques that are tractable in data complexity.

Horn rewritability of DL ontologies is strongly related to the rewritability of Disjunctive Datalog programs into Datalog, where both the source and target

* Work supported by the Royal Society, the EPSRC projects Score!, MaSI^3 and DBOnto, and the FP7 project Optique.

languages for rewriting are function-free. Kaminski et al. [12] characterised Datalog rewritability of Disjunctive Datalog programs in terms of linearity: a restriction that requires each rule to contain at most one body atom that is IDB (i.e., whose predicate also occurs in head position in the program). It was shown that every linear Disjunctive Datalog program can be rewritten into plain Datalog (and vice versa) by means of *program transposition*—a polynomial transformation in which rules are "inverted" by shuffling all IDB atoms between head and body while replacing their predicates by auxiliary ones. Subsequently, Kaminski et al. [11] proposed the class of *markable* Disjunctive Datalog programs, where the linearity requirement is relaxed so that it applies only to a subset of "marked" atoms. Every markable program can be polynomially rewritten into Datalog by exploiting a variant of transposition where only marked atoms are affected.

Our contributions in this paper are as follows. In Section 3, we show undecidability of Horn rewritability for ontologies in $\mathcal{ALCF}$. This is in consonance with the related undecidability results by Bienvenu et al. [3] and Lutz and Wolter [17]. In Section 4, we lift the markability condition and the transposition transformation in [11] for Disjunctive Datalog to first-order programs with function symbols. We then show that all markable programs admit Horn rewritings of polynomial size. This result is rather general and has potential implications in areas such as theorem proving [19] and knowledge compilation [5]. The notion of markability for first-order programs easily transfers to ontologies via the standard FOL translation of DLs [2]. This is, however, of limited practical value since Horn programs obtained via transposition may not be expressible using standard DL constructors. In Section 5, we introduce an alternative satisfiability-preserving translation from $\mathcal{ALCHIF}$ ontologies to first-order programs and show in Section 6 that the corresponding transposed programs can be translated back into Horn-$\mathcal{ALCHIF}$ ontologies. Finally, we focus on complexity and show that reasoning over markable $\mathcal{L}$-ontologies is ExpTime-complete in combined complexity and PTime-complete w.r.t. data for each DL $\mathcal{L}$ between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. All our results immediately extend to DLs with transitive roles (e.g., $\mathcal{SHIF}$) by exploiting standard transitivity elimination techniques [2].

We have implemented markability checking and evaluated our techniques on a large ontology repository. Our results indicate that many real-world ontologies are markable and thus admit Horn rewritings of polynomial size.

All proofs are deferred to an extended version (see `arXiv:1504.05150`).

2 Preliminaries

We assume standard first-order syntax and semantics. We treat the universal truth $\top$ and falsehood $\bot$ symbols as well as equality ($\approx$) as ordinary predicates of arity one ($\top$ and $\bot$) and two ($\approx$), the meaning of which will be axiomatised.

Programs A *(first-order) rule* is a sentence $\forall \boldsymbol{x} \forall \boldsymbol{z}.[\varphi(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \psi(\boldsymbol{x})]$ where variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are disjoint, $\varphi(\boldsymbol{x}, \boldsymbol{z})$ is a conjunction of distinct atoms over $\boldsymbol{x} \cup \boldsymbol{y}$, and $\psi(\boldsymbol{x})$ is a disjunction of distinct atoms over $\boldsymbol{x}$. Formula φ is the *body* of r, and ψ is the *head*. Quantifiers are omitted for brevity, and safety is assumed

$T1.$	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$	$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{j=1}^{m} C_j(x)$
$T2.$	$\exists R.A \sqsubseteq C$	$\mathsf{at}(R, x, y) \wedge A(y) \rightarrow C(x)$
$T3.$	$A \sqsubseteq \exists R.B$	$A(x) \rightarrow \mathsf{at}(R, x, f(x));\ A(x) \rightarrow B(f(x))$
$T4.$	$A \sqsubseteq \forall R.C$	$A(x) \wedge \mathsf{at}(R, x, y) \rightarrow C(y)$
$T5.$	$S \sqsubseteq R$	$S(x, y) \rightarrow \mathsf{at}(R, x, y)$
$T6.$	$A \sqsubseteq \leq 1\, R.B$	$A(z) \wedge \mathsf{at}(R, z, x_1) \wedge \mathsf{at}(R, z, x_2) \wedge B(x_1) \wedge B(x_2) \rightarrow x_1 \approx x_2$

Table 1. Normalised DL axioms. A, B are named or $\top$; C named or $\bot$; role S is named and R is a (possibly inverse) role.

(all variables in the rule occur in the body). We define the following sets of rules for a finite signature Σ: *(i)* $\mathcal{P}_\Sigma^\top$ consists of a rule $P(x_1, \ldots, x_n) \rightarrow \top(x_i)$ for each predicate $P \in \Sigma$ and each $1 \leq i \leq n$ and a rule $\rightarrow \top(a)$ for each constant $a \in \Sigma$; *(ii)* $\mathcal{P}_\Sigma^\bot$ consists of the rule with $\bot(x)$ in the body and an empty head; and *(iii)* $\mathcal{P}_\Sigma^\approx$ consists of the standard axiomatisation of $\approx$ as a congruence over Σ. A *program* is a finite set of rules $\mathcal{P} = \mathcal{P}_0 \cup \mathcal{P}_\Sigma^\top \cup \mathcal{P}_\Sigma^\bot \cup \mathcal{P}_\Sigma^\approx$ with Σ the signature of $\mathcal{P}_0$, where we assume w.l.o.g. that the body of each rule in $\mathcal{P}_0$ does not mention $\bot$ or $\approx$, and the head is non-empty and does not mention $\top$. We omit Σ for the components of $\mathcal{P}$ and write $\mathcal{P}^\top$, $\mathcal{P}^\bot$ and $\mathcal{P}^\approx$. A rule is *Horn* if its head consists of at most one atom, and a program is Horn if so are all of its rules. Finally, a *fact* is a ground, function-free atom, and a *dataset* is a finite set of facts.

Ontologies A signature Σ consists of disjoint countable sets of concept names Σ_C and role names Σ_R. A role is an element of $\Sigma_R \cup \{R^- \mid R \in \Sigma_R\}$. The function inv is defined over roles as follows, where $R \in \Sigma_R$: $\mathsf{inv}(R) = R^-$ and $\mathsf{inv}(R^-) = R$. W.l.o.g., we consider normalised axioms as on the left-hand side of Table 1. An $\mathcal{ALCHIF}$ ontology $\mathcal{O}$ is a finite set of axioms of type T1-T6 in Table 1. An ontology is *Horn* if it contains no axiom T1 where $m \geq 2$. Given $\mathcal{O}$, we write $\sqsubseteq^*$ for the minimal reflexive and transitive relation over roles in $\mathcal{O}$ s.t. $R_1 \sqsubseteq^* R_2$ and $\mathsf{inv}(R_1) \sqsubseteq^* \mathsf{inv}(R_2)$ hold whenever $R_1 \sqsubseteq R_2 \in \mathcal{O}$.

We refer to the DL where only axioms T1–T3 are available and inverse roles are disallowed as $\mathcal{ELU}$. The logic $\mathcal{ALC}$ extends $\mathcal{ELU}$ with axioms T4. We then use standard naming conventions for DLs based on the presence of inverses ($\mathcal{I}$), axioms T5 ($\mathcal{H}$) and axioms T6 ($\mathcal{F}$). An ontology is $\mathcal{EL}$ if it is $\mathcal{ELU}$ and Horn.

Table 1 also provides the standard translation π from normalised axioms into rules, where $\mathsf{at}(R, x, y)$ stands for $R(x, y)$ if R is named and $S(y, x)$ if $R = S^-$. We define $\pi(\mathcal{O})$ as the smallest program containing $\pi(\alpha)$ for each axiom α in $\mathcal{O}$. Given a dataset $\mathcal{D}$, we say that $\mathcal{O} \cup \mathcal{D}$ is *satisfiable* iff so is $\pi(\mathcal{O}) \cup \mathcal{D}$ in FOL.

3 Horn Rewritability

Our focus is on satisfiability-preserving rewritings. Standard reasoning tasks in DLs are reducible to unsatisfiability checking [2], which makes our results practically relevant. We start by formulating our general notion of rewriting.

Definition 1. *Let $\mathcal{F}, \mathcal{F}'$ be sets of rules. Then $\mathcal{F}'$ is a* rewriting *of $\mathcal{F}$ if it holds that $\mathcal{F} \cup \mathcal{D}$ is satisfiable iff so is $\mathcal{F}' \cup \mathcal{D}$ for each dataset $\mathcal{D}$ over predicates from $\mathcal{F}$.*

We are especially interested in computing Horn rewritings of ontologies—that is, rewritings where the given ontology $\mathcal{O}_1$ is expressed in a DL $\mathcal{L}_1$ and the rewritten ontology $\mathcal{O}_2$ is in a Horn DL $\mathcal{L}_2$ (where preferably $\mathcal{L}_2 \subseteq \mathcal{L}_1$). This is not possible in general: satisfiability checking is co-NP-complete in data complexity even for the basic logic $\mathcal{ELU}$ [14], whereas data complexity is tractable even for highly expressive Horn languages such as Horn-$\mathcal{SROIQ}$ [18]. Horn rewritability for DLs can be formulated as a decision problem as follows:

Definition 2. *The $(\mathcal{L}_1, \mathcal{L}_2)$-Horn rewritability problem for DLs $\mathcal{L}_1$ and $\mathcal{L}_2$ is to decide whether a given $\mathcal{L}_1$-ontology admits a rewriting expressed in Horn-$\mathcal{L}_2$.*

Our first result establishes undecidability whenever the input ontology contains at-most cardinality restrictions and thus equality. This result fits in with the related undecidability results by Bienvenu et al. [3] and Lutz and Wolter [17] for Datalog rewritability and non-uniform data complexity for $\mathcal{ALCF}$ ontologies.

Theorem 3. *$(\mathcal{L}_1, \mathcal{L}_2)$-Horn rewritability is undecidable for $\mathcal{L}_1 = \mathcal{ALCF}$ and $\mathcal{L}_2$ any DL between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. This result holds if* PTIME≠NP.

Intractability results in data complexity rely on the ability of non-Horn DLs to encode co-NP-hard problems, such as non-3-colourability [14, 9]. In practice, however, it can be expected that ontologies do not encode such problems. Thus, our focus from now onwards will be on identifying classes of ontologies that admit (polynomial size) Horn rewritings.

4 Program Markability and Transposition

In this section, we introduce the class of *markable* programs and show that every markable program can be rewritten into a Horn program by means of a polynomial transformation, which we refer to as *transposition*. Roughly speaking, transposition inverts the rules in a program $\mathcal{P}$ by moving certain atoms from head to body and vice versa while replacing their corresponding predicates with fresh ones. Markability of $\mathcal{P}$ ensures that we can pick a set of predicates (a *marking*) such that, by shuffling only atoms with a marked predicate, we obtain a Horn rewriting of $\mathcal{P}$. Our results in this section generalise the results by Kaminski et al. [11] for Disjunctive Datalog to first-order programs with function symbols.

To illustrate our definitions throughout this section, we use an example program $\mathcal{P}_{\mathsf{ex}}$ consisting of the following rules:

$$A(x) \to B(x) \qquad B(x) \to C(x) \vee D(x)$$
$$C(x) \to \bot(x) \qquad D(x) \to C(f(x))$$

Markability. The notion of markability involves a partitioning of a program's predicates into *Horn* and *disjunctive*: the extension of Horn predicates for all

datasets depends only on the Horn rules in the program while the extension of disjunctive predicates may depend on a disjunctive rule. This intuition can be formalised using the standard notion of a dependency graph in logic programming.

Definition 4. *The* dependency graph $G_{\mathcal{P}} = (V, E, \mu)$ *of a program* $\mathcal{P}$ *is the smallest edge-labeled digraph such that: (i)* V *contains all predicates in* $\mathcal{P}$*; (ii)* $r \in \mu(P, Q)$ *whenever* $r \in \mathcal{P}$*,* P *is in the body of* r*, and* Q *is in the head of* r*; and (iii)* $(P, Q) \in E$ *whenever* $\mu(P, Q) \neq \emptyset$*. A predicate* Q depends on $r \in \mathcal{P}$ *if* $G_{\mathcal{P}}$ *has a path ending in* Q *and involving an* r*-labeled edge. Predicate* Q *is* Horn *if it depends only on Horn rules; otherwise,* Q *is* disjunctive*.*

For instance, predicates C, D, and $\bot$ are disjunctive in our example program $\mathcal{P}_{\mathsf{ex}}$, whereas A and B are Horn. We can now introduce the notion of a *marking*—a subset of the disjunctive predicates in a program $\mathcal{P}$ ensuring that the transposition of $\mathcal{P}$ where only marked atoms are shuffled between head and body results in a Horn program.

Definition 5. *A* marking *of a program* $\mathcal{P}$ *is a set* M *of disjunctive predicates in* $\mathcal{P}$ *satisfying the following properties, where we say that an atom is marked if its predicate is in* M*: (i) each rule in* $\mathcal{P}$ *has at most one marked body atom; (ii) each rule in* $\mathcal{P}$ *has at most one unmarked head atom; and (iii) if* $Q \in M$ *and* P *is reachable from* Q *in* $G_{\mathcal{P}}$*, then* $P \in M$*. A program is* markable *if it has a marking.*

Condition *(i)* in Def. 5 ensures that at most one atom is moved from body to head during transposition. Condition *(ii)* ensures that all but possibly one head atom are moved to the body. Finally, condition *(iii)* requires that all predicates depending on a marked predicate are also marked. We can observe that our example program $\mathcal{P}_{\mathsf{ex}}$ admits two markings: $M_1 = \{C, \bot\}$ and $M_2 = \{C, D, \bot\}$.

Markability can be efficiently checked via a 2-SAT reduction, where we assign to each predicate Q in $\mathcal{P}$ a variable X_Q and encode the constraints in Def. 5 as 2-clauses. For each rule $\varphi \wedge \bigwedge_{i=1}^{n} P_i(\boldsymbol{s}_i) \rightarrow \bigvee_{j=1}^{m} Q_j(\boldsymbol{t}_j)$, with φ the conjunction of all Horn atoms in the head, we include clauses *(i)* $\neg X_{P_i} \vee \neg X_{P_j}$ for all $1 \leq i < j \leq n$, which enforce at most one body atom to be marked; *(ii)* $X_{Q_i} \vee X_{Q_j}$ for $1 \leq i < j \leq m$, which ensure that at most one head atom is unmarked; and *(iii)* $\neg X_{P_i} \vee X_{Q_j}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, which close markings under rule dependencies. Each model of the resulting clauses yields a marking of $\mathcal{P}$.

Transposition. Before defining transposition, we illustrate the main intuitions using program $\mathcal{P}_{\mathsf{ex}}$ and marking M_1.

The first step to transpose $\mathcal{P}_{\mathsf{ex}}$ is to introduce fresh unary predicates $\overline{C}$ and $\overline{\bot}$, which stand for the negation of the marked predicates C and $\bot$. To capture the intended meaning of these predicates, we introduce rules $X(x) \rightarrow \overline{\bot}(x)$ for $X \in \{A, B, C, D\}$ and a rule $\overline{\bot}(x) \rightarrow \overline{\bot}(f(x))$ for the unique function symbol f in $\mathcal{P}_{\mathsf{ex}}$. The first rules mimick the usual axiomatisation of $\top$ and ensure that an atom $\overline{\bot}(c)$ holds in a Herbrand model of the transposed program whenever $X(c)$ also holds. The last rule ensures that $\overline{\bot}$ holds for all terms in the Herbrand universe of the transposed program—an additional requirement that is consistent with the intended meaning of $\overline{\bot}$, and critical to the completeness of transposition

in the presence of function symbols. Finally, a rule $\overline{\bot}(z) \land C(x) \land \overline{C}(x) \to \bot(z)$ ensures that the fresh predicate $\overline{C}$ behaves like the negation of C.

The key step of transposition is to invert the rules involving the marked predicates by shuffling marked atoms between head and body while replacing their predicate with the corresponding fresh one. In this way, rule $B(x) \to C(x) \lor D(x)$ yields $B(x) \land \overline{C}(x) \to D(x)$, and $C(x) \to \bot(x)$ yields $\overline{\bot}(x) \to \overline{C}(x)$. Additionally, rule $D(x) \to C(f(x))$ is transposed as $\overline{\bot}(z) \land D(x) \land \overline{C}(f(x)) \to \bot(z)$ to ensure safety. Finally, transposition does not affect rules containing only Horn predicates, e.g., rule $A(x) \to B(x)$ is included unchanged.

Definition 6. *Let M be a marking of a program $\mathcal{P}$. For each disjunctive predicate P in $\mathcal{P}$, let $\overline{P}$ be a fresh predicate of the same arity. The M-transposition of $\mathcal{P}$ is the smallest program $\Xi_M(\mathcal{P})$ containing every rule in $\mathcal{P}$ involving only Horn predicates and all rules given next, where φ is the conjunction of all Horn atoms in a rule, $\varphi_\top$ is the least conjunction of $\overline{\bot}$-atoms making a rule safe:*

1. $\varphi_\top \land \varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to \overline{Q}(\boldsymbol{t})$ *for each rule in $\mathcal{P}$ of the form* $\varphi \land Q(\boldsymbol{t}) \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$ *where $Q(\boldsymbol{t})$ is the only marked body atom;*
2. $\overline{\bot}(x) \land \varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to \bot(x)$, *where x a fresh variable, for each rule in $\mathcal{P}$ of the form* $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$, *with no marked body atoms and no unmarked head atoms;*
3. $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to P(\boldsymbol{s})$ *for each rule in $\mathcal{P}$ of the form* $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to P(\boldsymbol{s}) \lor \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$ *where $P(\boldsymbol{s})$ is the only unmarked head atom;*
4. $\overline{\bot}(z) \land P(\boldsymbol{x}) \land \overline{P}(\boldsymbol{x}) \to \bot(z)$ *for marked predicate P;*
5. $P(x_1, \ldots, x_n) \to \overline{\bot}(x_i)$ *for each P in $\mathcal{P}$ and $1 \leq i \leq n$;*
6. $\overline{\bot}(x_1) \land \ldots \land \overline{\bot}(x_n) \to \overline{\bot}(f(x_1, \ldots, x_n))$ *for each n-ary function symbol f in $\mathcal{P}$.*

Clearly, $\mathcal{P}_{\mathsf{ex}}$ is unsatisfiable when extended with fact $A(a)$. To see that $\Xi_{M_1}(\mathcal{P}_{\mathsf{ex}}) \cup \{A(a)\}$ is also unsatisfiable, note that $B(a)$ is derived by the unchanged rule $A(x) \to B(x)$. Fact $\overline{C}(a)$ is derived using $A(x) \to \overline{\bot}(x)$ and the transposed rule $\overline{\bot}(x) \to \overline{C}(x)$. We derive $D(a)$ using $B(x) \land \overline{C}(x) \to D(x)$. But then, to derive a contradiction we need to apply rule $\overline{\bot}(z) \land D(x) \land \overline{C}(f(x)) \to \bot(z)$, which is not possible unless we derive $\overline{C}(f(a))$. For this, we first use $\overline{\bot}(x) \to \overline{\bot}(f(x))$, which ensures that $\overline{\bot}$ holds for $f(a)$, and then $\overline{\bot}(x) \to \overline{C}(x)$.

Theorem 7. *Let M be a marking of a program $\mathcal{P}$. Then $\Xi_M(\mathcal{P})$ is a polynomial-size Horn rewriting of $\mathcal{P}$.*

It follows that every markable set of non-Horn clauses $\mathcal{N}$ can be polynomially transformed into a set of Horn clauses $\mathcal{N}'$ such that $\mathcal{N} \cup \mathcal{D}$ and $\mathcal{N}' \cup \mathcal{D}$ are equisatisfiable for every set of facts $\mathcal{D}$. This result is rather general and has potential applications in first-order theorem proving, as well as in knowledge compilation, where Horn clauses are especially relevant [5, 6].

5 Markability of DL Ontologies

The notion of markability is applicable to first-order programs and hence can be seamlessly adapted to ontologies via the standard translation π in Table 1.

Ontology $\mathcal{O}_{\mathsf{ex}}$	Rule translation $\xi(\mathcal{O}_{\mathsf{ex}})$	Transposition $\Xi_{M_{\mathsf{ex}}}(\xi(\mathcal{O}_{\mathsf{ex}}))$	Horn DL rewriting
$\alpha_1\ A \sqsubseteq B \sqcup C$	$A(x) \rightarrow B(x) \vee C(x)$	$A(x) \wedge \overline{B}(x) \rightarrow C(x)$	$A \sqcap \overline{B} \sqsubseteq C$
$\alpha_2\ B \sqsubseteq \exists R.D$	$B(x) \rightarrow D(f_{R,D}(x))$	$\overline{D}(f_{R,D}(x)) \rightarrow \overline{B}(x)$	$\exists R_D.\overline{D} \sqsubseteq \overline{B}$
$\alpha_3\ \exists R.D \sqsubseteq D$	$R(x,y) \wedge D(y) \rightarrow D(x)$	$R(x,y) \wedge \overline{D}(x) \rightarrow \overline{D}(y)$	$\overline{D} \sqsubseteq \forall R.\overline{D}$
	$D(f_{R,D}(x)) \rightarrow D(x)$	$\overline{D}(x) \rightarrow \overline{D}(f_{R,D}(x))$	$\overline{D} \sqsubseteq \forall R_D\overline{D}$
	$D(f_{R,B}(x)) \rightarrow D(x)$	$\overline{D}(x) \rightarrow \overline{D}(f_{R,B}(x))$	$\overline{D} \sqsubseteq \forall R_B\overline{D}$
$\alpha_4\ C \sqsubseteq \exists R.B$	$C(x) \rightarrow B(f_{R,B}(x))$	$\overline{\bot}(z) \wedge C(x) \wedge \overline{B}(f_{R,B}(x)) \rightarrow \bot(z)$	$C \sqcap \exists R_B.\overline{B} \sqsubseteq \bot$
$\alpha_5\ D \sqcap E \sqsubseteq \bot$	$D(x) \wedge E(x) \rightarrow \bot(x)$	$E(x) \wedge \overline{\bot}(x) \rightarrow \overline{D}(x)$	$E \sqcap \overline{\bot} \sqsubseteq \overline{D}$
		$X(x) \rightarrow \overline{\bot}(x),\ X \in \{A,B,C,D,E\}$	$X \sqsubseteq \overline{\bot}$
		$R(x_1,x_2) \rightarrow \overline{\bot}(x_i),\ 1 \leq i \leq 2$	$\top \sqsubseteq \forall R.\overline{\bot},\ \exists R.\top \sqsubseteq \overline{\bot}$
		$\overline{\bot}(x) \rightarrow \overline{\bot}(f_{R,Y}(x)),\ Y \in \{B,D\}$	$\overline{\bot} \sqsubseteq \exists R_Y.\overline{\bot}$

Table 2. Rewriting the example $\mathcal{ELU}$ ontology $\mathcal{O}_{\mathsf{ex}}$ into a Horn-$\mathcal{ALC}$ ontology using the marking $M_{\mathsf{ex}} = \{B, D, \bot\}$.

This, however, would be of limited value since the Horn programs resulting from transposition may not be expressible in Horn-$\mathcal{ALCHIF}$.

Consider any ontology with an axiom $\exists R.A \sqsubseteq B$ and any marking M involving R. Rule $R(x,y) \wedge A(y) \rightarrow B(x)$ stemming from π would be transposed as $\overline{B}(x) \wedge A(y) \rightarrow \overline{R}(x,y)$, which cannot be captured in $\mathcal{ALCHIF}$.[1]

To address this limitation we introduce an alternative translation ξ from DL axioms into rules, which we illustrate using the example ontology $\mathcal{O}_{\mathsf{ex}}$ in Table 2. The key idea is to encode existential restrictions in axioms T3 as unary atoms over functional terms. For instance, axiom α_2 in $\mathcal{O}_{\mathsf{ex}}$ would yield $B(x) \rightarrow D(f_{R,D}(x))$, where the "successor" relation between an instance b of B and some instance of D in a Herbrand model is encoded as a term $f_{R,D}(b)$, instead of a binary atom of the form $R(b, g(b))$. This encoding has an immediate impact on markings: by marking B we are only forced to also mark D (rather than both R and D). In this way, we will ensure that markings consist of unary predicates only.

To compensate for the lack of binary atoms involving functional terms in Herbrand models, we introduce new rules when translating axioms T2, T4, and T6 using ξ. For instance, $\xi(\alpha_3)$ yields the following rules in addition to $\pi(\alpha_3)$: a rule $D(f_{R,D}(x)) \rightarrow D(x)$ to ensure that all objects c with an R-successor $f_{R,D}(c)$ generated by $\xi(\alpha_2)$ are instances of D; a rule $D(f_{R,B}(x)) \rightarrow D(x)$, which makes sure that an object whose R-successor generated by $\xi(\alpha_4)$ is an instance of D is also an instance of D. Finally, axioms α_1 and α_5, which involve no binary predicates, are translated as usual.

Definition 8. *Let $\mathcal{O}$ be an ontology. For each concept $\exists R.B$ in an axiom of type T3, let $f_{R,B}$ be a unary function symbol, and Φ the set of all such symbols. We define $\xi(\mathcal{O})$ as the smallest program containing $\pi(\alpha)$ for each axiom α of type T1–T2 and T4–T6, as well as the following rules:*

– *$A(x) \rightarrow B(f_{R,B}(x))$ for each axiom T3;*

[1] Capturing such a rule would require a DL that can express products of concepts [20].

– $A(f_{R',Y}(x)) \to C(x)$ *for each axiom T2 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'),Y}(x)) \to C(x)$ *for each ax. T4 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$*,* $R' \sqsubseteq^* R$*;*
– $A(x) \wedge Y(f_{\mathsf{inv}(R'),Y}(x)) \to C(f_{\mathsf{inv}(R'),Y}(x))$ *for each axiom T2 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(x) \wedge Y(f_{R',Y}(x)) \to C(f_{R',Y}(x))$ *for each axiom T4 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(z) \wedge B(f_{R',Y}(z)) \wedge \mathsf{at}(R, z, x) \wedge B(x) \to f_{R',Y}(z) \approx x$ *for each ax. T6 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'),Y}(x)) \wedge B(x) \wedge \mathsf{at}(R, f_{\mathsf{inv}(R'),Y}(x), y) \wedge B(y) \to x \approx y$ *for each axiom T6 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(z) \wedge B(f_{R'_1,Y_1}(z)) \wedge B(f_{R'_2,Y_2}(z)) \to f_{R'_1,Y_1}(z) \approx f_{R'_2,Y_2}(z)$ *for each axiom T6 and* $f_{R'_i,Y_i} \in \Phi$ *s.t.* $R'_i \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'_1),Y_1}(x)) \wedge B(x) \wedge B(f_{R'_2,Y_2}(f_{\mathsf{inv}(R'_1),Y_1}(x))) \to x \approx f_{R'_2,Y_2}(f_{\mathsf{inv}(R'_1),Y_1}(x))$ *for each axiom T6 and* R'_i, Y_i *s.t.* $\{f_{\mathsf{inv}(R'_1),Y_1}, f_{R'_2,Y_2}\} \subseteq \Phi$ *and* $R'_i \sqsubseteq^* R$*.*

The translation $\xi(\mathcal{O}_{\mathsf{ex}})$ of our example ontology $\mathcal{O}_{\mathsf{ex}}$ is given in the second column of Table 2. Clearly, $\mathcal{O}_{\mathsf{ex}}$ is unsatisfiable when extended with $A(a)$ and $E(a)$. We can check that $\xi(\mathcal{O}_{\mathsf{ex}}) \cup \{A(a), E(a)\}$ is also unsatisfiable.

Theorem 9. *For every ontology* $\mathcal{O}$ *and dataset* $\mathcal{D}$ *over predicates in* $\mathcal{O}$ *we have that* $\mathcal{O} \cup \mathcal{D}$ *is satisfiable iff so is* $\xi(\mathcal{O}) \cup \mathcal{D}$*.*

This translation has a clear benefit for markability checking: in contrast to $\pi(\mathcal{O})$, binary predicates in $\xi(\mathcal{O})$ do not belong to any minimal marking. In particular, $M_{\mathsf{ex}} = \{B, D, \bot\}$ is the only minimal marking of $\xi(\mathcal{O}_{\mathsf{ex}})$.

Proposition 10. *(i) If* $\approx$ *is Horn in* $\xi(\mathcal{O})$ *then so are all binary predicates in* $\xi(\mathcal{O})$*. (ii) If* $\xi(\mathcal{O})$ *is markable, it has a marking containing only unary predicates.*

Thus, we define markability of ontologies in terms of ξ rather than π. We can check that $\pi(\mathcal{O}_{\mathsf{ex}})$ is not markable, whereas $\xi(\mathcal{O}_{\mathsf{ex}})$ admits the marking M_{ex}.

Definition 11. *An ontology* $\mathcal{O}$ *is markable if so is* $\xi(\mathcal{O})$*.*

We conclude this section with the observation that markability of an ontology $\mathcal{O}$ can be efficiently checked by first computing the program $\xi(\mathcal{O})$ and then exploiting the 2-SAT encoding sketched in Section 4.

6 Rewriting Markable Ontologies

It follows from the correctness of transposition in Theorem 7 and ξ in Theorem 9 that every $\mathcal{ALCHIF}$ ontology $\mathcal{O}$ admitting a marking M has a Horn rewriting of polynomial size given as the program $\Xi_M(\xi(\mathcal{O}))$. In what follows, we show that this rewriting can be expressed within Horn-$\mathcal{ALCHIF}$.

Let us consider the transposition of $\xi(\mathcal{O}_{\mathsf{ex}})$ via the marking M_{ex}, which is given in the third column of Table 2. The transposition of α_1 and α_5 corresponds directly to DL axioms via the standard translation in Table 1. In contrast, the transposition of all other axioms leads to rules that have no direct correspondence in DLs. The following lemma establishes that the latter rules are restricted to the types T7–T20 specified on the left-hand side of Table 3.

$T7.$	$\overline{\bot}(z) \wedge B(x) \wedge R(x,y) \wedge A(y) \rightarrow \bot(z)$	$B \sqcap \exists R.A \sqsubseteq \bot$
$T8.$	$\overline{\bot}(z) \wedge A(f_{R,Y}(x)) \wedge B(x) \rightarrow \bot(z)$	$B \sqcap \exists R_Y.A \sqsubseteq \bot$
$T9.$	$\overline{\bot}(x) \rightarrow \overline{\bot}(f_{R,Y}(x))$	$\overline{\bot} \sqsubseteq \exists R_Y.\overline{\bot}$
$T10.$	$B(x) \rightarrow A(f_{R,Y}(x))$	$B \sqsubseteq \forall R_Y.A \quad$ if $A \neq \overline{\bot}$ or $B \neq \overline{\bot}$
$T11.$	$B(f_{R,Y}(x)) \rightarrow A(x)$	$\exists R_Y.B \sqsubseteq A$
$T12.$	$A(x) \wedge B(f_{R,Y}(x)) \rightarrow C(f_{R,Y}(x))$	$A \sqcap \exists R_Y.B \sqsubseteq \forall R_Y.C$
$T13.$	$\overline{\bot}(z) \wedge A(x) \wedge B(f_{R,Y}(x)) \wedge C(f_{R,Y}(x)) \rightarrow \bot(z)$	$A \sqcap \exists R_Y(B \sqcap C) \sqsubseteq \bot$
$T14.$	$B(f_{R,Y}(x)) \wedge C(f_{R,Y}(x)) \rightarrow A(x)$	$\exists R_Y(B \sqcap C) \sqsubseteq A$
$T15.$	$A(z) \wedge B(f_{R',Y}(z)) \wedge \mathsf{at}(R,z,x) \wedge B(x) \rightarrow f_{R',Y}(z) \approx x$	$R'_Y \sqsubseteq S_{\{R'_Y,R\}},\ R \sqsubseteq S_{\{R'_Y,R\}},\ A \sqsubseteq \leq 1 S_{\{R'_Y,R\}}.B$
$T16.$	$A(f_{R',Y}(x)) \wedge B(x) \wedge \mathsf{at}(R,f_{R',Y}(x),y) \wedge B(y) \rightarrow x \approx y$	$\tilde{R}'_Y \sqsubseteq S_{\{\tilde{R}'_Y,R\}},\ R \sqsubseteq S_{\{\tilde{R}'_Y,R\}},\ A \sqsubseteq \leq 1 S_{\{\tilde{R}'_Y,R\}}.B,\ \tilde{R}'_Y \equiv \mathsf{inv}(R'_Y)$
$T17.$	$A(z) \wedge B(f_{R,Y}(z)) \wedge B(f_{R',Z}(z)) \rightarrow f_{R,Y}(z) \approx f_{R',Z}(z)$	$R_Y \sqsubseteq S_{\{R_Y,R'_Z\}},\ R'_Z \sqsubseteq S_{\{R_Y,R'_Z\}},\ A \sqsubseteq \leq 1 S_{\{R_Y,R'_Z\}}.B$
$T18.$	$A(f_{R,Y}(x)) \wedge B(x) \wedge B(f_{R',Z}(f_{R,Y}(x))) \rightarrow x \approx f_{R',Z}(f_{R,Y}(x))$	$\tilde{R}_Y \sqsubseteq S_{\{\tilde{R}_Y,R'_Z\}},\ R'_Z \sqsubseteq S_{\{\tilde{R}_Y,R'_Z\}},\ A \sqsubseteq \leq 1 S_{\{\tilde{R}_Y,R'_Z\}}.B,\ \tilde{R}_Y \equiv \mathsf{inv}(R_Y)$
$T19.$	$R(x,y) \rightarrow \overline{\bot}(x)$	$\exists R.\top \sqsubseteq \overline{\bot}$
$T20.$	$R(x,y) \rightarrow \overline{\bot}(y)$	$\top \sqsubseteq \forall R.\overline{\bot}$

Table 3. Transformation Ψ from transposed rules to DLs. Role names $\tilde{R}$ are fresh for every R, and $S_{\{R,R'\}}$ for every $\{R,R'\}$.

Lemma 12. *Let $\mathcal{O}$ be an ontology and M a minimal marking of $\xi(\mathcal{O})$. Then $\Xi_M(\xi(\mathcal{O}))$ contains only Horn rules of type T1–T2 and T4–T6 in Table 1 as well as type T7–T20 in Table 3.*

We can now specify a transformation Ψ that allows us to translate rules T7–T20 in Table 3 back into DL axioms.

Definition 13. *We define Ψ as the transformation mapping (i) each Horn rule r of type T1–T2 and T4–T6 in Table 1 to the DL axiom $\pi^{-1}(r)$ (ii) each rule T7–T20 on the left-hand side of Table 3 to the axioms on the right-hand side.*[2]

Intuitively, Ψ works as follows: *(i)* Function-free rules are "rolled up" as usual into DL axioms (see e.g., T7). *(ii)* Unary atoms $A(f_{R,Y}(x))$ with $A \neq \overline{\bot}$ that involve a functional term are translated as existentially or universally quantified concepts depending on whether they occur in the body or in the head (e.g., T10, T11); in contrast, atoms $\overline{\bot}(f_{R,Y}(x))$ in rules $\overline{\bot}(x) \rightarrow \overline{\bot}(f_{R,Y}(x))$ are translated as $\exists R_Y.\overline{\bot}$ instead of $\forall R_Y.\overline{\bot}$ (T9). *(iii)* Rules T15–T18, which involve $\approx$ in the head and roles R' and R in the body, are rolled back into axioms of type T6 over the "union" of R and R', which is captured using fresh roles and role inclusions.

The ontology obtained by applying Ψ to our running example is given in the last column of Table 2. Correctness of Ψ and its implications for the computation of Horn rewritings are summarised in the following lemma.

[2] For succinctness, axioms resulting from T7, T8, T12, T13, T14, T16 and T18 are not given in normal form.

Lemma 14. *Let $\mathcal{O}$ be a markable $\mathcal{ALCHIF}$ ontology and let M be a marking of $\mathcal{O}$. Then the ontology $\Psi(\Xi_M(\xi(\mathcal{O})))$ is a Horn rewriting of $\mathcal{O}$.*

A closer look at our transformations reveals that our rewritings do not introduce constructs such as inverse roles and cardinality restrictions if these were not already present in the input ontology. In contrast, fresh role inclusions may originate from cardinality restrictions in the input ontology. As a result, our approach is language-preserving: if the input $\mathcal{O}_1$ is in a DL $\mathcal{L}_1$ between $\mathcal{ALC}$ and $\mathcal{ALCHI}$, then its rewriting $\mathcal{O}_2$ stays in the Horn fragment of $\mathcal{L}_1$; furthermore, if $\mathcal{L}_1$ is between $\mathcal{ALCF}$ and $\mathcal{ALCIF}$, then $\mathcal{O}_2$ may contain fresh role inclusions ($\mathcal{H}$). A notable exception is when $\mathcal{O}_1$ is an $\mathcal{ELU}$ ontology, in which case axioms T2 and T3 in $\mathcal{O}_1$ may yield axioms of type T4 in $\mathcal{O}_2$. The following theorem follows from these observations and Lemma 14.

Theorem 15. *Every markable $\mathcal{L}$ ontology is polynomially Horn-$\mathcal{L}$ rewritable whenever $\mathcal{L}$ is between $\mathcal{ALC}$ and $\mathcal{ALCHI}$. If $\mathcal{L}$ is between $\mathcal{ALCF}$ and $\mathcal{ALCHIF}$, every markable $\mathcal{L}$ ontology is polynomially rewritable into Horn-$\mathcal{LH}$. Finally, every markable $\mathcal{ELU}$ ontology is polynomially rewritable into Horn-$\mathcal{ALC}$.*

We conclude by establishing the complexity of satisfiability checking over markable ontologies. We first show that the problem is ExpTime-hard for markable $\mathcal{ELU}$ ontologies, which implies that it is not possible to polynomially rewrite every markable $\mathcal{ELU}$ ontology into $\mathcal{EL}$. Thus, our rewriting approach is optimal for $\mathcal{ELU}$ in the sense that introducing universal restrictions (or equivalently inverse roles) in the rewriting is unavoidable.

Lemma 16. *Satisfiability checking over markable $\mathcal{ELU}$ is* ExpTime*-hard.*

All Horn DLs from $\mathcal{ALC}$ to $\mathcal{ALCHIF}$ are ExpTime-complete in combined complexity and PTime-complete in data complexity [15]. By Theorem 15, the same holds for markable ontologies in DLs from $\mathcal{ALC}$ to $\mathcal{ALCHIF}$. Finally, Lemma 16 shows that these results extend to markable $\mathcal{ELU}$ ontologies.

Theorem 17. *Let $\mathcal{L}$ be in-between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. Satisfiability checking over markable $\mathcal{L}$-ontologies is* ExpTime*-complete and* PTime*-complete in data.*

7 Related Work

Horn logics are common target languages for knowledge compilation [5]. Selman and Kautz [21] proposed an algorithm for compiling a set of propositional clauses into a set of Horn clauses s.t. their Horn consequences coincide. This approach was generalised to FOL by Del Val [6], without termination guarantees.

Bienvenu et al. [3] showed undecidability of Datalog rewritability for $\mathcal{ALCF}$ and decidability in NExpTime for $\mathcal{SHI}$. Cuenca Grau et al. [4] and Kaminski et al. [11] proposed practical techniques for computing Datalog rewritings of $\mathcal{SHI}$ ontologies based on a two-step process. First, $\mathcal{O}$ is rewritten using a resolution calculus Ω into a Disjunctive Datalog program $\Omega(\mathcal{O})$ of exponential

size [10]. Second, $\Omega(\mathcal{O})$ is rewritten into a Datalog program $\mathcal{P}$. For the second step, Kaminski et al. [11] propose the notion of markability of a Disjunctive Datalog program and show that $\mathcal{P}$ can be polynomially computed from $\Omega(\mathcal{O})$ using transposition whenever $\Omega(\mathcal{O})$ is markable. In contrast to our work, Kaminski et al. [11] focus on Datalog as target language for rewriting (rather than Horn DLs). Furthermore, their Datalog rewritings may be exponential w.r.t. the input ontology and cannot generally be represented in DLs.

Gottlob et al. [8] showed tractability in data complexity of fact entailment for the class of first-order rules with single-atom bodies, which is sufficient to capture most DLs in the DL-Lite$_{\mathsf{bool}}$ family [1].

Lutz and Wolter [17] investigated (non-uniform) data complexity of query answering w.r.t. *fixed* ontologies. They studied the boundary of PTime and co-NP-hardness and established a connection with constraint satisfaction problems. Finally, Lutz et al. [16] studied model-theoretic rewritability of ontologies in a DL $\mathcal{L}_1$ into a fragment $\mathcal{L}_2$ of $\mathcal{L}_1$. These rewritings are equivalence-preserving; this is in contrast to our approach, which requires only satisfiability preservation.

8 Proof of Concept

To assess the practical implications of our results, we have evaluated whether real-world ontologies are markable (and hence polynomially Horn rewritable). We analysed 120 non-Horn ontologies extracted from the Protege Ontology Library, BioPortal (http://bioportal.bioontology.org/), the corpus by Gardiner et al. [7], and the EBI linked data platform (http://www.ebi.ac.uk/rdf/platform). To check markability, we have implemented the 2-SAT reduction in Section 4 and a simple 2-SAT solver.

We found that a total of 32 ontologies were markable and thus rewritable into a Horn ontology, including some ontologies commonly used in applications, such as ChEMBL (see http://www.ebi.ac.uk/rdf/services/chembl/) and BioPAX Reactome (http://www.ebi.ac.uk/rdf/services/reactome/). When using π as first-order logic translation, we obtained 30 markable ontologies—a strict subset of the ontologies markable using ξ. However, only 27 ontologies were rewritable to a Horn DL since in three cases the marking contained a role.

9 Conclusion and Future Work

We have presented the first practical technique for rewriting non-Horn ontologies into a Horn DL. Our rewritings are polynomial, and our experiments suggest that they are applicable to widely-used ontologies. We anticipate several directions for future work. First, we would like to conduct an extensive evaluation to assess whether the use of our rewritings can significantly speed up satisfiability checking in practice. Second, we will investigate relaxations of markability that would allow us to capture a wider range of ontologies.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Tans. Database Syst. 39(4), 33 (2014)
4. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Computing datalog rewritings beyond Horn ontologies. In: IJCAI. pp. 832–838 (2013)
5. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. 17, 229–264 (2002)
6. Del Val, A.: First order LUB approximations: Characterization and algorithms. Artif. Intell. 162(1-2), 7–48 (2005)
7. Gardiner, T., Tsarkov, D., Horrocks, I.: Framework for an automated comparison of description logic reasoners. In: ISWC. pp. 654–667 (2006)
8. Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: MFCS. pp. 1–18 (2012)
9. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: IJCAI. pp. 466–471 (2005)
10. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
11. Kaminski, M., Nenov, Y., Cuenca Grau, B.: Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In: RR. pp. 76–91 (2014)
12. Kaminski, M., Nenov, Y., Cuenca Grau, B.: Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In: AAAI. pp. 1077–1083 (2014)
13. Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: KR. pp. 274–284 (2008)
14. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: LPAR. pp. 333–347 (2007)
15. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexities of Horn description logics. ACM Trans. Comput. Log. 14(1) (2013)
16. Lutz, C., Piro, R., Wolter, F.: Description logic TBoxes: Model-theoretic characterizations and rewritability. In: IJCAI. pp. 983–988 (2011)
17. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: KR. pp. 297–307 (2012)
18. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: IJCAI. pp. 1039–1044 (2011)
19. Robinson, A., Voronkov, A. (eds.): Handbook of Automated Reasoning. Elsevier (2001)
20. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: DL (2008)
21. Selman, B., Kautz, H.: Knowledge compilation and theory approximation. J. ACM 43(2), 193–224 (1996)

Reasoning Efficiently with Ontologies and Rules in the Presence of Inconsistencies (Extended Abstract)*

Tobias Kaminski, Matthias Knorr, and João Leite

NOVA LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa

In this paper, we address the problem of dealing with inconsistent knowledge bases consisting of ontologies and non-monotonic rules, following a paraconsistent reasoning approach with a focus on efficiency.

Description Logics (*DLs*) and *Logic Programs* (*LPs*) provide different strengths when used for *Knowledge Representation and Reasoning*. While DLs employ the *Open World Assumption* and are suited for defining ontologies, LPs adopt the *Closed World Assumption* and are able to express non-monotonic rules with exceptions and preference orders. Combining features of both formalisms has been actively pursued over the last few years, resulting in different proposals with different levels of integration and complexity: while some extend DLs with rules [18, 25], others follow a hybrid combination of ontologies with non-monotonic rules, either providing a modular approach where rules and ontologies use their own semantics, and allowing limited interaction between them [10], or defining a unifying framework for both components [29, 24]. Equipped with semantics that are faithful to their constitutive parts, these proposals allow for the specification of so-called *hybrid knowledge bases* (*hybrid KBs*) either from scratch, benefiting from the added expressivity, or by combining existing ontologies and rule bases.

The complex interactions between the ontology component and the rule component of these hybrid KBs – even more so when they result from combining existing ontologies and rule bases developed independently – can easily lead to contradictions, which, under classical semantics, trivialize standard reasoning and prevent us from drawing any meaningful conclusions, ultimately rendering these hybrid KBs useless.

Example 1. Consider the following simplified (ground) hybrid KB $\mathcal{K}_G$ for assessing the risk of goods at a port.

$$HasCertifiedSender \sqsubseteq \neg IsMonitored \tag{1}$$

$$\mathbf{K}IsMonitored(g) \leftarrow \mathbf{K}risk(g). \tag{2}$$

$$\mathbf{K}risk(g) \leftarrow \mathbf{not}\, isLabelled(g). \tag{3}$$

$$\mathbf{K}isLabelled(g) \leftarrow \mathbf{not}\, risk(g). \tag{4}$$

$$\mathbf{K}resolvedRisk(g) \leftarrow \mathbf{K}IsMonitored(g). \tag{5}$$

$$\mathbf{K}HasCertifiedSender(g) \leftarrow \tag{6}$$

$$\mathbf{K}risk(g) \leftarrow \tag{7}$$

* This is an extended abstract of [22]. Partially supported by Fundação para a Ciência e a Tecnologia under project PTDC/EIA-CCO/121823/2010 and strategic project PEst/UID/CEC/04516/2013. M. Knorr was also supported by grant SFRH/BPD/86970/2012.

Rules (3) and (4) state that good g is either a risk (r) or it is labeled (iL). Any risk is monitored (IM) (2), thus a resolved risk (rR) (5). As g has a certified sender (HCS) (6), it can be proven by means of axiom (1) that it is not monitored. Thus, g can be derived to be monitored and not monitored at the same time if it is considered to be a risk (7), i.e., the hybrid KB is inconsistent, which trivializes standard reasoning.

One way to deal with this problem is to employ some method based on belief revision (e.g. [26, 30, 35, 37, 9] for LPs, [14, 7, 23] for DLs, and [38, 36] for hybrid KBs) to regain consistency so that standard reasoning services can be used, or some method based on repairing (e.g. [5] for LPs, [17] for DLs, and [12, 11] for dl-programs [10]) where hypothetical belief revision is employed for consistent query answering, without actually changing the KB. However, this is not always feasible e.g. because, we may not have permission to change the KB – as for instance in [1] where the KB encodes laws and norms – or because the usual high complexity of belief revision and repairing methods simply renders their application prohibitive. When these methods are not possible or not feasible, paraconsistent reasoning services, typically based on some many-valued logic, offer an alternative by being able to draw meaningful conclusions in the presence of contradictions.

Paraconsistent reasoning has been extensively studied in both base formalisms of hybrid KBs. For DLs, most work [31, 39, 27, 41, 28] focuses on four-valued semantics varying which classical rules of inferences they satisfy. Among them, [27, 28] is most general as it covers $\mathcal{SROIQ}$, the DL behind OWL 2, considers tractable subclasses and truth value removals, and permits re-using classical reasoners. Three-valued semantics for DLs [40] and measuring the degree of inconsistency in *DL-Lite* [42] have also been considered. For LPs, the comprehensive survey [8] discusses e.g. a four-valued semantics without default negation [6], a four-, six-, and nine-valued semantics [34] for answer sets [16], and a seven- [33] and nine-valued [3] well-founded semantics [15]. More recently, a very general framework for arbitrary bilattices of truth values [2] and paraconsistent Datalog [4] have been considered. At the same time, paraconsistent reasoning is still a rather unexplored field in the context of hybrid KBs. Notable exceptions are [20, 19, 13], yet their computation is not tractable in general even if reasoning in the DL component is.

In this paper, we investigate efficient paraconsistent semantics for hybrid KBs. We adopt the base framework of [29] because of its generality and tight integration between the ontology and the rules – cf. [29] for a thorough argument in its favor – under the semantics of [24] because of its computational properties. We extend this semantics with additional truth values to evaluate contradictory pieces of knowledge, following two common views on how to deal with contradictory knowledge bases.

According to one view, contradictions are dealt with locally, in a minimally intrusive way, such that a new truth value is introduced to model inconsistencies, but non-contradictory knowledge only derivable from the inconsistent part of a KB is still considered to be true in the classical sense. This view is adopted in paraconsistent semantics for DLs, e.g. [28], LPs, e.g. [33, 34], and hybrid KBs [20, 13]. Since two different kinds of inconsistencies are identified in the three-valued semantics of [24], two further truth values are introduced when following this first approach in extending the work of [24], resulting in a five-valued semantics. Namely, we extend the set of truth

values *true* (**t**), *false* (**f**), and *undefined* (**u**) used in [24] by the truth value **b** for *both*, which is assigned whenever an atom is considered *true* and *false* at the same time, and the truth value **uf** for *undefined false*, which is used whenever an atom would be considered simultaneously *undefined* and *false*.

The alternative view is to distinguish truth which depends on the inconsistent part of a KB from truth which is derivable without involving any contradictory knowledge. This view, commonly referred to as *Suspicious Reasoning*, is adopted in paraconsistent semantics for LPs, e.g. [3, 33, 34] and hybrid KBs [19]. In order to extend the approach of [24] in a way that allows for paraconsistency in combination with Suspicious Reasoning, a sixth truth value *suspiciously true* (**st**) is introduced in addition to those already occurring in the five-valued semantics. This truth value is assigned to atoms only derivable by involving a contradiction in the program. At the same time, the truth value **uf** is replaced by the slightly different truth value *classically false* (**cf**), with the aim to also capture "propagation" on derived classical falsity.

As a result, we obtain solutions following both views through the definition of a five-valued and a six-valued paraconsistent semantics for hybrid KBs, the latter implementing Suspicious Reasoning. This requires the integration of quite different concepts and assumptions w.r.t. paraconsistency developed independently for each of the two base formalisms, e.g. Suspicious Reasoning has not been considered in DLs, while LP semantics may sometimes be defined procedurally. In spite of these obstacles, we can show that both of the resulting semantics enjoy a number of desirable properties.

- Firstly, both semantics are sound w.r.t. the three-valued semantics for consistent hybrid KBs by [24]. In fact, the so-called 5- and 6-models corresponding to models in [24] coincide in this case, so consistent hybrid KBs establish a link between our two semantics.
- Secondly, the semantics assigned to a hybrid KB of which the program component is empty is limited, in both cases, to only three truth values (**t**, **f**, and **b**), which arguably leads to a stronger consequence relation than in common four-valued paraconsistent DL semantics [32]. Still, we can show that, in this case, both semantics coincide with the well-known paraconsistent DL semantics $\mathcal{ALC}4$ by [28] if we omit the truth value **u** (referred to as "removal of gaps"). Moreover, we show that the six-valued semantics is faithful w.r.t. the paraconsistent semantics for extended logic programs $WFSX_p$ [3] when classical negation is only applied to unary atoms. Consequently, properties shown for these paraconsistent semantics for the two base formalisms directly carry over to our approach, e.g. it implements the *Coherence Principle*, which states that classical negation implies default negation.
- Thirdly, we present a sound and complete fixpoint algorithm, which extends the alternating fixpoint construction defined for the three-valued approach in [24]. The algorithm preserves the efficiency of the previous approach in that it is tractable whenever consequences in the DL used for formalizing the ontology component can be computed in polynomial time.

Finally, our approach and results can benefit existing implementations for hybrid knowledge bases. In fact, the comparison between our two fixpoint computations and that in [24] suggest an adaptation of the implementation of the latter, the Protégé plug-in *NoHR* [21], to also consider paraconsistent reasoning based on our semantics.

References

1. Alberti, M., Gomes, A.S., Gonçalves, R., Leite, J., Slota, M.: Normative systems represented as hybrid knowledge bases. In: Procs. of CLIMA XII. Springer (2011)
2. Alcântara, J., Damásio, C.V., Pereira, L.M.: An encompassing framework for paraconsistent logic programs. J. Applied Logic 3(1), 67–95 (2005)
3. Alferes, J.J., Damásio, C.V., Pereira, L.M.: A logic programming system for nonmonotonic reasoning. J. Autom. Reasoning 14(1), 93–147 (1995)
4. de Amo, S., Pais, M.S.: A paraconsistent logic programming approach for querying inconsistent databases. Int. J. Approx. Reasoning 46(2), 366–386 (2007)
5. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Procs of PODS. ACM Press (1999)
6. Blair, H.A., Subrahmanian, V.S.: Paraconsistent logic programming. Theor. Comput. Sci. 68(2), 135–154 (1989)
7. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of DL-Lite knowledge bases. In: Procs. of ISWC. Springer (2010)
8. Damásio, C.V., Pereira, L.M.: A survey of paraconsistent semantics for logic programs. In: Reasoning with Actual and Potential Contradictions. Springer (1998)
9. Delgrande, J., Schaub, T., Tompits, H., Woltran, S.: A model-theoretic approach to belief change in answer set programming. ACM TOCL 14(2), 14 (2013)
10. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artif. Intell. 172(12-13), 1495–1539 (2008)
11. Eiter, T., Fink, M., Stepanova, D.: Computing repairs for inconsistent dl-programs over *EL* ontologies. In: Procs. of JELIA. Springer (2014)
12. Eiter, T., Fink, M., Stepanova, D.: Towards practical deletion repair of inconsistent dl-programs. In: Procs. of ECAI. IOS Press (2014)
13. Fink, M.: Paraconsistent hybrid theories. In: Procs of KR. AAAI Press (2012)
14. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. Knowledge Eng. Review 23(2), 117–152 (2008)
15. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)
16. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
17. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: Procs. of ISWC. Springer (2005)
18. Horrocks, I., Patel-Schneider, P.: A proposal for an OWL rules language. In: Procs of WWW. ACM (2004)
19. Huang, S., Hao, J., Luo, D.: Incoherency problems in a combination of description logics and rules. J. Applied Mathematics 2014 (2014)
20. Huang, S., Li, Q., Hitzler, P.: Paraconsistent semantics for hybrid MKNF knowledge bases. In: Procs of RR. Springer (2011)
21. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Procs. of ISWC. Springer (2013)
22. Kaminski, T., Knorr, M., Leite, J.: Efficient paraconsistent reasoning with ontologies and rules. In: Procs. of IJCAI. AAAI press (2015), to appear
23. Kharlamov, E., Zheleznyakov, D., Calvanese, D.: Capturing model-based ontology evolution at the instance level: The case of dl-lite. Journal of Computer and System Sciences 79(6), 835–872 (2013)

24. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9-10), 1528–1554 (2011)
25. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Procs. of WWW. ACM (2011)
26. Leite, J.: Evolving Knowledge Bases. IOS Press (2003)
27. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with OWL. In: Procs. of ESWC. Springer (2007)
28. Maier, F., Ma, Y., Hitzler, P.: Paraconsistent OWL and related logics. Semantic Web 4(4), 395–427 (2013)
29. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
30. Osorio, M., Cuevas, V.: Updates in answer set programming: An approach based on basic structural properties. TPLP 7(4), 451–479 (2007)
31. Patel-Schneider, P.: A four-valued semantics for terminological logics. Artif. Intell. 38(3), 319–351 (1989)
32. Priest, G.: The logic of paradox. Journal of Philosophical logic 8(1), 219–241 (1979)
33. Sakama, C.: Extended well-founded semantics for paraconsistent logic programs. In: Procs. of FGCS. IOS Press (1992)
34. Sakama, C., Inoue, K.: Paraconsistent stable semantics for extended disjunctive programs. J. Log. Comput. 5(3), 265–285 (1995)
35. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Procs. of KR. AAAI Press (2012)
36. Slota, M., Leite, J.: A unifying perspective on knowledge updates. In: Procs. of JELIA. Springer (2012)
37. Slota, M., Leite, J.: The rise and fall of semantic rule updates based on se-models. TPLP 14(6), 869–907 (2014)
38. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP 11(4-5), 801–819 (2011)
39. Straccia, U.: A sequent calculus for reasoning in four-valued description logics. In: Procs. of TABLEAUX. Springer (1997)
40. Zhang, X., Lin, Z., Wang, K.: Towards a paradoxical description logic for the semantic web. In: Procs. of FoIKS. Springer (2010)
41. Zhang, X., Xiao, G., Lin, Z.: A tableau algorithm for handling inconsistency in OWL. In: Procs. of ESWC. Springer (2009)
42. Zhou, L., Huang, H., Qi, G., Ma, Y., Huang, Z., Qu, Y.: Paraconsistent query answering over DL-Lite ontologies. Web Intelligence and Agent Systems 10(1), 19–31 (2012)

Advancing ELK: Not Only Performance Matters

Yevgeny Kazakov and Pavel Klinov

The University of Ulm, Germany
{yevgeny.kazakov, pavel.klinov}@uni-ulm.de

Abstract. This paper reports on the recent development of ELK, a consequence-based reasoner for $\mathcal{EL}^+_\bot$ ontologies. It covers novel reasoning techniques which aim at improving efficiency and providing foundation for new reasoning services. On the former front we present a simple optimization for handling of role composition axioms, such as transitivity, which substantially reduces the number of rule applications. For the latter, we describe a new rule application strategy that takes advantage of concept definitions to avoid many redundant inferences without making rules dependent on derived conclusions. This improvement is not visible to the end user but considerably simplifies implementation for incremental reasoning and proof generation. We also present a rewriting of low-level inferences used by ELK to higher-level proofs that can be defined in the standard DL syntax, and thus be used for automatic verification of reasoning results or (visual) ontology debugging. We demonstrate the latter capability using a new ELK Protégé plugin.

1 Introduction

ELK is an ontology reasoner designed for top classification performance on OWL EL ontologies [1]. Its characteristic features are consequence-based calculus, highly parallelizable reasoning, and aggressive optimizations to reduce the number of derived axioms sufficient for classification (deriving all subsumptions between concept names).

Sheer performance has been the sole goal for the first few versions of ELK and it enabled it to become the reasoner of choice in biomedical circles where large $\mathcal{EL}$ ontologies are built to manage scientific terminologies [2–7]. After that ELK started to evolve towards providing additional reasoning-related services, such as incremental classification [8] and proof tracing [9]. It turned out that some traits of ELK's classification procedure, in particular, the non-deterministic saturation, can complicate the development or weaken the guarantees of such extra services. For example, the composition/decomposition optimization (cf. [1]) has to be off when incrementally retracting inferences [8]. Also, the proof tracing method guarantees only that all proofs performed by ELK will be generated, not all proofs supported by the calculus [9]. This, in particular, means that one cannot in general obtain *all* justifications [10] from proofs.

In this paper we describe the steps towards adapting the main reasoning procedure to rectify this sort of issues without major performance setbacks. On the performance front we show a technique to reduce the number of inferences on roles. We also present a rewriting of the low-level traced inferences into higher-level proof-based explanations which could be shown to the user or verified using automated reasoning tools. Due to the space constraints, some results are deferred to the technical report [11].

$$\mathsf{E}_0\ \frac{}{C \sqsubseteq C} \qquad \mathsf{E}^+_\sqcap\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2}$$

$$\mathsf{E}_\top\ \frac{}{C \sqsubseteq \top} \qquad \mathsf{E}_\bot\ \frac{C \sqsubseteq \exists R.D \quad D \sqsubseteq \bot}{C \sqsubseteq \bot}$$

$$\mathsf{E}_\sqsubseteq\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O} \qquad \mathsf{E}_\exists\ \frac{C \sqsubseteq \exists R.D \quad D \sqsubseteq E}{C \sqsubseteq \exists R.E}$$

$$\mathsf{E}^-_\sqcap\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2} \qquad \mathsf{E}_\circ\ \frac{C_{i-1} \sqsubseteq \exists R_i.C_i \quad (1 \leq i \leq k,\ k \geq 0)}{C_0 \sqsubseteq \exists R.C_k} : R_1 \cdots R_k \sqsubseteq R \in \mathcal{O}$$

Fig. 1. Basic inference rules for reasoning in $\mathcal{EL}^+_\bot$

2 Consequence-Based Reasoning in $\mathcal{EL}^+_\bot$

We first describe ELK's consequence-based procedure for $\mathcal{EL}^+_\bot$ reasoning. Most theoretical results, such as completeness, redundancy elimination, and goal-directed rule application, are minor variations of [1] but the rules for dealing with role chain axioms without binarization and rules for reasoning with reflexive roles are new.

2.1 The Description Logic $\mathcal{EL}^+_\bot$

The syntax of $\mathcal{EL}^+_\bot$ is defined using a vocabulary consisting of countably infinite sets of *(atomic) roles* and *atomic concepts*. $\mathcal{EL}^+_\bot$ *concepts* are defined using the grammar $\mathbf{C} ::= A \mid \top \mid \bot \mid C_1 \sqcap C_2 \mid \exists R.C$, where A is an atomic concept, R a role, and $C_{(i)} \in \mathbf{C}$. $\mathcal{EL}^+_\bot$ *role chains* are defined using the grammar $\mathbf{P} ::= \epsilon \mid R{\cdot}\rho$, where ϵ is the *empty role chain*, R a role and $\rho \in \mathbf{P}$. We usually write role chains as $R_1{\cdot}R_2 \cdots R_n$ instead of $R_1{\cdot}(R_2 \cdots (R_n{\cdot}\epsilon))$. An $\mathcal{EL}^+_\bot$ *axiom* is either a *concept inclusion* $C_1 \sqsubseteq C_2$ for $C_1, C_2 \in \mathbf{C}$ or a *role inclusion* $\rho \sqsubseteq R$ for $\rho \in \mathbf{P}$ and a role R. We regard the *concept equivalence* $C_1 \equiv C_2$ as an abbreviation for two concept inclusions $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. We also call $\epsilon \sqsubseteq R$ a *role reflexivity axiom*. An $\mathcal{EL}^+_\bot$ *ontology* $\mathcal{O}$ is a finite set of $\mathcal{EL}^+_\bot$ axioms. Semantics of $\mathcal{EL}^+_\bot$ is defined in the usual way (ϵ is interpreted as identity). A concept C is *subsumed* by D w.r.t. $\mathcal{O}$ if $\mathcal{O} \models C \sqsubseteq D$. In this case, we call $C \sqsubseteq D$ an *entailed subsumption* (w.r.t. $\mathcal{O}$). The *ontology classification task* requires to compute all entailed subsumptions between atomic concepts occurring in $\mathcal{O}$.

2.2 Inference Rules

Classification of $\mathcal{EL}^+_\bot$ ontologies is usually performed by applying rules that derive logical consequences of axioms. Figure 1 lists the $\mathcal{EL}^+_\bot$-rules that are similar to those usually considered in the literature [12, 13]. The *premises* of the rules are written above the horizontal line, the *conclusions* below, and the axioms in the ontology (a.k.a. *side conditions*) that trigger rule applications after the colon. Note that rule $\mathsf{E}_\circ$ can be used with $k = 0$, in which case it has no premises and uses the reflexivity axiom $\epsilon \sqsubseteq R \in \mathcal{O}$.

Example 1. Consider the $\mathcal{EL}^+_\bot$ ontology $\mathcal{O} = \{A \sqsubseteq \exists R.B,\ B \sqsubseteq \exists S.C,\ R{\cdot}S{\cdot}H \sqsubseteq V,\ \epsilon \sqsubseteq H\}$. Then it is possible to derive $A \sqsubseteq \exists V.C$ using the rules in Figure 1 as follows:

$A \sqsubseteq A$ by $\mathbf{E_0}()$, (1)

$A \sqsubseteq \exists R.B$ by $\mathbf{E}_{\sqsubseteq}(A \sqsubseteq A) : A \sqsubseteq \exists R.B \in \mathcal{O}$, (2)

$B \sqsubseteq B$ by $\mathbf{E_0}()$, (3)

$B \sqsubseteq \exists S.C$ by $\mathbf{E}_{\sqsubseteq}(B \sqsubseteq B) : B \sqsubseteq \exists S.C \in \mathcal{O}$, (4)

$C \sqsubseteq \exists H.C$ by $\mathbf{E}_{\circ}() : \epsilon \sqsubseteq H \in \mathcal{O}$, (5)

$A \sqsubseteq \exists V.C$ by $\mathbf{E}_{\circ}(A \sqsubseteq \exists R.B, B \sqsubseteq \exists S.C, C \sqsubseteq \exists H.C) : R{\cdot}S{\cdot}H \sqsubseteq V \in \mathcal{O}$. (6)

Formally, a *derivation* for $\mathcal{EL}$ ontology $\mathcal{O}$ (using the rules in Figure 1) is a sequence of $\mathcal{EL}^+_\bot$ axioms $d = \{\alpha_i \mid i \geq 1\}$ such that each α_i with $i \geq 0$ is obtained from axioms $\{\alpha_j \mid 1 \leq j < i\}$ using one of the rules in Figure 1 and axioms in $\mathcal{O}$ as side conditions. The size $\|d\|$ of d is the number of axioms in d. For example, the sequence of axioms (1)–(6) in Example 1 forms a derivation, in which every axiom is obtained from the previous axioms by the rules in Figure 1 as indicated next to the axioms.

The rules in Figure 1 are simple to understand but not very efficient to implement. The problem is caused by rule $\mathbf{E}_{\circ}$, which may produce many conclusions for ontologies with deep role hierarchies. For example, consider $\mathcal{O} = \{R_{j-1} \sqsubseteq R_j \mid 1 \leq j \leq m\} \cup \{C_{i-1} \sqsubseteq \exists R_0.C_i,\ \exists R_m.D_i \sqsubseteq D_{i-1} \mid 1 \leq i \leq n\} \cup \{C_n \sqsubseteq D_n\}$. Then one can only derive $C_0 \sqsubseteq D_0$ by the rules in Figure 1 by deriving quadratically-many intermediate axioms $C_{i-1} \sqsubseteq \exists R_j.C_i$ by $\mathbf{E}_{\circ}$ using $R_{j-1} \sqsubseteq R_j \in \mathcal{O}$ ($1 \leq i \leq n$, $1 \leq j \leq m$). Therefore, ELK implements optimized rules listed in Figure 2 that help avoiding this problem by deriving subsumptions on role (chains) separately [1]. To formulate these rules, we have slightly extended the syntax of $\mathcal{EL}^+_\bot$. First, we can derive role chains on the right-hand side of role inclusions: $\rho_1 \sqsubseteq \rho_2$ ($\mathcal{I} \models \rho_1 \sqsubseteq \rho_2$ if $\rho_1^{\mathcal{I}} \subseteq \rho_2^{\mathcal{I}}$). Second, we allow role chains to occur in existential restrictions: $\exists(R{\cdot}\rho).C$ is rewritten to $\exists R.C$ if $\rho = \epsilon$, or to $\exists R.\exists\rho.C$ otherwise (whenever we write $\exists\rho.C$ we assume that $\rho \neq \epsilon$). The *extended* $\mathcal{EL}^+_\bot$ *axioms* can be used in derivations, but not in the ontology $\mathcal{O}$.

Example 2. Below is the derivation for $A \sqsubseteq \exists V.C$ for the ontology $\mathcal{O}$ in Example 1 using the rules in Figure 2:

$A \sqsubseteq A$ by $\mathbf{C_0}()$, (7)

$A \sqsubseteq \exists R.B$ by $\mathbf{C}_{\sqsubseteq}(A \sqsubseteq A) : A \sqsubseteq \exists R.B \in \mathcal{O}$, (8)

$R \sqsubseteq R$ by $\mathbf{R_0}()$, (9)

$B \sqsubseteq B$ by $\mathbf{C_0}()$, (10)

$B \sqsubseteq \exists S.C$ by $\mathbf{C}_{\sqsubseteq}(B \sqsubseteq B) : B \sqsubseteq \exists S.C \in \mathcal{O}$, (11)

$S \sqsubseteq S$ by $\mathbf{R_0}()$, (12)

$C \sqsubseteq C$ by $\mathbf{C_0}()$, (13)

$\epsilon \sqsubseteq \epsilon$ by $\mathbf{R_0}()$, (14)

$\epsilon \sqsubseteq H$ by $\mathbf{R}_{\sqsubseteq}(\epsilon \sqsubseteq \epsilon) : \epsilon \sqsubseteq H \in \mathcal{O}$, (15)

$S \sqsubseteq S{\cdot}H$ by $\mathbf{R}^{\epsilon}_{\mathbf{r}}(S \sqsubseteq S, \epsilon \sqsubseteq H)$, (16)

$A \sqsubseteq \exists(R{\cdot}S{\cdot}H).C$ by $\mathbf{C}_{\circ}(A \sqsubseteq \exists R.B, R \sqsubseteq R, B \sqsubseteq \exists S.C, S \sqsubseteq S{\cdot}H)$, (17)

$$\mathbf{R_0}\ \frac{}{\rho \sqsubseteq \rho} \qquad \mathbf{R}^{\epsilon}_{\mathsf{l}}\ \frac{\epsilon \sqsubseteq R \quad \rho_1 \sqsubseteq \rho_2}{\rho_1 \sqsubseteq R{\cdot}\rho_2}$$

$$\mathbf{R}_{\sqsubseteq}\ \frac{\rho_1 \sqsubseteq \rho_2}{\rho_1 \sqsubseteq R} : \rho_2 \sqsubseteq R \in \mathcal{O} \qquad \mathbf{R}^{\epsilon}_{\mathsf{r}}\ \frac{\rho_1 \sqsubseteq R \quad \epsilon \sqsubseteq \rho_2}{\rho_1 \sqsubseteq R{\cdot}\rho_2}$$

$$\mathbf{C_0}\ \frac{}{C \sqsubseteq C} \qquad \mathbf{C}_{\bot}\ \frac{C \sqsubseteq \exists \rho.D \quad D \sqsubseteq \bot}{C \sqsubseteq \bot}$$

$$\mathbf{C}_{\top}\ \frac{}{C \sqsubseteq \top} \qquad \mathbf{C}^{\epsilon}_{\exists}\ \frac{C \sqsubseteq D \quad \epsilon \sqsubseteq R}{C \sqsubseteq \exists R.D}$$

$$\mathbf{C}_{\sqsubseteq}\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O} \qquad \mathbf{C}_{\exists}\ \frac{C \sqsubseteq \exists \rho.D \quad \rho \sqsubseteq R \quad D \sqsubseteq E}{C \sqsubseteq \exists R.E}$$

$$\mathbf{C}^{-}_{\sqcap}\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2} \qquad \mathbf{C}_{\circ}\ \frac{C \sqsubseteq \exists \rho_1.D \quad \rho_1 \sqsubseteq R \quad D \sqsubseteq \exists \rho_2.E \quad \rho_2 \sqsubseteq \rho}{C \sqsubseteq \exists (R{\cdot}\rho).E}$$

$$\mathbf{C}^{+}_{\sqcap}\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2}$$

Fig. 2. Optimized inference rules for reasoning in $\mathcal{EL}^+_\bot$ implemented in ELK

$$R{\cdot}S{\cdot}H \sqsubseteq R{\cdot}S{\cdot}H \qquad \text{by } \mathbf{R_0}(), \tag{18}$$

$$R{\cdot}S{\cdot}H \sqsubseteq V \qquad \text{by } \mathbf{R}_{\sqsubseteq}(R{\cdot}S{\cdot}H \sqsubseteq R{\cdot}S{\cdot}H) : R{\cdot}S{\cdot}H \sqsubseteq V \in \mathcal{O}, \tag{19}$$

$$A \sqsubseteq \exists V.C \qquad \text{by } \mathbf{C}_{\exists}(A \sqsubseteq \exists (R{\cdot}S{\cdot}H).C, R{\cdot}S{\cdot}H \sqsubseteq V, C \sqsubseteq C). \tag{20}$$

As can be seen from Examples 1 and 2, derivations using the rules in Figure 2 can be more difficult to understand because they use relatively complex rules such as $\mathbf{R}^{\epsilon}_{\mathsf{r}}$ and $\mathbf{C}_{\circ}$ and manipulate with extended $\mathcal{EL}^+$ axioms such as $A \sqsubseteq \exists (R{\cdot}S{\cdot}H).C$ and $S \sqsubseteq S{\cdot}H$, the last of which is not even expressible in $\mathcal{EL}^+$. Fortunately, it is always possible to rewrite any derivation by the rules in Figure 2 into the one by the rules in Figure 1 using a simple recursive procedure (with an unavoidable quadratic blowup):

Theorem 1. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology, d a derivation by the rules in Figure 2 for $\mathcal{O}$, and $F \sqsubseteq G \in d$ an (ordinary) $\mathcal{EL}^+_\bot$ axiom. Then one can construct a derivation e by the rules in Figure 1 for $\mathcal{O}$ with $F \sqsubseteq G \in e$ such that $\|e\| = O(\|d\|^2)$.*

The proof of Theorem 1 can be found in the technical report [11], which also contains an overview of the Protégé plug-in for displaying proofs based on this result.

2.3 Composed Conclusions, Redundancy and Completeness

From now on, we focus in the rules in Figure 2, so when we talk about axioms derived by these rules, we mean extended $\mathcal{EL}^+_\bot$ axioms. It is easy to see that the rules in Figure 2 are *sound*, that is, the conclusions of the rules are logical consequences of the premises and the side conditions, if there are any. Therefore, every derivation contains only axioms entailed by the ontology. It turns out that the converse property also holds:

Theorem 2. *Let $\mathcal{O}$ be an $\mathcal{EL}^+_\bot$ ontology and $F \sqsubseteq G$ an $\mathcal{EL}^+_\bot$ concept inclusion such that $\mathcal{O} \models F \sqsubseteq G$. Then there exists a derivation d using the rules in Figure 2 such that either $F \sqsubseteq G \in d$ or $F \sqsubseteq \bot \in d$.*

Similarly to existing results [12, 1], Theorem 2 is proved by constructing a canonical model using the set of all derivable axioms. One can actually prove a stronger version of this theorem, namely that every entailed subsumption is derivable by an optimized derivation—a derivation which does not use a certain kind of redundant inferences [11].

Definition 1. *We say that an axiom α in a derivation d is* composed *if can be obtained by rules $\mathsf{C}_\sqcap^+$, $\mathsf{C}_\exists^\epsilon$ or $\mathsf{C}_\exists$ from the previous axioms. An application of a rule in Figure 2 to premises in d is* redundant *if it is an application by rule $\mathsf{C}_\sqcap^-$ in which the premise is composed, by rule $\mathsf{C}_\exists$ in which the first premise is composed, or by rule $\mathsf{C}_\circ$ in which the first or the third premise is composed. The derivation d is* optimized *if every axiom α in d is obtained from the previous axioms using a non-redundant rule application.*

Example 3. Consider the ontology $\mathcal{O} = \{A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ C \sqsubseteq D\}$. Then the following derivation using the rules in Figure 2 is possible:

$$
\begin{array}{lll}
 & A \sqsubseteq A & \text{by } \mathsf{C_0}(), \quad (21)\\
 & A \sqsubseteq \exists R.B & \text{by } \mathsf{C}_\sqsubseteq(A \sqsubseteq A) : A \sqsubseteq \exists R.B \in \mathcal{O}, \quad (22)\\
 & B \sqsubseteq B & \text{by } \mathsf{C_0}(), \quad (23)\\
 & B \sqsubseteq C & \text{by } \mathsf{C}_\sqsubseteq(B \sqsubseteq B) : B \sqsubseteq C \in \mathcal{O}, \quad (24)\\
 & B \sqsubseteq D & \text{by } \mathsf{C}_\sqsubseteq(B \sqsubseteq C) : C \sqsubseteq D \in \mathcal{O}, \quad (25)\\
 & R \sqsubseteq R & \text{by } \mathsf{R_0}(), \quad (26)\\
(+) & A \sqsubseteq \exists R.C & \text{by } \mathsf{C}_\exists(A \sqsubseteq \exists R.B, R \sqsubseteq R, B \sqsubseteq C), \quad (27)\\
 & C \sqsubseteq C & \text{by } \mathsf{C_0}(), \quad (28)\\
 & C \sqsubseteq D & \text{by } \mathsf{C}_\sqsubseteq(C \sqsubseteq C) : C \sqsubseteq D \in \mathcal{O}, \quad (29)\\
(+) & A \sqsubseteq \exists R.D & \text{by } \mathsf{C}_\exists(A \sqsubseteq \exists R.C, R \sqsubseteq R, C \sqsubseteq D). \quad (30)
\end{array}
$$

In this derivation, axioms (27) and (30) (labeled with $+$) are composed because they were obtained by rule $\mathsf{C}_\exists$. Hence the inference that has produced (30) is redundant because it is made by an application of $\mathsf{C}_\exists$ to a composed first premise (27). Still, the derivation (21)–(30) is optimized because (30) *can* be obtained from the previous axioms by another (non-redundant) application of $\mathsf{C}_\exists$ to a non-composed premise (22):

$$
(+) \quad A \sqsubseteq \exists R.D \qquad \text{by } \mathsf{C}_\exists(A \sqsubseteq \exists R.B, R \sqsubseteq R, B \sqsubseteq D). \quad (31)
$$

In other words, since a derivation is a sequence of axioms and not a sequence of rule applications, it matters by which inferences axioms *can* be obtained from the previous axioms. Note that if (25) is removed from the derivation, then the inference (31) is no longer possible and the derivation becomes non-optimized.

In practice, the optimization above means that when applying the rules in Figure 2 to check entailment of concept inclusion, it is not necessary to apply $\mathsf{C}_\sqcap^-$ to premises derived by $\mathsf{C}_\sqcap^+$ or apply $\mathsf{C}_\exists$ and $\mathsf{C}_\circ$ to premises derived by $\mathsf{C}_\exists$ [1].

2.4 The Subformula Property and Goal-Directed Rule Application

So far Theorem 2 cannot be used for effectively checking if a given subsumption $C \sqsubseteq D$ is entailed by the ontology $\mathcal{O}$ since there are infinitely many axioms that can be

derived by the rules in Figure 2—already $\mathbf{C_0}$ can produce infinitely many conclusions. It turns out, it is sufficient to derive only axioms of the form $\rho_1 \sqsubseteq \rho_2$, $C_1 \sqsubseteq C_2$, and $C_1 \sqsubseteq \exists\rho_2.C_2$ such that all concepts C_i and role chains ρ_i ($i = 1, 2$) occur (possibly as sub-expressions) either in the ontology $\mathcal{O}$ or in the given subsumption $F \sqsubseteq G$ tested for entailment [1]. In other words, if $\mathcal{O} \models F \sqsubseteq G$ then $F \sqsubseteq G$ or $F \sqsubseteq \bot$ is derivable by the rules in Figure 2 without creating new concepts or role chains. We refer to this property as *subformula property*. The subformula property implies that checking entailment $\mathcal{O} \models F \sqsubseteq G$ can be done in polynomial time since there are at most polynomially-many different axioms of the above forms, and one can compute a derivation d containing all such axioms by repeatedly applying the rules in Figure 2.

The rules, however, can be restricted even further. Specifically, an axiom $C_1 \sqsubseteq C_2$ needs to be derived in d only if $C_1 = F$ for the tested subsumption $F \sqsubseteq G$ or if some non-composed axiom of the form $C \sqsubseteq \exists\rho.C_1$ is already derived in d. Indeed, it is easy to observe from the rules in Figure 2 that an axiom $C_1 \sqsubseteq C_2$ can be used in a rule application only if this rule derives a concept inclusion axiom with the same left-hand side C_1 or it uses another non-composed axiom of the form $C \sqsubseteq \exists\rho.C_1$ (rules $\mathbf{C_\exists}$ and $\mathbf{C_\circ}$). Similarly, an axiom $\rho_1 \sqsubseteq \rho_2$ needs to be derived only if $\rho_1 = \epsilon$ or if some other non-composed axiom of the form $C \sqsubseteq \exists\rho_1.D$ is already derived. We call this optimization *goal-directed rule application*.

Example 4. Consider the ontology $\mathcal{O} = \{A \sqsubseteq \exists R.B,\ A \sqsubseteq \exists R.C,\ B \sqsubseteq C,\ C \sqsubseteq B\}$. Suppose we want to check whether $\mathcal{O} \models A \sqsubseteq B$. Then the following goal-directed non-redundant rule applications can be performed:

$$
\begin{array}{lll}
 & A \sqsubseteq A & \text{by } \mathbf{C_0}(), \qquad (32)\\
 & A \sqsubseteq \exists R.B & \text{by } \mathbf{C_\sqsubseteq}(A \sqsubseteq A) : A \sqsubseteq \exists R.B \in \mathcal{O}, \qquad (33)\\
 & B \sqsubseteq B & \text{by } \mathbf{R_0}(), \qquad (34)\\
 & B \sqsubseteq C & \text{by } \mathbf{R_\sqsubseteq}(B \sqsubseteq B) : B \sqsubseteq C \in \mathcal{O}, \qquad (35)\\
 & R \sqsubseteq R & \text{by } \mathbf{R_0}(), \qquad (36)\\
(+) & A \sqsubseteq \exists R.C & \text{by } \mathbf{C_\exists}(A \sqsubseteq \exists R.B,\ R \sqsubseteq R,\ B \sqsubseteq C), \qquad (37)\\
(+) & A \sqsubseteq \exists R.C & \text{by } \mathbf{C_\sqsubseteq}(A \sqsubseteq A) : A \sqsubseteq \exists R.C \in \mathcal{O}. \qquad (38)
\end{array}
$$

Note that deriving axioms with the left-hand side C (e.g., $C \sqsubseteq C$ by rule $\mathbf{C_0}$) is not necessary since the axiom $A \sqsubseteq \exists R.C$ is composed (and thus, e.g., cannot be used in rule $\mathbf{C_\exists}$ like axiom $A \sqsubseteq \exists R.B$). Since no further rules need to be applied and the axiom $A \sqsubseteq B$ is not derived, we can conclude that $\mathcal{O} \not\models A \sqsubseteq B$. Note that if we swap (33) with (38), then the axiom $A \sqsubseteq \exists R.C$ would not be composed and we would need to derive subsumptions $C \sqsubseteq C$ and $C \sqsubseteq B$ by rules $\mathbf{C_0}$ and $\mathbf{C_\sqsubseteq}$. Thus the set of derived axioms depends on the order in which the rules are applied.

Note that if a derivation d contains $C \sqsubseteq D$ or $\rho_1 \sqsubseteq \rho_2$ then $C \sqsubseteq C$ or $\rho_1 \sqsubseteq \rho_1$ must be derived in d respectively by $\mathbf{C_0}$ and $\mathbf{R_0}$ before that. Therefore, to save space, from now on we skip applications of the rules $\mathbf{C_0}$ and $\mathbf{R_0}$ (e.g., like (32), (34), and (36) in Example 4). We will also skip application of the rules $\mathbf{C_\sqsubseteq}$ and $\mathbf{R_\sqsubseteq}$ producing axioms in the ontology from the conclusions of $\mathbf{C_0}$ and $\mathbf{R_0}$ (e.g., like (33) and (35) in Example 4).

3 Avoiding Duplicate Role Compositions

In this section, we present an optimization, using which one can avoid some duplicate conclusions of rule $\mathbf{C}_\circ$. Intuitively, this optimization is designed to deal more efficiently with specific role chain axioms such as transitivity $T{\cdot}T \sqsubseteq T$. It is closely related to a similar optimization for role chain axioms presented previously for a fragment of $\mathcal{EL}^+_\bot$ [13]. To illustrate the problem addressed, consider the following example.

Example 5. Consider the ontology $\mathcal{O} = \{A \sqsubseteq \exists L.B,\ B \sqsubseteq \exists P.C,\ C \sqsubseteq \exists P.D,\ L{\cdot}P \sqsubseteq L,\ P{\cdot}P \sqsubseteq P\}$. The roles L and P can be thought of as expressing the 'located-in' and 'part-of' relations. So the last axiom of $\mathcal{O}$, in particular, expresses that if x is located in y and y is a part of z then x is located in z. Let us try to determine whether $\mathcal{O} \models A \sqsubseteq B$ using goal-direct application of rules (skipping applications of $\mathbf{C_0}$ and $\mathbf{R_0}$, and applications of $\mathbf{C}_\sqsubseteq$, $\mathbf{R}_\sqsubseteq$ producing axioms in $\mathcal{O}$ as noted before):

$$A \sqsubseteq \exists(L{\cdot}P).C \quad \text{by } \mathbf{C}_\circ(A \sqsubseteq \exists L.B,\ L \sqsubseteq L,\ B \sqsubseteq \exists P.C,\ P \sqsubseteq P), \tag{39}$$

$$A \sqsubseteq \exists(L{\cdot}P).D \quad \text{by } \mathbf{C}_\circ(A \sqsubseteq \exists(L{\cdot}P).C,\ L{\cdot}P \sqsubseteq L,\ C \sqsubseteq \exists P.D,\ P \sqsubseteq P), \tag{40}$$

$$B \sqsubseteq \exists(P{\cdot}P).D \quad \text{by } \mathbf{C}_\circ(B \sqsubseteq \exists P.C,\ P \sqsubseteq P,\ C \sqsubseteq \exists P.D,\ P \sqsubseteq P), \tag{41}$$

$$A \sqsubseteq \exists(L{\cdot}P).D \quad \text{by } \mathbf{C}_\circ(A \sqsubseteq \exists L.B,\ L \sqsubseteq L,\ B \sqsubseteq \exists(P{\cdot}P).D,\ P{\cdot}P \sqsubseteq P). \tag{42}$$

Note that the axiom $A \sqsubseteq \exists(L{\cdot}P).D$ was derived two times by $\mathbf{C_0}$ in (40) and (42). Intuitively, this is because the role chain inclusion $L{\cdot}P{\cdot}P \sqsubseteq L{\cdot}P$ can be proved in two ways: either as $(L{\cdot}P){\cdot}P \sqsubseteq L{\cdot}P$ using $L{\cdot}P \sqsubseteq L$ or as $L{\cdot}(P{\cdot}P) \sqsubseteq L{\cdot}P$ using $P{\cdot}P \sqsubseteq P$.

In general, suppose that we have a derivation where some axiom is derived by $\mathbf{C}_\circ$:

$$C \sqsubseteq \exists(R{\cdot}\rho).F \quad \text{by } \mathbf{C}_\circ(C \sqsubseteq \exists S.D,\ S \sqsubseteq R,\ D \sqsubseteq \exists(T{\cdot}\mu).F,\ (T{\cdot}\mu) \sqsubseteq \rho). \tag{43}$$

Let us try to determine when the same conclusion can be derived differently. Suppose that $\mu \neq \epsilon$. Then $D \sqsubseteq \exists(T{\cdot}\mu).F$ can be only derived by $\mathbf{C}_\circ$:

$$D \sqsubseteq \exists(T{\cdot}\mu).F \quad \text{by } \mathbf{C}_\circ(D \sqsubseteq \exists\tau.E,\ \tau \sqsubseteq T,\ E \sqsubseteq \exists\eta.F,\ \eta \sqsubseteq \mu). \tag{44}$$

Now suppose that we also have $S \sqsubseteq S$, $S{\cdot}T \sqsubseteq R$ and $\eta \sqsubseteq \rho$ in the derivation. Then $C \sqsubseteq \exists(R{\cdot}\rho).F$ can be derived as follows:

$$C \sqsubseteq \exists(S{\cdot}T).E \quad \text{by } \mathbf{C}_\circ(C \sqsubseteq \exists S.D,\ S \sqsubseteq S,\ D \sqsubseteq \exists\tau.E,\ \tau \sqsubseteq T), \tag{45}$$

$$C \sqsubseteq \exists(R{\cdot}\rho).F \quad \text{by } \mathbf{C}_\circ(C \sqsubseteq \exists(S{\cdot}T).E,\ S{\cdot}T \sqsubseteq R,\ E \sqsubseteq \exists\eta.F,\ \eta \sqsubseteq \rho). \tag{46}$$

Note that when we replace the rule application (43) with the two rule applications (45) and (46), the third premises $D \sqsubseteq \exists\tau.E$ and $E \sqsubseteq \exists\eta.F$ used in these applications appear in the derivation before the third premise $D \sqsubseteq \exists(T{\cdot}\mu).F$ of (43) since the latter was obtained from the former by (44). Therefore, if we apply this transformation repeatedly to all inferences by $\mathbf{C}_\circ$, it will always terminate.

To summarize, we can always avoid applying $\mathbf{C}_\circ$ with $S \sqsubseteq R$ and $T{\cdot}\mu \sqsubseteq \rho$ as the second and the last premises if $\mu \neq \epsilon$ and we can derive $S{\cdot}T \sqsubseteq R$ ($S \sqsubseteq S$ is always derivable by $\mathbf{R_0}$) and $\tau \sqsubseteq \rho$ for every derivable $\tau \sqsubseteq \mu$. Due to the subformula

property, we can precompute all subsumptions on role chains occurring in the ontology and compute all pairs $\langle \rho_1 \sqsubseteq R, \rho_2 \sqsubseteq \rho \rangle$ of such role subsumptions with ρ_1, ρ_2 and $R{\cdot}\rho$ occurring in the ontology, excluding the pairs $\langle S \sqsubseteq R, T{\cdot}\mu \sqsubseteq \rho \rangle$ for which the above condition holds. Only the remaining pairs of subsumptions should then be used in $\mathbf{C}_\circ$.

Example 6. Continuing Example 5, we can show that the above conditions hold for the pair $\langle S \sqsubseteq R, T{\cdot}\mu \sqsubseteq \rho \rangle = \langle L \sqsubseteq L, P{\cdot}P \sqsubseteq P \rangle$. Indeed, $S{\cdot}T \sqsubseteq R = L{\cdot}P \sqsubseteq L$ can be derived, and since $\mu = \rho = P$, $\tau \sqsubseteq \rho$ is derivable if $\tau \sqsubseteq \mu$ is. Thus, the pair $\langle L \sqsubseteq L, P{\cdot}P \sqsubseteq P \rangle$ should not be used in $\mathbf{C}_\circ$, and thus inference (42) is not necessary. One can show that the pair $\langle P \sqsubseteq P, P{\cdot}P \sqsubseteq P \rangle$ should not be used in $\mathbf{C}_\circ$ as well.

As mentioned, there is a close relation of the optimization above with a similar optimization for rules in Figure 1 [13]. The main idea is to identify the role chain axioms $\rho \sqsubseteq R$ for which rule $\mathbf{E}_\circ$ can be applied in a *left-linear way*, that is, only if all premises starting from the second one are derived by rules other than $\mathbf{E}_\circ$ with $k \geq 2$. This is the case, e.g., for both axioms $L{\cdot}P \sqsubseteq L$ and $P{\cdot}P \sqsubseteq P$ from ontology $\mathcal{O}$ in Example 5. For example, rule $\mathbf{E}_\circ$ using $L{\cdot}P \sqsubseteq L \in \mathcal{O}$ would be applied for $A \sqsubseteq \exists L.B$ and $B \sqsubseteq \exists P.C$ (derived by $\mathbf{E}_{\sqsubseteq}$), but would not be applied for $A \sqsubseteq \exists L.B$ and $B \sqsubseteq \exists P.D$ if $B \sqsubseteq \exists P.D$ is derived by $\mathbf{E}_\circ$ from $B \sqsubseteq \exists P.C$ and $C \sqsubseteq \exists P.D$ using $P{\cdot}P \sqsubseteq P \in \mathcal{O}$. The reason is that the same conclusion $A \sqsubseteq \exists L.D$ can be derived in a left-linear way from $A \sqsubseteq \exists L.C$ (derived by $\mathbf{E}_\circ$ using $L{\cdot}P \sqsubseteq L \in \mathcal{O}$) and $C \sqsubseteq \exists P.D$ (derived by $\mathbf{E}_{\sqsubseteq}$). The main difference between the two optimizations, is that, due to the differences in the rules in Figure 1 and Figure 2, instead of classifying which *stated* role chain axioms can be used in a left-linear way, we determine which *derived* role chain axioms should be 'concatenated' in rule $\mathbf{C}_\circ$. The latter is algorithmically easier to determine using the condition formulated above.

4 Deterministic Saturation

Recall from Example 4 that the set of axioms obtained by applying the rules in a goal-directed way may depend on the order in which the rules are applied. Although this side effect has no impact on reasoning results, it introduces some difficulties when extending ELK reasoning services beyond checking of logical entailment. Specifically, the procedures for incremental reasoning [8] and proof generation [9] implemented in ELK, require repeating some rule applications performed in the derivation, and if the rules are applied in a different order than originally, the procedures may result in incorrect results. In this section we describe a modification of our rule application procedure for which the derived axioms do not depend on the order of rule application.

Recall from Section 2.3 that to determine whether a rule such as $\mathbf{R}_\exists$ should be applied to some axioms in the derivation, i.e., it is not redundant, one has to check which of these axioms are composed, i.e., *can be derived* by certain rules from the *previous* axioms in the derivation. This property ensures that only necessary rules are applied, but it can make rule application dependent on when (i.e, after which axioms) the premises were derived. Instead, we may slightly relax this requirement and decide whether an axiom is composed or not only based on the rules by which it was actually derived, which would make rule applications not to depend on other axioms in the derivation.

$$\mathbf{C}^{-}_{\equiv}\ \frac{C \sqsubseteq A}{C \sqsubseteq D} : A \equiv D \in \mathcal{O} \qquad\qquad \mathbf{C}^{+}_{\equiv}\ \frac{C \sqsubseteq D}{C \sqsubseteq A} : A \equiv D \in \mathcal{O}$$

Fig. 3. The new inference rules for concept definitions

For example, axiom $A \sqsubseteq \exists R.C$ in Example 4 was derived by both rules $\mathbf{C}_{\exists}$ and $\mathbf{C}_{\sqsubseteq}$. We would then not apply $\mathbf{C}_{\exists}$ to the first conclusion of a 'composition' inference by $\mathbf{C}_{\exists}$, but apply it to the second conclusion of 'non-composition' inference by $\mathbf{C}_{\sqsubseteq}$. Clearly, the advantage here is that it does not matter which of the two inferences was made first. It may seem that axioms are rarely derived by multiple rules, so the relaxed rule application strategy might not result in too many unnecessary inferences. The following example illustrates that this may not be really the case in practice.

Example 7. Consider the ontology $\mathcal{O} = \{A \sqsubseteq B \sqcap \exists R.A,\ C \equiv \exists R.B\}$. Recall, that concept equivalence $C \equiv \exists R.B$ represents two axioms $C \sqsubseteq \exists R.B$ and $\exists R.B \sqsubseteq C$. To test $\mathcal{O} \models A \sqsubseteq C$, we apply the rules in Figure 2 in a goal-directed way:

$$\begin{array}{lll}
 & A \sqsubseteq B & \text{by } \mathbf{C}^{-}_{\sqcap}(A \sqsubseteq B \sqcap \exists R.A), \quad (47)\\
 & A \sqsubseteq \exists R.A & \text{by } \mathbf{C}^{-}_{\sqcap}(A \sqsubseteq B \sqcap \exists R.A), \quad (48)\\
(+) & A \sqsubseteq \exists R.B & \text{by } \mathbf{C}_{\exists}(A \sqsubseteq \exists R.A,\ R \sqsubseteq R,\ A \sqsubseteq B), \quad (49)\\
 & A \sqsubseteq C & \text{by } \mathbf{C}_{\sqsubseteq}(A \sqsubseteq \exists R.B) : \exists R.B \sqsubseteq C \in \mathcal{O}, \quad (50)\\
 & A \sqsubseteq \exists R.B & \text{by } \mathbf{C}_{\sqsubseteq}(A \sqsubseteq C) : C \sqsubseteq \exists R.B. \quad (51)
\end{array}$$

Note that axiom $A \sqsubseteq \exists R.B$ is derived by rules $\mathbf{C}_{\exists}$ and $\mathbf{C}_{\sqsubseteq}$, so we would need to consider the second conclusion (51) for applications of $\mathbf{C}_{\exists}$. Note that the second rule application is a direct result of the equivalence axiom $C \equiv \exists R.B$, which was used to replace the subsumer $\exists R.B$ in (49) with C and back with $\exists R.B$. So it is actually not possible to have application (51) before (49).

The scenario illustrated in Example 7 is rather common: whenever an axiom $C \sqsubseteq D$ is derived and D occurs in some concept equivalence in the ontology, the same axiom $C \sqsubseteq D$ will be derived again. To avoid such duplicate inferences, we introduce new rules in Figure 3 to deal specifically with *concept definitions*—concept equivalences $A \equiv D$ where A is a (*defined*) atomic concept. Most concept equivalences in existing ontologies are of this form. We assume that all concept equivalences in $\mathcal{O}$ are concept definitions and each atomic concept A is defined in at most one of them; remaining concept equivalences can be always replaced with concept inclusions. Finally, we extend Definition 1 by allowing composed axioms to be obtained by rule $\mathbf{C}^{+}_{\equiv}$ and redundant rule applications to include applications by rule $\mathbf{C}^{-}_{\equiv}$ in which the premise is composed. Note that if the premise of $\mathbf{C}^{-}_{\equiv}$ is composed, then it can only be obtained by $\mathbf{C}^{+}_{\equiv}$ using the same concept definition $A \equiv D$, and hence the conclusion of this rule must be already derived.

Example 8. Continuing Example 7, with the new rules in Figure 3 we will have just rule application (52) instead of (50)–(51); the application of rule $\mathbf{C}^{-}_{\equiv}$ to (52) is redundant.

$$(+) \quad A \sqsubseteq C \qquad \text{by } \mathbf{C}^{+}_{\equiv}(A \sqsubseteq \exists R.B) : C \equiv \exists R.B \in \mathcal{O}. \quad (52)$$

Table 1. Summary information for the test ontologies (numbers of the different kinds of axioms)

Ontology	$C \sqsubseteq D$	$C \equiv D$	$R \sqsubseteq S$	$R \cdot R \sqsubseteq R$	$R_1 \cdot R_2 \sqsubseteq S$
EL-GALEN	25,563	9,968	958	58	0
GALEN7	27,820	15,270	1000	0	385
GALEN8	53,449	113,622	1024	0	385
SNOMED CT (Jan 2014)	229,330	69,908	11	0	1
ANATOMY	17,551	21,831	4	3	2

5 Preliminary Experimental Evaluation

In this section we present the preliminary results of empirical evaluation of the two new reasoning techniques described in Sections 3 and 4: the role composition optimization and the deterministic saturation using new rules for concept definitions. For both experiments we use a set of the well-known biomedical ontologies:[1] the July 2014 version of SNOMED CT,[2] three versions of OpenGALEN[3] (EL-GALEN, GALEN7, and GALEN8), and ANATOMY (an experimental version of SNOMED CT which uses role chain axioms to model the body structure). These ontologies have been frequently used in the past for evaluation of $\mathcal{EL}$ reasoners [14, 15, 1]. The summary information about these ontologies is presented in Table 1.

For experiments we used a development version of ELK 0.5. The experimental setup is the same for all experiments: each ontology is classified 20 times, 10 warm-up runs to exclude the effects of JIT compilation and 10 measured runs, for which the results are averaged. The combined loading + classification wall clock time (in ms.) is used as the main performance metric. We used a PC with Intel Core i5-2520M 2.50GHz CPU, Java 1.6 and 4GB RAM available to JVM.

The first experiment evaluates effectiveness of the role composition optimization described in Section 3. All ontologies (except for SNOMED CT in which the only sub-role chain axiom has no effect) are classified with the optimization being turned on and off. The results are shown in Table 2. It can be seen that in most cases the optimization considerably reduces the number of inferences as well the number of derived subsumptions (many subsumptions are derived by several inferences). The difference translates into time savings. The ANATOMY ontology stands out as the case where the optimization is critically important since it cuts down the number of inferences by rule $\mathsf{C}_\circ$ by nearly an order of magnitude.

The aim of the the second experiment is to evaluate the effectiveness of the deterministic saturation optimization described in Section 4. We compare the classification time and the number of derived axioms in three cases: a) with the ELK' current non-deterministic saturation [1], b) with deterministic saturation using the rules in Figure 2, and c) with deterministic saturation using the additional rules for concept definitions (see Figure 3). The results are shown in Table 3.

[1] Unless specified otherwise, the ontologies can be downloaded from the ELK project page `elk.semanticweb.org`

[2] `http://www.ihtsdo.org/licensing/`

[3] `http://www.opengalen.org/sources/sources.html`

Table 2. Evaluation of the role composition optimization from Section 3

Ontology	Classification time		Inferences		Derived subsumptions	
Optimization	On	Off	On	Off	On	Off
EL-GALEN	964	1,039	2,080,194	2,207,823	1,485,247	1,550,695
GALEN7	1,632	1,998	5,707,082	7,001,796	2,787,261	2,952,952
GALEN8	15,587	16,981	40,239,327	49,719,156	19,377,172	20,136,178
ANATOMY	8,957	27,766	45,466,892	176,130,924	7,105,923	10,223,484

Table 3. Evaluation of deterministic saturation (see Section 3). The shortcuts "non-det", "det", and "det+defn" stand for non-deterministic saturation, unoptimized deterministic saturation, deterministic saturation with the new optimized rules for handling of concept definitions.

Ontology	Classification time			Inferences			Derived subsumptions		
	non-det	det	det+defn	non-det	det	det+defn	non-det	det	det+defn
EL-GALEN	959	1,513	1,032	2.1M	4.2M	2.3M	1.5M	2.7M	1.9M
GALEN7	1,984	3,408	2,699	5.7M	11.6M	9.9M	2.8M	5.3M	4.7M
GALEN8	16,750	27,339	19,866	40.2M	93.7M	65.3M	19.4M	37.2M	28.4M
SNOMED CT	14,441	21,791	15,431	25.5M	54.1M	31.2M	17.1M	30.3M	23.6M
ANATOMY	9,183	15,546	9,875	45.5M	73.0M	47.2M	7.1M	12.1M	8.3M

One can see that deterministic saturation without further optimizations is significantly slower and often makes nearly twice as many inferences as non-deterministic saturation. This is largely because many axioms are derived by multiple inferences due to equivalence axioms (as illustrated in Example 3). The special rules to deal with concept definitions reduce such redundant derivations and improve performance so that it is close to that of non-deterministic saturation. Still in some cases, e.g., for GALEN7 and GALEN8, the difference between non-deterministic and optimized deterministic strategies is visible and it remains our goal to investigate how it can be reduced even further.

6 Summary

The paper presented several recent developments in ELK which range from novel reasoning optimizations, such as efficient handling of role chain axioms, to modifications aimed at supporting additional reasoning services, such as proof-based explanations. Our experiments show that the latter changes *may* result in minor performance setbacks, and it remains our future goal to investigate how to avoid even such minor performance compromises.

References

1. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. of Automated Reasoning **53**(1) (2014) 1–61
2. Harris, M.A., Lock, A., Bühler, J., Oliver, S.G., Wood, V.: FYPO: the fission yeast phenotype ontology. Bioinformatics **29**(13) (2013) 1671–1678

3. Hoehndorf, R., Dumontier, M., Gkoutos, G.V.: Identifying aberrant pathways through integrated analysis of knowledge in pharmacogenomics. Bioinformatics **28**(16) (2012) 2169–2175
4. Hoehndorf, R., Harris, M.A., Herre, H., Rustici, G., Gkoutos, G.V.: Semantic integration of physiology phenotypes with an application to the cellular phenotype ontology. Bioinformatics **28**(13) (2012) 1783–1789
5. Jupp, S., Stevens, R., Hoehndorf, R.: Logical gene ontology annotations (GOAL): exploring gene ontology annotations with OWL. J. of Biomedical Semantics **3(Suppl 1)**(S3) (2012) 1–16
6. Osumi-Sutherland, D., Reeve, S., Mungall, C.J., Neuhaus, F., Ruttenberg, A., Jefferis, G.S.X.E., Armstrong, J.D.: A strategy for building neuroanatomy ontologies. Bioinformatics **28**(9) (2012) 1262–1269
7. The Gene Ontology Consortium: Gene ontology annotations and resources. Nucleic Acids Res (2012)
8. Kazakov, Y., Klinov, P.: Incremental reasoning in OWL EL without bookkeeping. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. (2013) 232–247
9. Kazakov, Y., Klinov, P.: Goal-directed tracing of inferences in $\mathcal{EL}$ ontologies. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II. (2014) 196–211
10. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P., eds.: Proc. 6th Int. Semantic Web Conf. (ISWC'07). Volume 4825 of LNCS., Springer (2007) 267–280
11. Kazakov, Y., Klinov, P.: Advancing ELK: Not only perormance matters. Technical report, University of Ulm (2015) available from `http://http://elk.semanticweb.org/publications/elk-advancing-trdl-2015.pdf`.
12. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L., Saffiotti, A., eds.: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), Professional Book Center (2005) 364–369
13. Kazakov, Y., Krötzsch, M., Simančík, F.: Unchain my $\mathcal{EL}$ reasoner. In Rosati, R., Rudolph, S., Zakharyaschev, M., eds.: Proc. 24th Int. Workshop on Description Logics (DL'11). Volume 745 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 202–212
14. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient reasoning in $\mathcal{EL}^+$. In Parsia, B., Sattler, U., Toman, D., eds.: Proc. 19th Int. Workshop on Description Logics (DL'06). Volume 189 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
15. Lawley, M.J., Bousquet, C.: Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In: Proc. 6th Australasian Ontology Workshop (IAOA'10). (2010) 45–49

Nonmonotonic Nominal Schemas Revisited

Matthias Knorr

NOVA LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa

Abstract. Recently, a very general description logic (DL) that extends $\mathcal{SROIQ}$ (the DL underlying OWL 2 DL) at the same time with nominal schemas and epistemic modal operators has been proposed, which encompasses some of the most prominent monotonic and non-monotonic rule languages, including Datalog under the answer set semantics. A decidable fragment is also presented, but the restricted language does not fully cover all formalisms encompassed by the complete language. In this paper, we aim to remedy that by studying an alternative set of restrictions to achieve decidability, and we show that the existing embeddings of the formalisms covered by the full language can be adjusted accordingly.

1 Introduction

Extending Description Logics (DLs) with modeling features admitting non-monotonic reasoning has been frequently requested in many application domains (see, e.g., [14] for semantic matchmaking on annotations at electronic online marketplaces). In fact, the vast amount of work dedicated to the topic may serve as a witness in its own right. DLs have been extended, for example, with defaults [2], with notions of circumscription [4,33], and epistemic reasoning provided by the inclusion of modal[1] operators within the language [8] or only in queries [29]. In addition, a plethora of approaches combine DLs with (often non-monotonic) rules (see, e.g., [9,30,19] and references in their sections on related work). As these approaches are commonly of different expressivity and based on quite advanced different formal grounds, a uniform overarching formalism allowing the integration of possibly all the various modeling features is an extremely complicated problem.

In [20], a very general DL language is introduced that extends the expressive DL underlying OWL 2, $\mathcal{SROIQ}$, with nominal schemas [24] and epistemic operators as defined in [8] with the aim of integrating the W3C standards OWL [15] and (non-monotonic) RIF [18] and their underlying formalisms, DLs and rule languages respectively, thus contributing towards the goal of a unifying logic for the Semantic Web (as foreseen in the well-known Semantic Web stack). The full language is in fact very expressive, capturing a variety of different formalisms, among them two based on MKNF logics [27] that had been considered of different expressivity so far – MKNF DLs [8], i.e., the epistemic extension of DLs, and Hybrid MKNF, one very expressive combination of DLs and non-monotonic rules. Though not the full language of the latter is

[1] In the remainder of the paper, we use the terms modal and epistemic operator interchangeably to refer to the same notion.

considered, coverage of Answer Set Programming [11] is ensured, arguably the most widely used non-monotonic reasoning rule formalism.

A decidable fragment of the full language is also considered in [20], which is strongly related to the one presented in [8]. In fact, the restrictions are such that the tableau algorithm presented in [8] can in principle be re-used. As it turns out, however, the decidable language does not encompass all the formalisms for which coverage within the full language is shown. While this does not invalidate the approach as such, in particular, if one views such a unifying formalism mainly as a conceptual underpinning, it is certainly undesired if one rather wants to use it for modeling and reasoning.

In this paper, we aim to solve this problem, i.e., we consider a different set of restrictions, for which we show that reasoning is decidable and that, at the same time, encompasses all the formalisms discussed in [20] with only minor adjustments to the previously presented embeddings. The principal idea builds on the usage of nominals and nominal schemas, which are necessarily present in the language by design anyway, to limit the applicability of concept inclusions containing modal operators. As an additional result, we believe that the new restrictions are more succinct and that the resulting adaptation of the procedure for verifying the existence of models becomes less complicated. To further simplify notation, here we do not consider the full language presented in [20], which is based on $\mathcal{SROIQ}$, but rather a language based on $\mathcal{ALC}$ with only the minimally necessary extensions and we term this language $e\mathcal{ALCOV}$ (see Sect. 2 for a detailed explanation on the name). As we can show, such language is already expressive enough to cover the desired non-monotonic modeling features.

The remainder of the paper proceeds as follows. In Sect. 2, we recall the syntax and semantics of the DL $e\mathcal{ALCOV}$ we consider here. We then introduce the new alternative conditions of so-called safe $e\mathcal{ALCOV}$ KBs in Sect. 3 and we subsequently show that these do ensure decidability of reasoning, i.e., checking (MKNF-)satisfiability. In Sect. 4, we show that, with minor adaptations, the applied changes do now permit coverage of the discussed formalisms in [20] within the decidable fragment (of safe $e\mathcal{ALCOV}$ KBs), before we conclude and discuss future work in Sect. 5.

2 Epistemic DLs

In this section, we recall the syntax and semantics of epistemic description logics (DLs) from [20]. Here, we focus on a subset of the language considered in [20] to make the presentation more concise and to ease the reading. Namely, we consider the epistemic DL $\mathcal{ALCK}_{\mathcal{NF}}$ [8], which is $\mathcal{ALC}$ enhanced with epistemic operators, extended by nominals, nominal schemas [24], and the universal role. Nominal schemas represent variable nominals that can only be bound to known individuals, and the universal role can be represented using role hierarchies and negation on roles [23], but as we want to keep the presentation simple, we leave this implicit. Since the name of the resulting language $\mathcal{ALCOVK}_{\mathcal{NF}}$ (or even $\mathcal{ALCHOV}(\neg)\mathcal{K}_{\mathcal{NF}}$ for the implicit encoding of the universal role) following standard and historic patterns would be quite cumbersome, we propose using the name $e\mathcal{ALCOV}$ instead, which stands for epistemic $\mathcal{ALCOV}$ (including the universal role). The term epistemic originates from the two epistemic/modal operators $\mathbf{K}$ and $\mathbf{A}$, where $\mathbf{K}$ is interpreted in terms of minimal knowledge, while $\mathbf{A}$ is interpreted

Table 1. Syntax and semantics of $e\mathcal{ALCOV}$

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta$
role name	V	$V^{\mathcal{I}} \subseteq \Delta \times \Delta$
individual name	a	$a^{\mathcal{I}} \in \Delta$
variable	x	$\mathcal{Z}(x) \in \Delta$
top	$\top$	Δ
bottom	$\bot$	$\emptyset$
nominal (schema)	$\{t\}$	$\{a \mid [a]_{\approx} \approx t^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}\}$
concept complement	$\neg C$	$\Delta \setminus C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
concept conjunction	$C \sqcap D$	$C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \cap D^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
concept disjunction	$C \sqcup D$	$C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \cup D^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
existential restriction	$\exists R.C$	$\{\delta \in \Delta \mid \exists \epsilon \text{ with } (\delta,\epsilon) \in R^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \text{ and } \epsilon \in C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}\}$
universal restriction	$\forall R.C$	$\{\delta \in \Delta \mid (\delta,\epsilon) \in R^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \text{ implies } \epsilon \in C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}\}$
knowledge concept	$\mathbf{K}C$	$\bigcap_{\mathcal{J}\in\mathcal{M}} C^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
assumption concept	$\mathbf{A}C$	$\bigcap_{\mathcal{J}\in\mathcal{N}} C^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
universal role	U	$\Delta \times \Delta$
knowledge role	$\mathbf{K}V$	$\bigcap_{\mathcal{J}\in\mathcal{M}} V^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
assumption role	$\mathbf{A}V$	$\bigcap_{\mathcal{J}\in\mathcal{N}} V^{(\mathcal{J},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
role assertion	$V(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in V^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$
TBox axiom	$C \sqsubseteq D$	$C^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}} \subseteq D^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$

Interpretation $\mathcal{I}$; MKNF structure $(\mathcal{I},\mathcal{M},\mathcal{N})$; variable assignment $\mathcal{Z}$; $A \in N_C$; $C, D \in \mathsf{C}$; $V \in N_R$; $R \in \mathsf{R}$; $a, b \in N_I$; $x \in N_V$, and $t \in N_V \cup N_I$.

as autoepistemic assumption and corresponds to $\neg\mathbf{not}$, i.e., the classical negation of the negation as failure operator $\mathbf{not}$ used in [27] instead of $\mathbf{A}$.

We consider a signature $\Sigma = \langle N_I, N_C, N_R, N_V \rangle$ where N_I, N_C, N_R, and N_V are pairwise disjoint and finite sets of *individual names*, *concept names*, *role names*, and *variables*. In the following, we assume that Σ has been fixed. We define concepts and roles in $e\mathcal{ALCOV}$ by the following grammar.

$$\mathsf{R} ::= V \mid U \mid \mathbf{K}V \mid \mathbf{A}V$$
$$\mathsf{C} ::= \top \mid \bot \mid A \mid \{i\} \mid \{x\} \mid \neg\mathsf{C} \mid \mathsf{C} \sqcap \mathsf{C} \mid \mathsf{C} \sqcup C \mid \exists\mathsf{R}.\mathsf{C} \mid \forall\mathsf{R}.\mathsf{C} \mid \mathbf{K}\mathsf{C} \mid \mathbf{A}\mathsf{C}$$

where $V \in N_R$, $A \in N_C$, $i \in N_I$, and $x \in N_V$. The names of the individual concepts and roles can be found in Table 1. *Epistemic concepts* are knowledge and assumption concepts, while *epistemic roles* are knowledge and assumption roles.

As usual, a *TBox axiom* (or *general concept inclusion* (GCI)) is an expression $C \sqsubseteq D$ where $C, D \in \mathsf{C}$. An *ABox axiom* is of the form $C(a)$ or $V(a,b)$ where $C \in \mathsf{C}$,

$V \in N_R$, and $a, b \in N_I$. An $e\mathcal{ALCOV}$ *axiom* is any ABox or TBox axiom, and an $e\mathcal{ALCOV}$ *knowledge base* (KB) is a finite set of $e\mathcal{ALCOV}$ axioms.

The semantics of $e\mathcal{ALCOV}$ as recalled from [20] is an adaptation from [24] and [8]. As common, an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a domain $\Delta^\mathcal{I} \neq \emptyset$ and a function $\cdot^\mathcal{I}$ that maps elements in N_I, N_C, and N_R to elements, sets, and relations of $\Delta^\mathcal{I}$ respectively. Additionally, nominal schemas require a *variable assignment* $\mathcal{Z}$ for an interpretation $\mathcal{I}$, which is a function $\mathcal{Z} : N_V \to \Delta^\mathcal{I}$ such that, for each $v \in N_V$, $\mathcal{Z}(v) = a^\mathcal{I}$ for some $a \in N_I$.

As common in MKNF-related semantics used to combine DLs with non-monotonic reasoning (see [8,17,19,30]), specific restrictions on interpretations are introduced to ensure that certain unintended logical consequences can be avoided (see, e.g., [30]). Here, we adopt the standard name assumption from [20]. An interpretation $\mathcal{I}$ (over Σ to which $\approx$ is added) employs the *standard name assumption* if

(1) N_I^* extends N_I with a countably infinite set of individuals that cannot be used in variable assignments, and $\Delta^\mathcal{I} = N_I^*$;
(2) for each i in N_I^*, $i^\mathcal{I} = i$; and
(3) equality $\approx$ is interpreted in $\mathcal{I}$ as a congruence relation – that is, $\approx$ is reflexive, symmetric, transitive, and allows for the replacement of equals by equals [10].

The first condition fixes the (infinite) universe, but limits the application of variable assignments to a finite subset, the second condition defines $\mathcal{I}$ as a bijective function, while the third ensures that we still can identify elements of the domain. As an immediate side-effect, the variable assignment is no longer tied to a specific interpretation and we can simplify notation by using Δ without reference to a concrete interpretation.

Now, the first-order semantics is lifted to satisfaction in MKNF structures that treat the modal operators w.r.t. sets of interpretations. An *MKNF structure* is a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ where $\mathcal{I}$ is an interpretation, $\mathcal{M}$ and $\mathcal{N}$ are sets of interpretations, and $\mathcal{I}$ and all interpretations in $\mathcal{M}$ and $\mathcal{N}$ are defined over Δ. For any such $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ and assignment $\mathcal{Z}$, the function $\cdot^{(\mathcal{I},\mathcal{M},\mathcal{N}),\mathcal{Z}}$ is defined for arbitrary $e\mathcal{ALCOV}$ expressions as shown in Table 1. $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ and $\mathcal{Z}$ *satisfy* an $e\mathcal{ALCOV}$ axiom α, written $(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z} \models \alpha$, if the corresponding condition in Table 1 holds. $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ *satisfies* α, written $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z} \models \alpha$ for all variable assignments $\mathcal{Z}$. A (non-empty) set of interpretations $\mathcal{M}$ *satisfies* α, written $\mathcal{M} \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \alpha$ holds for all $\mathcal{I} \in \mathcal{M}$, and $\mathcal{M}$ *satisfies* an $e\mathcal{ALCOV}$ knowledge base KB, written $\mathcal{M} \models KB$, if $\mathcal{M} \models \alpha$ for all axioms $\alpha \in KB$. Note the small deviation of the semantics of $\{t\}$ in Table 1 compared to that in [24], which is necessary to ensure that the semantics works as intended under standard name assumption.

It can be verified that the two sets of interpretations are each used to interpret one of the modal operators, but in the monotonic semantics above, they simply coincide. This changes with the non-monotonic MKNF model defined in the usual fashion [8,17,19,30]: $\mathcal{M}$ is fixed to interpret $\mathbf{A}$, and supersets $\mathcal{M}'$ of $\mathcal{M}$ are used to test whether the knowledge derived from $\mathcal{M}$ (via $\mathbf{K}$) is indeed minimal.

Definition 1. *Given an $e\mathcal{ALCOV}$ knowledge base KB, a (non-empty) set of interpretations $\mathcal{M}$ is an* MKNF model *of KB if (1) $\mathcal{M} \models KB$, and (2) for each $\mathcal{M}'$ with $\mathcal{M} \subset \mathcal{M}'$, $(\mathcal{I}', \mathcal{M}', \mathcal{M}) \not\models KB$ for some $\mathcal{I}' \in \mathcal{M}'$. KB is* MKNF-satisfiable *if an*

MKNF model of KB exists. An axiom α is MKNF-entailed by KB, *written $KB \models_{\mathbf{K}} \alpha$, if all MKNF models $\mathcal{M}$ of KB satisfy α.*

As noted in [8], since $\mathcal{M} \models KB$ is defined w.r.t. $(\mathcal{I}, \mathcal{M}, \mathcal{M})$, the operators $\mathbf{K}$ and $\mathbf{A}$ are interpreted in the same way, and so we can restrict instance checking $KB \models_{\mathbf{K}} C(a)$ and subsumption $KB \models_{\mathbf{K}} C \sqsubseteq D$ to C and D without occurrences of the operator $\mathbf{A}$. Also, in absence of modal operators in the $e\mathcal{ALCOV}$ KB, there is a unique MKNF model which simply contains all standard (first-order) models of KB as usual.

3 Reasoning in $e\mathcal{ALCOV}$

In [20], reasoning in $e\mathcal{SROIQ}$[2] is discussed following [8] for reasoning in $\mathcal{ALCK}_{\mathcal{NF}}$. The problem with this approach is that it is undecidable in general, so, as in [8], restrictions are applied in [20] to regain decidability, which in certain cases prevent coverage of the formalisms encompassed by the unrestricted language. To circumvent this, we still rely on the same idea in principle, but we revise the applied restrictions to achieve decidability making use of the gained expressiveness in $e\mathcal{ALCOV}$. In the following, we spell out the restrictions with some motivation right away, before we show that this indeed yields a decidable procedure for checking MKNF-satisfiability.

3.1 Safe $e\mathcal{ALCOV}$ KBs

Following [8], the overall idea is to reduce reasoning in $e\mathcal{ALCOV}$ to a number of reasoning tasks in non-modal $\mathcal{ALCOV}$ (again including U), for which each model of an $e\mathcal{ALCOV}$ KB is represented by means of an $\mathcal{ALCOV}$ KB. Formally, a set of interpretations $\mathcal{M}$ is $\mathcal{ALCOV}$ *representable* if there exists an $\mathcal{ALCOV}$ KB $KB_{\mathcal{M}}$ such that $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \text{ satisfies } KB_{\mathcal{M}}\}$. Then, undecidability can be caused by three sources. First, certain partially quantified expressions are not $\mathcal{ALCOV}$ representable (Theorem 4.1 in [8]), which is why we recall the notion of subjectively quantified KBs. For that purpose, we define that an $\mathcal{ALCOV}$ expression S is *subjective* if each $\mathcal{ALCOV}$ subexpression in S lies in the scope of at least one modal operator.

Definition 2. *An $e\mathcal{ALCOV}$ KB KB is* subjectively quantified *if each expression of the form $\exists R.C$, $\forall R.C$ occurring in KB satisfies one of the conditions: (1) R is an $\mathcal{ALCOV}$ role and C is an $\mathcal{ALCOV}$ concept, or (2) R and C are both subjective and C is of the form $\mathbf{K}D$, $\neg\mathbf{K}D$, $\mathbf{A}D$ or $\neg\mathbf{A}D$.*

There exists a slightly relaxed condition on subjectively quantified KBs [17], but for our purposes the original one suffices.

Second, even if subjectively quantified, certain nested expressions can be problematic, so we introduce (modally) flat concepts, that can be seen as a further restriction of simple concepts in [8], which prohibit such nesting altogether. Formally, an $e\mathcal{ALCOV}$ concept is *flat* if it does not contain any modal operator in scope of another, and an

[2] In [20], the term $\mathcal{SROIQV}(\mathcal{B}^s, \times)\mathcal{K}_{\mathcal{NF}}$ is used, but, for the sake of readability and with a slight abuse of notation, we follow our introduced naming scheme here.

$e\mathcal{ALCOV}$ KB KB is *flat* if each concept in it is flat. Thus, quantifier expressions of the form (2) in Def. 2 are flat as long as D does not contain further modal operators.

Third, intuitively, we have to make sure that GCIs involving modal operators cannot be used to derive an infinite number of true assertions (see also Theorem 4.10 in [8]). Rather than introducing simple KBs as in [8,20], we build on nominals and nominal schemas to introduce safe concepts.

Definition 3. *Given a subjectively quantified, flat $e\mathcal{ALCOV}$ KB KB, an $e\mathcal{ALCOV}$ concept C in KB is called* safe *if C is of the form $D \sqcap \{t\}$ for some* guard $t \in N_I \cup N_V$.

The idea is to use the nominal (schema) as a guard that restricts "applicability" of concepts involving modal operators to individuals occurring in KB. This all combines in the definition of safe $e\mathcal{ALCOV}$ KBs, for which we from now on consider two disjoint subsets of the $e\mathcal{ALCOV}$ TBox $\mathcal{T}$ of KB: $\mathcal{T}'$, the set of all axioms that contain no modal operators, and Γ, the set of all axioms that contain at least one modal operator.

Definition 4. *Let KB be an $e\mathcal{ALCOV}$ KB that is subjectively quantified and flat. Then, KB is* safe *if the following conditions are satisfied:*

1. *For each $C \sqsubseteq D \in \Gamma$, C is subjective and safe, D is safe for the same guard as C, and no operator $\mathbf{K}$ occurs in $\exists$ and $\forall$ restrictions in D;*
2. *There is no concept assertion in KB containing a subconcept of the form $\exists \mathbf{K}R.\mathbf{K}C$.*

Notably, due to KB also being subjectively quantified and flat, any $C \sqsubseteq D \in \Gamma$ in a safe KB can be rewritten into one such that all subjective subconcepts in it are safe. For example, the safe KB containing just the axiom

$$((\mathbf{K}(C \sqcup \exists R.D) \sqcap \exists \mathbf{K}R.\mathbf{K}G) \sqcup \neg \mathbf{A}E) \sqcap \{x\} \sqsubseteq \forall \mathbf{A}S.\mathbf{A}F \sqcap \{x\}$$

can be straightforwardly rewritten into

$$((\mathbf{K}(C \sqcup \exists R.D) \sqcap \{x\}) \sqcap (\exists \mathbf{K}R.\mathbf{K}G \sqcap \{x\})) \sqcup (\neg \mathbf{A}E \sqcap \{x\}) \sqsubseteq \forall \mathbf{A}S.\mathbf{A}F \sqcap \{x\}.$$

In the following, we assume that any $C \sqsubseteq D \in \Gamma$ in a safe $e\mathcal{ALCOV}$ KB is already rewritten this way, i.e., all subjective subconcepts in Γ are assumed safe.

Comparing to simple KBs (Def. 8 in [20]), intuitively, KB being flat covers condition 3. while 1. of Def. 4 covers to 1. and 2. there. Condition 2. in Def. 4 is not strictly required for decidability (as the case can be handled by finitely many models up to renaming of individuals [8]), but it will simplify the subsequent material without affecting coverage of related formalisms. These new conditions for safe KBs certainly have quite a different flavor compared to simple KBs in [8,20], but we believe that they are overall simpler and more easy to grasp, and at the same time not jeopardizing coverage of related formalisms. Of course, we still need to show that there is a decidable procedure for reasoning with safe $e\mathcal{ALCOV}$ KBs.

3.2 Determining MKNF Models

We start by grounding a given safe $e\mathcal{ALCOV}$ KB, i.e., we replace all occurring nominal schemas with nominals in all possible ways in the usual manner. This yields an $e\mathcal{ALCO}$ KB (again including U), which is trivially safe and obtained in finite time, though, in general, of exponential size in terms of the input KB.

From now on, we follow the principal argument from [8] as used in [20], but with some variations and simplifications due to our different restrictions and to some extent inspired by the reasoning algorithms for the related Hybrid MKNF [30].

First, we will collect a set of modal atoms based on the occurrence of epistemic concepts and roles in a given KB. In difference to [8], we only consider atoms over individuals occurring in the given KB. In the following, we use $\mathbf{M}$ to denote either $\mathbf{K}$ or $\mathbf{A}$, and $\mathbf{N}$ to denote either $\mathbf{M}$ or $\neg\mathbf{M}$, we assume $Q \in \{\exists, \forall\}$, and we remind that we consider that all subjective concepts in Γ are safe. Given a safe $e\mathcal{ALCO}$ KB KB, the set of *modal atoms* $MA(KB)$ is defined inductively as follows:

(1) if $\mathbf{M}D \sqcap \{a\}$ for some $a \in N_I$ occurs in KB, then $\mathbf{K}D(a) \in MA(KB)$;
(2) if $\mathbf{M}D$ occurs (non-safe) in concept assertion $C(a)$, then $\mathbf{K}D(a) \in MA(KB)$;
(3) if $Q\mathbf{M}R.\mathbf{N}D \sqcap \{a\}$ for some $a \in N_I$ occurs in KB, then $\mathbf{K}R(a,i), \mathbf{K}D(i) \in MA(KB)$ for all $i \in N_I$;
(4) if $Q\mathbf{M}R.\mathbf{N}D$ occurs (non-safe) in concept assertion $C(a)$, then $\mathbf{K}R(a,i), \mathbf{K}D(i) \in MA(KB)$ for all $i \in N_I$;
(5) nothing else belongs to $MA(KB)$.

A further difference to [8] is that we only collect modal atoms under $\mathbf{K}$. This is justified by the fact that for ensuring condition (1) of Def. 1, the same set of interpretations $\mathcal{M}$ is considered for evaluating formulas under $\mathbf{K}$ and $\mathbf{A}$. As an immediate benefit, when introducing partitions of these modal atoms next and guessing model candidates, we do not have to verify whether modal atoms under $\mathbf{K}$ and $\mathbf{A}$ are aligned.

We now introduce a *partition* of $MA(KB)$, which is a pair (P, N) of *positive modal atoms* P and *negative modal atoms* N such that $P \cap N = \emptyset$ and $P \cup N = MA(KB)$. As already mentioned, such partition can be understood as a guess about which modal atoms are supposed to be true (P) and false (N), and we can use it to simplify an $e\mathcal{ALCO}$ KB as follows. Given a safe $e\mathcal{ALCO}$ KB KB and a partition (P, N) of $MA(KB)$, $KB[P]$ denotes the $e\mathcal{ALCO}$ KB obtained from KB and (P, N) by:

1. replacing each occurrence of the form $\mathbf{M}D \sqcap \{a\}$ in KB and each (non-safe) occurrence of the form $\mathbf{M}D$ in a concept assertion $C(a) \in KB$ with $\top$ if $\mathbf{K}D(a) \in P$ and with $\bot$ otherwise;
2. replacing each occurrence of $\exists\mathbf{M}R.\mathbf{M}_1D \sqcap \{a\}$ ($\exists\mathbf{M}R.\neg\mathbf{M}_1D \sqcap \{a\}$) in KB and each (non-safe) occurrence of the form $\exists\mathbf{M}R.\mathbf{M}_1D$ ($\exists\mathbf{M}R.\neg\mathbf{M}_1D$) in a concept assertion $C(a) \in KB$ with $\top$ if there exists y such that $\mathbf{K}R(a,y) \in P$ and $\mathbf{K}D(y) \in P$ ($\mathbf{K}D(y) \notin P$) and with $\bot$ otherwise;
3. replacing each occurrence of $\forall\mathbf{M}R.\mathbf{M}_1D \sqcap \{a\}$ ($\forall\mathbf{M}R.\neg\mathbf{M}_1D \sqcap \{a\}$) in KB and each (non-safe) occurrence of the form $\forall\mathbf{M}R.\mathbf{M}_1D$ ($\forall\mathbf{M}R.\neg\mathbf{M}_1D$) in a concept assertion $C(a) \in KB$ with $\top$ if for each y such that $\mathbf{K}R(a,y) \in P$, $\mathbf{K}D(y) \in P$ ($\mathbf{K}D(y) \notin P$) and with $\bot$ otherwise.

Note that we leave N implicit here, as it is completely specified by P and the definition of a partition of $MA(KB)$. We generalize the notion of $KB[P]$, based on two partitions $(P, N), (P', N')$ of $MA(KB)$, to $KB[P'][P]$ which is obtained from KB in exactly the same way, only that if $\mathbf{M}$ or $\mathbf{M}_1$ is $\mathbf{K}$, then P' is used in the evaluation of the conditions, while for $\mathbf{M}$ or $\mathbf{M}_1$ being $\mathbf{A}$, P is used. In either case, it can readily be verified that the resulting KB does not contain any modal operators, hence is an $\mathcal{ALCO}$ KB (admitting U) for which satisfiability can be checked using standard DL reasoners.

With this in place, we can define an $\mathcal{ALCO}$ KB which takes the modal atoms guessed to be true into account, and use the resulting KB to check whether a guess is consistent with the original $e\mathcal{ALCO}$ KB. Let KB be a safe $e\mathcal{ALCO}$ KB and (P, N), (P', N') partitions of $MA(KB)$. Then, $Ob_{KB,P',P}$ denotes the following $\mathcal{ALCO}$ KB:

$$Ob_{KB,P',P} = KB[P'][P] \cup \{C(x) \mid \mathbf{K}C(x) \in P'\} \cup \{R(x,y) \mid \mathbf{K}R(x,y) \in P'\}$$

Then, partition (P, N) of $MA(KB)$ is *consistent with* the (safe) $e\mathcal{ALCO}$ KB if the following conditions hold:
(1) the $\mathcal{ALCO}$ KB $Ob_{KB,P,P}$ is satisfiable;
(2) $Ob_{KB,P,P} \not\models C(x)$ for each $\mathbf{K}C(x) \in N$;
(3) $Ob_{KB,P,P} \not\models R(x,y)$ for each $\mathbf{K}R(x,y) \in N$.
Basically, item (1) checks whether the guessed P does not yield contradictions w.r.t. KB, while (2) and (3) verify that no modal atom occurs wrongfully in N.

A link between a set of interpretations and partitions is established next. Let $\mathcal{M}$ be a set of interpretations over Δ. Then, $\mathcal{M}$ *induces* the partition (P, N) of $MA(KB)$:

$$\begin{aligned} P = {} & \{\mathbf{K}C(x) \mid \mathbf{K}C(x) \in MA(KB) \text{ and } \mathcal{M} \models \mathbf{K}C(x)\} \\ & \cup \{\mathbf{K}R(x,y) \mid \mathbf{K}R(x,y) \in MA(KB) \text{ and } \mathcal{M} \models \mathbf{K}R(x,y)\} \\ N = {} & \{\mathbf{K}C(x) \mid \mathbf{K}C(x) \in MA(KB) \text{ and } \mathcal{M} \not\models \mathbf{K}C(x)\} \\ & \cup \{\mathbf{K}R(x,y) \mid \mathbf{K}R(x,y) \in MA(KB) \text{ and } \mathcal{M} \not\models \mathbf{K}R(x,y)\} \end{aligned}$$

We can show that the intended correspondence indeed holds.

Lemma 1. *Let KB be a safe $e\mathcal{ALCO}$ KB, $\mathcal{M}$ a set of interpretations over Δ that satisfies KB such that $\mathcal{M} \models \mathbf{K}R(i_1, i_2)$ only if $i_1 \approx a \in N_I$ and $i_2 \approx b \in N_I$, and (P, N) the partition of $MA(KB)$ induced by $\mathcal{M}$. Then (P, N) is consistent with KB.*

Note that here, the particular restriction on $\mathcal{M}$ is necessary, otherwise the property would not hold. Take $(\exists \mathbf{A}R.\mathbf{A}C)(a)$. Then, $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models R(a,i) \wedge C(i)$ for some $i \in \Delta$ and $i \not\approx a\}$ clearly satisfies the assertion, yet the induced partition with $P = \emptyset$ is not consistent with KB (because of (1) and the fact that that $KB[P][P]$ only considers modal atoms in $MA(KB)$). The same restriction is no longer necessary for MKNF models for which the following one-to-one correspondence between every MKNF model $\mathcal{M}$ of KB and the partition induced by $\mathcal{M}$ can be shown.

Theorem 1. *A set $\mathcal{M}$ of interpretations over Δ is an MKNF model for a safe $e\mathcal{ALCO}$ KB KB iff the partition (P, N) of $MA(KB)$ induced by $\mathcal{M}$ satisfies the following:*
(1) (P, N) is consistent with KB;
(2) $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models Ob_{KB,P,P}\}$; and
(3) for each partition (P', N') of $MA(KB)$ such that $P' \subset P$, at least one of the following conditions does not hold:
  (a) the $\mathcal{ALCO}$ KB $Ob_{KB,P',P}$ is satisfiable;
  (b) $Ob_{KB,P',P} \not\models C(x)$ for each $\mathbf{K}C(x) \in N'$;
  (c) $Ob_{KB,P',P} \not\models R(x,y)$ for each $\mathbf{K}R(x,y) \in N'$.

As an immediate consequence of this procedure, we can show that safe $e\mathcal{ALCOV}$ KBs are $\mathcal{ALCOV}$ representable.

Corollary 1. *Let KB be a safe $e\mathcal{ALCOV}$ KB, $\mathcal{M}$ an MKNF model of KB, and (P, N) be the partition of $MA_{\Delta}(KB)$ induced by $\mathcal{M}$. Then $\mathcal{M} = \{\mathcal{I} \mid \mathcal{I} \models Ob_{KB,P,P}\}$.*

4 Coverage within the Decidable Fragment

The established result in the previous section is certainly interesting in its own right, since, arguably, the imposed restrictions and the applied construction is considerably less complicated in terms of notation than the one applied in [8,20]. Anyway, the main outlined purpose of this revision is to ensure that the new decidable fragment encompasses all the formalisms for which coverage was shown in [20] only for the full language. In this section, we revisit this material and discuss relevant changes.

4.1 Monotonic Approaches

Naturally, our decidable language fragment of safe $e\mathcal{ALCOV}$ KBs covers $\mathcal{ALCOV}$ (with and without U) and all its sub-languages. This does not include $\mathcal{SROIQ}$, i.e., OWL 2 DL and its tractable profiles, but a trivial adjustment following the ideas in [20] where modal operators are limited to $\mathcal{ALCOV}$ concepts is easily conceivable.

Coverage of RIF-CORE [3], i.e., n-ary Datalog, interpreted as DL-safe Rules [31] carries over from [20] or alternatively from [25]. In fact, the latter does not even require the usage of the universal role U which is just fine if we only want to embed a Datalog program. However, if we want to cover an embedding of n-ary Datalog interacting with DLs, then the former is required: consider the Datalog rule $C(a) \rightarrow D(a)$ and a concept assertion $C(a)$. The rule can be translated to $\exists atom.(C \sqcap \{a\}) \sqsubseteq \exists atom.(D \sqcap \{a\})$ (slightly adjusted from [25]), but there is $\mathcal{I}$ such that $atom^{\mathcal{I}} = \emptyset$, i.e., $D(a)$ is no longer derivable. Thus, based on the former Datalog embedding, coverage of DL-safe SWRL [31], $\mathcal{AL}$-log [7], and CARIN [26] carries over, i.e., without much surprise all monotonic approaches as outlined in [20] are covered.

4.2 $\mathcal{ALCK}_{\mathcal{NF}}$

In [8], it is shown how several non-monotonic reasoning features (defaults, integrity constraints, and role and concept closure) can be modeled in the full language $\mathcal{ALCK}_{\mathcal{NF}}$ and it is argued that the restriction to simple KBs applied to achieve decidability does not impede coverage. The full $e\mathcal{ALCOV}$ language obviously includes $\mathcal{ALCK}_{\mathcal{NF}}$ by design, but since we have changed the restrictions to achieve decidability, these results do not carry over automatically, so we briefly discuss coverage of these features for safe $e\mathcal{ALCOV}$ KBs (and refer the reader for the detailed discussion to [8]).

First, closed $\mathcal{DL}$-defaults [2] of the form

$$d = \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$$

are covered in [8], where α, β_i, and γ are $\mathcal{DL}$ concepts and $n \geq 0$. Closed defaults are limited in their applicability to individuals explicitly mentioned in the knowledge base. This is achieved in [8] by using a new atomic concept I in each translation of

a default and adding the assertions $I(a)$ for each a appearing in the knowledge base. Conceptually, this matches the idea of nominal schemas, so the translation of closed defaults can be presented as a safe $e\mathcal{ALCOV}$ axiom

$$\tau_{DK}(d) = \mathbf{K}\alpha \sqcap \neg\mathbf{A}\neg\beta_1 \sqcap \cdots \sqcap \neg\mathbf{A}\neg\beta_n \sqcap \{x\} \sqsubseteq \mathbf{K}\gamma \sqcap \{x\}$$

without the need to introduce new concepts or adding additional assertions, and it is easy to see that Theorem 3.1 from [8] can be adapted accordingly.

Theorem 2. *Let $\langle \Sigma, \mathcal{D}\rangle$ be an $\mathcal{ALC}$ KB with defaults, where Σ is an $\mathcal{ALC}$ KB and $\mathcal{D}$ is a set of $\mathcal{ALC}$-defaults. The $e\mathcal{ALCOV}$ KB $\tau(\Sigma, \mathcal{D})$ is such that, for every $\mathcal{ALC}$-concept C and every individual $a \in N_I$, it holds $\langle \Sigma, \mathcal{D}\rangle \models C(a)$ iff $\tau_{DK}(\Sigma, \mathcal{D}) \models C(a)$.*

Secondly, integrity constraints (ICs) are considered, and it is argued that ICs commonly apply to individuals explicitly mentioned in the considered KB and impose restrictions without changing the content of the KB. This is in line with our restrictions on safe $e\mathcal{ALCOV}$ KBs and it can be verified that all examples discussed in [8] can be made safe explicitly by introducing guards $\{x\}$ as for defaults. Finally, similar observations hold for the considerations on role and concept closure, i.e., all modeling features presented in [8] can indeed be adjusted to safe $e\mathcal{ALCOV}$ KBs without much effort.

4.3 Hybrid MKNF

Hybrid MKNF as a combination of DLs with non-monotonic rules is based on MKNF logics as well, but of different expressivity due to the different restrictions applied to the full MKNF language in each of the two approaches [30]. In [20], an embedding of hybrid MKNF into epistemic DLs is presented (we refer to that paper for the technical details). Though not the full language of hybrid MKNF is embedded, the presented fragment suffices to cover Answer Set Programming [11], i.e., disjunctive Datalog with classical negation and non-monotonic negation under the answer set semantics. Unfortunately, the presented embedding is in general not covered within the decidable fragment in [20] as shown with the simple example $\top \sqsubseteq \exists U.(\{a\} \sqcap C)$ and $\mathbf{K}D(a) \leftarrow \mathbf{K}C(a)$ as the latter would be embedded as $\mathbf{K}(\exists U.(\{a\} \sqcap C) \sqsubseteq \mathbf{K}(\exists U.(\{a\} \sqcap C)$ which is not simple [20]. This can be remedied with safe $e\mathcal{ALCOV}$ KB by changing the translation of MKNF rules $\mathsf{dl}(\mathbf{K}H_1 \vee \mathbf{K}H_l \leftarrow \mathbf{K}A_1, \ldots, \mathbf{K}A_n, \mathbf{not}B_1, \ldots, \mathbf{not}B_m)$ in [20] to

$$\mathbf{K}\mathsf{dl}(A_1) \sqcap \ldots \sqcap \mathbf{K}\mathsf{dl}(A_n) \sqcap \neg\mathbf{A}\mathsf{dl}(B_1) \sqcap \ldots \sqcap \neg\mathbf{A}\mathsf{dl}(B_m) \sqcap \{i\} \sqsubseteq \mathbf{K}\mathsf{dl}(H_1) \sqcup \ldots \sqcup \mathbf{K}\mathsf{dl}(H_l) \sqcap \{i\}$$

where i is a fresh individual and dl the translation function on (possibly classically negated) atoms defined in [20]. Essentially, in the original embedding, such translated concepts in GCIs would due to the universal role either be interpreted as Δ or $\emptyset$. Here, we introduce a nominal as guard that acts as a surrogate for all elements in Δ, thus reducing such interpretation to either i or $\emptyset$. An adaptation of the results in [20] (Lemma 3 and Theorem 4) are then straightforward.

Theorem 3. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF KB. $\mathcal{M}$ is an MKNF model of $\mathcal{K}$ iff $\mathcal{M}_1 = \{\mathcal{J} \mid \mathcal{J} \in fam(\mathcal{I})$ with $\mathcal{I} \in \mathcal{M}\}$ is a hybrid MKNF model of $\mathsf{dl}(\mathcal{K})$.*

This ensures that safe $e\mathcal{ALCOV}$ KBs in fact embed the restricted version hybrid MKNF and therefore also ASP.

5 Conclusions

We have studied epistemic extensions of DLs focusing here on $e\mathcal{ALCOV}$, i.e., $\mathcal{ALC}$ extended with nominals, nominals schemas, the universal role, and two epistemic operators for modeling non-monotonic reasoning. We have shown that this language encompasses all non-monotonic modeling features and approaches discussed in [20], and that an extension to a few missing monotonic languages (e.g., $\mathcal{SROIQ}$) is easily conceivable. We have introduced a set of restrictions on the general language which is different from that in [20], and we have shown that, under these restrictions, reasoning, i.e., checking MKNF-satisfiability becomes decidable, and, unlike in previous work, the restricted language still covers all the discussed modeling features.

An immediate matter for follow-up work is the computational complexity when reasoning with epistemic DLs, a question that has only received limited attention so far (in [8] a triple exponential space upper bound for reasoning with simple KBs has been pointed out, while no results are mentioned in [20]). It is clear that the complexity results established for reasoning with nominal schemas (without epistemic operators) [25] can serve as first necessary lower bounds, i.e., for $e\mathcal{ALCOV}$ in particular a minimal lower bound is established by the fact that reasoning in $\mathcal{ALCOV}$ is 2ExpTime-complete. This does neither account for the universal role nor the epistemic operators. Since $\mathcal{ALC}$ with arbitrary Boolean role constructors is NExpTime-complete [28,36] (as the restriction to safe Boolean role constructors in [36] does not suffice to cover U), and the decision procedure for MKNF-satisfiability requires nondeterministically guessing the right partition, by combining the different sources of complexity we conjecture a lower bound of at least N2ExpTime.

Another interesting topic for future work is to establish coverage for further related formalisms, for example by extending the expressiveness of rules permitted in the embedding of Hybrid MKNF, which then allows us to include already established embedding results for, e.g., [9,32] in [30] or by considering among others work on circumscription in DLs [4,33]. Building on the existing relation between epistemic extensions of DLs and Hybrid MKNF, we can also investigate the relation to parameterized logic programs [12,13], or multi-context systems [5] using the established connection between these and Hybrid MKNF [21]. Finally, an implementation may be considered given a) the more simple decision procedure proposed here and b) the recent work on implementing nominal schemas [22,37,6,34] and Hybrid MKNF [1,16]. In particular the encouraging results for Konclude [34,35] seem to indicate that this may in fact be achievable in a feasible manner.

Acknowledgments This work was partially supported by Fundação para a Ciência e a Tecnologia under project "ERRO – Efficient Reasoning with Rules and Ontologies" (PTDC/EIA-CCO/121823/2010) and strategic project PEst/UID/CEC/04516/2013, and by FCT grant SFRH/BPD/86970/2012.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Query-driven procedures for hybrid MKNF knowledge bases. ACM Trans. Comput. Log. 14(2), 16 (2013)
2. Baader, F., Hollunder, B.: Embedding defaults into terminological representation systems. Journal of Automated Reasoning 14, 149–180 (1995)
3. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Recommendation 22 June 2010 (2010), available from http://www.w3.org/TR/rif-core/
4. Bonatti, P.A., Lutz, C., Wolter, F.: The complexity of circumscription in dls. J. Artif. Intell. Res. (JAIR) 35, 717–773 (2009)
5. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence. pp. 385–390. AAAI Press (2007)
6. Carral, D., Wang, C., Hitzler, P.: Towards an efficient algorithm to reason over description logics extended with nominal schemas. In: Faber, W., Lembo, D. (eds.) Web Reasoning and Rule Systems - 7th International Conference, RR 2013. Proceedings. LNCS, vol. 7994, pp. 65–79. Springer (2013)
7. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: AL-log: Integrating datalog and description logics. Journal of Intelligent Information Systems 10(3), 227–252 (1998)
8. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic 3(2), 177–225 (2002)
9. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artificial Intelligence 172(12–13), 1495–1539 (2008)
10. Fitting, M.: First-order logic and automated theorem proving. Springer, 2nd edn. (1996)
11. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
12. Gonçalves, R., Alferes, J.J.: Parametrized logic programming. In: Janhunen, T., Niemelä, I. (eds.) Logics in Artificial Intelligence - 12th European Conference, JELIA 2010. Proceedings. LNCS, vol. 6341, pp. 182–194. Springer (2010)
13. Gonçalves, R., Alferes, J.J.: Decidability and implementation of parametrized logic programs. In: Cabalar, P., Son, T.C. (eds.) Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013. Proceedings. LNCS, vol. 8148, pp. 361–373. Springer (2013)
14. Grimm, S., Hitzler, P.: Semantic Matchmaking of Web Resources with Local Closed-World Reasoning. International Journal of Electronic Commerce 12(2), 89–126 (2008)
15. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009 (2009), available from http://www.w3.org/TR/owl2-primer/
16. Ivanov, V., Knorr, M., Leite, J.: A query tool for $\mathcal{EL}$ with non-monotonic rules. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Proceedings, Part I. LNCS, vol. 8218, pp. 216–231. Springer (2013)
17. Ke, P.: Nonmonotonic Reasoning with Description Logics. Ph.D. thesis, University of Manchester (2011)
18. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note 22 June 2010 (2010), available from http://www.w3.org/TR/rif-overview/
19. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artificial Intelligence 175(9–10), 1528–1554 (2011)

20. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the semantic web. In: Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI 2012 - 20th European Conference on Artificial Intelligence. Proceedings. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 474–479. IOS Press (2012)
21. Knorr, M., Slota, M., Leite, J., Homola, M.: What if no hybrid reasoner is available? hybrid MKNF in multi-context systems. J. Log. Comput. 24(6), 1279–1311 (2014)
22. Krisnadhi, A., Hitzler, P.: A tableau algorithm for description logics with nominal schema. In: Web Reasoning and Rule Systems - 6th International Conference, RR 2012. Proceedings. LNCS, vol. 7497, pp. 234–237. Springer (2012)
23. Krötzsch, M.: Description Logic Rules, Studies on the Semantic Web, vol. 008. IOS Press/AKA (2010)
24. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International World Wide Web Conference, WWW2011. pp. 645–654. ACM (2011)
25. Krötzsch, M., Rudolph, S.: Nominal schemas in description logics: Complexities clarified. In: Baral, C., Giacomo, G.D., Eiter, T. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014. AAAI Press (2014)
26. Levy, A.Y., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104, 165–209 (1998)
27. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) Proceedings of the 12th International Joint Conferences on Artifical Intelligence, IJCAI'91. pp. 381–386. Morgan Kaufmann (1991)
28. Lutz, C., Sattler, U.: The complexity of reasoning with Boolean modal logics. In: Wolter, F., Wansing, H., de Rijke, M., Zakharyaschev, M. (eds.) Proc. of the 3rd Int. Conf. on Advances in Modal Logic (AiML 2000). pp. 329–348. World Scientific (2002)
29. Mehdi, A., Rudolph, S.: Revisiting semantics for epistemic extensions of description logics. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011. AAAI Press (2011)
30. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. Journal of the ACM 57(5), 93–154 (2010)
31. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)
32. Rosati, R.: DL+Log: A tight integration of description logics and disjunctive datalog. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) 10th International Conference on the Principles of Knowledge Representation and Reasoning, (KR'06), Proceedings. pp. 68–78. AAAI Press (2006)
33. Sengupta, K., Krisnadhi, A.A., Hitzler, P.: Local closed world semantics: Grounded circumscription for OWL. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (eds.) The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Proceedings. LNCS, vol. 7031, pp. 617–632. Springer (2011)
34. Steigmiller, A., Glimm, B., Liebig, T.: Reasoning with nominal schemas through absorption. J. Autom. Reasoning 53(4), 351–405 (2014)
35. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. Web Sem. 27, 78–85 (2014)
36. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)
37. Wang, C., Hitzler, P.: A resolution procedure for description logics with nominal schemas. In: Takeda, H., Qu, Y., Mizoguchi, R., Kitamura, Y. (eds.) Second Joint International Conference, JIST 2012. Proceedings. LNCS, vol. 7774, pp. 1–16. Springer (2012)

Conservative Rewritability of Description Logic TBoxes: First Results

Boris Konev[1], Carsten Lutz[2], Frank Wolter[1], and Michael Zakharyaschev[3]

[1] Department of Computer Science, University of Liverpool, U.K.
[2] Fachbereich Informatik, Universität Bremen, Germany
[3] Department of Computer Science and Information Systems, Birkbeck, University of London, U.K.

Abstract. We want to understand when a given TBox $\mathcal{T}$ in a description logic $\mathcal{L}$ can be rewritten into a TBox $\mathcal{T}'$ in a weaker description logic $\mathcal{L}'$. Two notions of rewritability are considered: model-conservative rewritability ($\mathcal{T}'$ entails $\mathcal{T}$ and all models of $\mathcal{T}$ can be expanded to models of $\mathcal{T}'$) and $\mathcal{L}$-conservative rewritability ($\mathcal{T}'$ entails $\mathcal{T}$ and every $\mathcal{L}$-consequence of $\mathcal{T}'$ in the signature of $\mathcal{T}$ is a consequence of $\mathcal{T}$) and investigate rewritability of TBoxes in $\mathcal{ALCI}$ to $\mathcal{ALC}$, $\mathcal{ALCQ}$ to $\mathcal{ALC}$, $\mathcal{ALC}$ to $\mathcal{EL}_\bot$, and $\mathcal{ALCI}$ to *DL-Lite*$_{horn}$. We compare conservative rewritability with equivalent rewritability, give model-theoretic characterizations of conservative rewritability, prove complexity results for deciding rewritability, and provide some rewriting algorithms.

Over the past 30 years, a multitude of different description logics (DLs) have been designed, investigated, and used in practice as ontology languages. The introduction of new DLs has been driven both by the need for additional expressive power (such as transitive roles in the 1990s) and by applications that require efficient reasoning of a novel type (such as ontology-based data access in the 2000s). While the resulting flexibility in choosing DLs has had the positive effect of making DLs available for a large number of domains and applications, it has also led to the development of ontologies with language constructors that are not really required to axiomatize their knowledge. For a constructor to be 'not required' can mean different things here, ranging from the high-level 'this domain can be represented in an adequate way in a weaker DL' to the very concrete 'this ontology is logically equivalent to an ontology in a weaker DL'. In this paper, we take the latter understanding as our starting point. Equivalent rewritability of a given DL ontology (TBox) to a weaker DL has been investigated in [17], where model-theoretic characterizations and the complexity of deciding rewritability were investigated. For example, equivalent rewritability of an $\mathcal{ALC}$ TBox to an $\mathcal{EL}_\bot$ TBox has been characterized in terms of preservation under products and global equisimilations, and a NExpTime upper bound for deciding equivalent rewritability has been established. Equivalent rewritability is a very strong notion, however, that appears to apply to a very small number of real-world TBoxes. A more practically relevant notion we propose in this paper is *conservative* rewritability, which allows one to use new concept and

role names when rewriting a given ontology into a weaker DL. In this case, we clearly cannot demand that the new TBox is logically equivalent to the original one, but only that it entails the original TBox. To avoid uncontrolled additional consequences of the new TBox, we can also require that (i) it does not entail any new consequences in the language of the original TBox, or even that (ii) every model of the original TBox can be expanded a model of the new TBox. The latter type of conservative extension is known as *model-conservative extension* [16], and we call a TBox $\mathcal{T}$ model-conservatively $\mathcal{L}$-rewritable if a model-conservative rewriting of $\mathcal{T}$ in the DL $\mathcal{L}$ exists. The former type of conservative extension is known as a *language-conservative extension* or *deductive conservative extension* [12] and, given a DL $\mathcal{L}$ in which $\mathcal{T}$ is formulated and a weaker DL $\mathcal{L}'$, we call $\mathcal{T}$ $\mathcal{L}$-conservatively $\mathcal{L}'$-rewritable if there is a TBox $\mathcal{T}'$ in $\mathcal{L}'$ such that $\mathcal{T}'$ has the same $\mathcal{L}$-consequences as $\mathcal{T}$ in the signature of $\mathcal{T}$. Model-conservative rewritability is the more robust notion as it is language-independent and does not only leave unchanged the entailed concept inclusions of the original TBox but also, for example, certain answers if the ontologies are used to access data.

The main result of this paper is that there are important DLs for which model-conservative and $\mathcal{L}$-conservative rewritability can be transparently characterized, effectively decided, and for which rewriting algorithms can be designed. This is in contrast to the undecidability of the problem whether one TBox is a model-conservative extension of another one even for weak DLs such as $\mathcal{EL}$ [18, 16]. In particular, we show that, given an $\mathcal{ALCI}$ TBox, one can compute in polynomial time its model-conservative $\mathcal{ALC}$-rewriting provided that such a rewriting exists, which can be decided in ExpTime. We characterize model-conservative $\mathcal{ALC}$-rewritability in terms of preservation under generated subinterpretations and show that $\mathcal{ALCI}$-conservative $\mathcal{ALC}$-rewritability coincides with model-conservative one. For $\mathcal{ALCQ}$ TBoxes, we show that model-conservative $\mathcal{ALC}$-rewritability coincides with equivalent rewritability, but is different from $\mathcal{ALCQ}$-conservative rewritability. The latter can be characterized using bounded morphisms, and all these notions of rewritability are decidable in 2ExpTime. Unlike the $\mathcal{ALCI}$ case, we currently do not have polynomial rewritings for $\mathcal{ALCQ}$ TBoxes. As to rewritability from $\mathcal{ALCI}$ to *DL-Lite*$_{horn}$, we observe that all our notions of rewritability coincide and are ExpTime-complete. In contrast, for rewritability from $\mathcal{ALC}$ to $\mathcal{EL}_\bot$ they are all distinct and, in fact, rather intricate and difficult to analyse. We prove decidability of model-conservative rewritability and give necessary semantic conditions for both $\mathcal{ALC}$-conservative and model-conservative $\mathcal{EL}_\bot$-rewritability.

Related work. Conservative rewritings of TBoxes are ubiquitous in the DL research. For example, many rewritings of TBoxes into normal forms are model-conservative [14, 4]. Regarding rewritability of TBoxes into weaker DLs, the focus has been on polynomial satisfability preserving rewritings as a pre-processing step to reasoning [11, 9, 8] or to prove complexity results for reasoning [10]. Such rewritings are mostly not conservative. There has been significant work on rewritings of ontology-mediated queries (pairs of ontologies and queries), which preserve their certain answers, into datalog or ontology-mediated queries based on

weaker DLs [13, 5]. It seems, however, that this problem is different from TBox conservative rewritability. In [2], the expressive power of DLs and corresponding notions of rewritability are introduced based on a variant of model-conservative extension, and the relationship to $\mathcal{L}$-conservative extensions is discussed.

For omitted proofs, see http://cgi.csc.liv.ac.uk/~frank/publ/publ.html.

1 Conservative Rewritability

We consider the standard description logics $\mathcal{ALC}$, $\mathcal{ALCI}$, $\mathcal{ALCQ}$, $\mathcal{EL}_\bot$, and *DL-Lite*$_{horn}$ [3, 4, 7, 1], where $\mathcal{EL}_\bot$ is $\mathcal{EL}$ extended with the concept $\bot$, and *DL-Lite*$_{horn}$ is *DL-Lite*$_{core}$ extended with conjunctions of basic concepts on the left-hand side of concept inclusions. As usual, the alphabet of DLs consists of countably infinite sets $\mathsf{N_C}$ of *concept names* and $\mathsf{N_R}$ of *role names*. By a *signature*, Σ, we mean any set of concept and role names. The *signature* $\mathsf{sig}(\mathcal{T})$ *of a TBox* $\mathcal{T}$ is the set of concept and role names occurring in $\mathcal{T}$.

Before introducing our notions of conservative rewritability, we remind the reader of a simpler notion of TBox rewritability. Suppose $\mathcal{L}$ and $\mathcal{L}'$ are DLs; we typically assume that $\mathcal{L}$ is more expressive than $\mathcal{L}'$.

Definition 1 (equivalent $\mathcal{L}$-to-$\mathcal{L}'$ rewritability). An $\mathcal{L}'$ TBox $\mathcal{T}'$ is called an *equivalent $\mathcal{L}'$-rewriting* of an $\mathcal{L}$ TBox $\mathcal{T}$ if $\mathcal{T} \models \mathcal{T}'$ and $\mathcal{T}' \models \mathcal{T}$ (in other words, if $\mathcal{T}$ and $\mathcal{T}'$ have the same models). An $\mathcal{L}$ TBox is called *equivalently $\mathcal{L}'$-rewritable* if it has an equivalent $\mathcal{L}'$-rewriting.

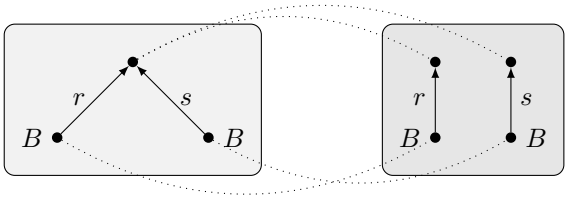
Equivalent $\mathcal{L}$-to-$\mathcal{L}'$ rewritability has been studied in [17], where semantic characterizations are given and complexity results for deciding equivalent rewritability are obtained for various DLs $\mathcal{L}$ and $\mathcal{L}'$. For example, if $\mathcal{L}$ is $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ and $\mathcal{L}'$ is $\mathcal{ALC}$, then an $\mathcal{L}$ TBox $\mathcal{T}$ is equivalently $\mathcal{L}'$-rewritable just in case its class of models is preserved under global bisimulations, which are defined as follows. Given interpretations $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$, for $i = 1, 2$, and a signature Σ, we call a relation $S \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ a *Σ-bisimulation between $\mathcal{I}_1$ and $\mathcal{I}_2$* if

- for any $A \in \Sigma$, whenever $(d_1, d_2) \in S$ then $d_1 \in A^{\mathcal{I}_1}$ iff $d_2 \in A^{\mathcal{I}_2}$;
- for any $r \in \Sigma$ and $(d_1, d_2) \in S$,
  - if $(d_1, e_1) \in r^{\mathcal{I}_1}$ then there is e_2 such that $(e_1, e_2) \in S$ and $(d_2, e_2) \in r^{\mathcal{I}_2}$,
  - if $(d_2, e_2) \in r^{\mathcal{I}_2}$ then there is e_1 such that $(e_1, e_2) \in S$ and $(d_1, e_1) \in r^{\mathcal{I}_1}$.

S is a *global* Σ-bisimulation between $\mathcal{I}_1$ and $\mathcal{I}_2$ if $\Delta^{\mathcal{I}_1}$ is the domain of S and $\Delta^{\mathcal{I}_2}$ its range. $\mathcal{I}_1$ and $\mathcal{I}_2$ are *globally Σ-bisimilar* if there is a global Σ-bisimulation between them, in which case we write $\mathcal{I}_1 \sim^{\Sigma}_{\mathcal{ALC}} \mathcal{I}_2$. For $d_1 \in \Delta^{\mathcal{I}_1}$ and $d_2 \in \Delta^{\mathcal{I}_2}$, we say that $(\mathcal{I}_1, d_1)$ is *Σ-bisimilar to* $(\mathcal{I}_2, d_2)$ if there is a Σ-bisimulation S between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $(d_1, d_2) \in S$. If $\Sigma = \mathsf{N_C} \cup \mathsf{N_R}$, we omit Σ, write $\mathcal{I}_1 \sim_{\mathcal{ALC}} \mathcal{I}_2$ and say simply '(global) bisimulation.'

Example 1. The $\mathcal{ALCI}$ TBox $\{\exists r^-.B \sqsubseteq A\}$ can be equivalently rewritten to the $\mathcal{ALC}$ TBox $\{B \sqsubseteq \forall r.A\}$. However, the $\mathcal{ALCI}$ TBox $\mathcal{T} = \{\exists r^-.B \sqcap \exists s^-.B \sqsubseteq A\}$ is not equivalently $\mathcal{ALC}$-rewritable. Indeed, the interpretation on the right-hand

side in the picture below is a model of $\mathcal{T}$ and globally bisimilar to the interpretation on the left-hand side, which is not a model of $\mathcal{T}$.



We now introduce two subtler notions of TBox rewritability, which allow the use of fresh concept and role names in rewritings. For an interpretation $\mathcal{I}$ and signature Σ, the *Σ-reduct of $\mathcal{I}$* is the interpretation $\mathcal{I}_{|\Sigma}$ coinciding with $\mathcal{I}$ on the names in Σ and having $X^{\mathcal{I}_{|\Sigma}} = \emptyset$ for all $X \notin \Sigma$. We say that interpretations $\mathcal{I}$ and $\mathcal{J}$ *coincide on* Σ and write $\mathcal{I} =_\Sigma \mathcal{J}$ if the Σ-reducts of $\mathcal{I}$ and $\mathcal{J}$ coincide. A TBox $\mathcal{T}'$ is a *model-conservative extension* of $\mathcal{T}$ if an interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ just in case there is a model $\mathcal{I}'$ of $\mathcal{T}'$ such that $\mathcal{I} =_{\mathsf{sig}(\mathcal{T})} \mathcal{I}'$.

Definition 2 (model-conservative $\mathcal{L}$-to-$\mathcal{L}'$-rewritability). An $\mathcal{L}'$ TBox $\mathcal{T}'$ is called a *model-conservative $\mathcal{L}'$-rewriting* of an $\mathcal{L}$ TBox $\mathcal{T}$ if $\mathcal{T}'$ is a model-conservative extension of $\mathcal{T}$. An $\mathcal{L}$ TBox $\mathcal{T}$ is *model-conservatively $\mathcal{L}'$-rewritable* if a model-conservative $\mathcal{L}'$-rewriting of $\mathcal{T}$ exists.

Clearly, any equivalent $\mathcal{L}'$-rewriting of a TBox $\mathcal{T}$ is also a model-conservative $\mathcal{L}'$-rewriting of $\mathcal{T}$. The next example shows that the converse does not hold.

Example 2. The $\mathcal{ALCI}$ TBox $\mathcal{T} = \{\exists r^-.B \sqcap \exists s^-.B \sqsubseteq A\}$ from Example 1 is model-conservatively $\mathcal{ALC}$-rewritable to

$$\mathcal{T}' = \{B \sqsubseteq \forall r.B_{\exists r^-.B},\ B \sqsubseteq \forall s.B_{\exists s^-.B},\ B_{\exists r^-.B} \sqcap B_{\exists s^-.B} \sqsubseteq A\},$$

where $B_{\exists r^-.B}$, $B_{\exists s^-.B}$ are *fresh* concept names.

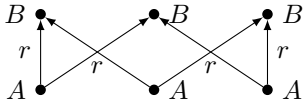
A TBox $\mathcal{T}'$ is called an *$\mathcal{L}$-conservative extension* of $\mathcal{T}$ if $\mathcal{T}' \models \mathcal{T}$ and $\mathcal{T}' \models C \sqsubseteq D$ implies $\mathcal{T} \models C \sqsubseteq D$, for every $\mathcal{L}$-concept inclusion $C \sqsubseteq D$ formulated in $\mathsf{sig}(\mathcal{T})$.

Definition 3 ($\mathcal{L}$-conservative $\mathcal{L}'$-rewritability). An $\mathcal{L}'$ TBox $\mathcal{T}'$ is called an *$\mathcal{L}$-conservative $\mathcal{L}'$-rewriting* of an $\mathcal{L}$ TBox $\mathcal{T}$ if $\mathcal{T}'$ is an $\mathcal{L}$-conservative extension of $\mathcal{T}$. An $\mathcal{L}$ TBox $\mathcal{T}$ is *$\mathcal{L}$-conservatively $\mathcal{L}'$-rewritable* if an $\mathcal{L}$-conservative $\mathcal{L}'$-rewriting of $\mathcal{T}$ exists.

It should be clear that every model-conservative $\mathcal{L}'$-rewriting of an $\mathcal{L}$ TBox $\mathcal{T}$ is also an $\mathcal{L}$-conservative $\mathcal{L}'$-rewriting of $\mathcal{T}$. The next example shows that the converse implication does not hold.

Example 3. The $\mathcal{ALCQ}$ TBox $\mathcal{T} = \{A \sqsubseteq \geq 2\, r.B\}$ is $\mathcal{ALCQ}$-conservatively $\mathcal{ALC}$-rewritable to $\mathcal{T}' = \{A \sqsubseteq \exists r.C,\ A \sqsubseteq \exists r.D,\ C \sqsubseteq \neg D,\ C \sqcup D \sqsubseteq B\}$, where C and D are *fresh* concept names. However, $\mathcal{T}'$ is not a model-conservative rewriting of $\mathcal{T}$ because the model of $\mathcal{T}$ shown below is not the $\mathsf{sig}(\mathcal{T})$-reduct of any model

of $\mathcal{T}'$. Note that $\mathcal{T}$ is not equivalently $\mathcal{ALC}$-rewritable.



In our examples so far, we have used fresh concept names but no fresh role names. This is no accident: it turns out that, for the DLs considered in this paper, fresh role names in conservative rewritings are not required. More precisely, we call a model-conservative or $\mathcal{L}$-conservative $\mathcal{L}'$-rewriting $\mathcal{T}'$ of $\mathcal{T}$ a model-conservative or, respectively, $\mathcal{L}$-conservative $\mathcal{L}'$-*concept rewriting* of $\mathcal{T}$ if $\mathsf{sig}_R(\mathcal{T}) = \mathsf{sig}_R(\mathcal{T}')$, where $\mathsf{sig}_R(\mathcal{T})$ is the set of role names in $\mathcal{T}$.

Say that a DL $\mathcal{L}$ *reflects disjoint unions* if, for any $\mathcal{L}$ TBox $\mathcal{T}$, whenever the *disjoint* union $\bigcup_{i\in I}\mathcal{I}_i$ of interpretations $\mathcal{I}_i$ is a model of $\mathcal{T}$, then each $\mathcal{I}_i$, $i \in I$, is also a model of $\mathcal{T}$. All the DLs considered in this paper reflect disjoint unions.

Theorem 1. *Let $\mathcal{L}$ be a DL reflecting disjoint unions, $\mathcal{T}$ an $\mathcal{L}$ TBox, and let $\mathcal{L}' \in \{\mathcal{ALC}, \mathcal{EL}_\bot, \text{DL-Lite}_{horn}\}$. Then $\mathcal{T}$ is model-conservatively (or $\mathcal{L}$-conservatively) $\mathcal{L}'$-rewritable if and only if it is model-conservatively (or, respectively, $\mathcal{L}$-conservatively) $\mathcal{L}'$-concept rewritable.*

2 $\mathcal{ALCI}$-to-$\mathcal{ALC}$ Rewritability

Equivalent $\mathcal{ALCI}$-to-$\mathcal{ALC}$ rewritability was studied in [17], where the characterization in terms of global bisimulations was used to design a 2ExpTime algorithm for checking this property. Here, we give a characterization of model-conservative $\mathcal{ALC}$ rewritability of $\mathcal{ALCI}$ TBoxes in terms of generated subinterpretations and use it to show that (i) model-conservative $\mathcal{ALCI}$-to-$\mathcal{ALC}$ rewritings are of polynomial size and can be constructed in polynomial time (if they exist), and that (ii) deciding model-conservative $\mathcal{ALCI}$-to-$\mathcal{ALC}$ rewritability is ExpTime-complete. We also observe that $\mathcal{ALCI}$-conservative $\mathcal{ALC}$-rewritability coincides with model-conservative rewritability.

We remind the reader that an interpretation $\mathcal{I}$ is a *subinterpretation* of an interpretation $\mathcal{J}$ if $\Delta^{\mathcal{I}} \subseteq \Delta^{\mathcal{J}}$, $A^{\mathcal{I}} = A^{\mathcal{J}} \cap \Delta^{\mathcal{I}}$ for all concept names A, and $r^{\mathcal{I}} = r^{\mathcal{J}} \cap (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ for all role names r. $\mathcal{I}$ is a *generated subinterpretation* of $\mathcal{J}$ if, in addition, whenever $d \in \Delta^{\mathcal{I}}$ and $(d, d') \in r^{\mathcal{J}}$, r a role name, then $d' \in \Delta^{\mathcal{I}}$. We say that a TBox $\mathcal{T}$ is *preserved under generated subinterpretations* if every generated subinterpretation of a model of $\mathcal{T}$ is also a model of $\mathcal{T}$. As well known, every $\mathcal{ALC}$ TBox is preserved under generated subinterpretations.

Suppose we want to find a model-conservative $\mathcal{ALC}$-rewriting of an $\mathcal{ALCI}$ TBox $\mathcal{T}$. Without loss of generality, we assume that $\mathcal{T} = \{\top \sqsubseteq C_{\mathcal{T}}\}$ and $C_{\mathcal{T}}$ is built using $\neg$, $\sqcap$ and $\exists$ only. Let $\mathsf{sub}(\mathcal{T})$ be the closure under single negation of the set of (subconcepts) of concepts in $\mathcal{T}$. For every role name r in $\mathcal{T}$, we take a fresh role name $\bar{r}$ and, for every $\exists r.C$ in $\mathsf{sub}(\mathcal{T})$ (where r is a role name or its inverse), we take a fresh concept name $B_{\exists r.C}$. Denote by $D^\sharp$ the $\mathcal{ALC}$-concept obtained from any $D \in \mathsf{sub}(\mathcal{T})$ by replacing every top-most occurrence

of a subconcept of the form $\exists r.C$ in it with $B_{\exists r.C}$. Now, let $\mathcal{T}^\dagger$ be an $\mathcal{ALC}$ TBox comprised of the following concept inclusions, for $r \in \mathsf{N_R}$: $\top \sqsubseteq C^\sharp_{\mathcal{T}}$,

$$C^\sharp \sqsubseteq \forall \bar{r}.B_{\exists r.C}, \qquad B_{\exists r.C} \equiv \exists r.C^\sharp, \qquad \text{for every } \exists r.C \in \mathsf{sub}(\mathcal{T}),$$
$$C^\sharp \sqsubseteq \forall r.B_{\exists r^-.C}, \quad B_{\exists r^-.C} \equiv \exists \bar{r}.C^\sharp, \qquad \text{for every } \exists r^-.C \in \mathsf{sub}(\mathcal{T}).$$

Clearly, $\mathcal{T}^\dagger$ can be constructed in polynomial time in the size of $\mathcal{T}$.

Theorem 2. *An $\mathcal{ALCI}$ TBox $\mathcal{T}$ is model-conservatively $\mathcal{ALC}$-rewritable iff $\mathcal{T}$ is preserved generated subinterpretations. Moreover, if $\mathcal{T}$ is model-conservatively $\mathcal{ALC}$-rewritable, then $\mathcal{T}^\dagger$ is its model-conservative $\mathcal{ALC}$-rewriting.*

It is now easy to show that model-conservative $\mathcal{ALCI}$-to-$\mathcal{ALC}$ rewritability is decidable in ExpTime. By Theorem 2, this amounts to deciding whether $\mathcal{T}^\dagger$ is a model-conservative extension of $\mathcal{T}$. In general, this is an undecidable problem. It is, however, easy to see that, for every model $\mathcal{I}$ of $\mathcal{T}$, there is a model $\mathcal{I}'$ of $\mathcal{T}^\dagger$ such that $\mathcal{I} =_{\mathsf{sig}(\mathcal{T})} \mathcal{I}'$. It thus remains to decide whether every interpretation $\mathcal{I}$ with $\mathcal{I} =_{\mathsf{sig}(\mathcal{T})} \mathcal{I}'$, for some model $\mathcal{I}'$ of $\mathcal{T}^\dagger$, is a model of $\mathcal{T}$. In other words, this means to decide whether $\mathcal{T}^\dagger \models \mathcal{T}$, which can be done in ExpTime. A matching lower bound is easily obtained by reducing satisfiability in $\mathcal{ALC}$.

Corollary 1. *The problem of deciding model-conservative $\mathcal{ALCI}$-to-$\mathcal{ALC}$ rewritability is* ExpTime*-complete.*

$\mathcal{ALCI}$-conservative $\mathcal{ALC}$-rewritability of $\mathcal{ALCI}$ TBoxes coincides with model-conservative $\mathcal{ALC}$-rewritability. This can be proved using the characterization via subinterpretations and robustness under replacement of $\mathcal{ALCI}$ TBoxes, an important property in the context of modular ontology design [15, Theorem 4].

Theorem 3. *An $\mathcal{ALCI}$ TBox $\mathcal{T}$ is $\mathcal{ALCI}$-conservatively $\mathcal{ALC}$-rewritable iff $\mathcal{T}$ is model-conservatively $\mathcal{ALC}$-rewritable.*

3 $\mathcal{ALCQ}$-to-$\mathcal{ALC}$ Rewritability

Equivalent $\mathcal{ALCQ}$-to-$\mathcal{ALC}$ rewritability was characterized in [17] in terms of preservation under global bisimulations. Below, we use this characterization to give a 2ExpTime algorithm for checking equivalent $\mathcal{ALC}$-rewritability.

We first prove a characterization of $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewritability in terms of preservation under inverse bounded morphisms and use it to show that one can (*i*) decide $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewritability in 2ExpTime and (*ii*) construct effectively an $\mathcal{ALCQ}$-conservative rewriting if it exists. We also show that, unlike $\mathcal{ALCI}$-to-$\mathcal{ALC}$-rewritability, model-conservative $\mathcal{ALC}$-rewritability of $\mathcal{ALCQ}$ TBoxes coincides with equivalent rewritability.

A *bounded Σ-morphism* from an interpretation $\mathcal{I}_1$ to an interpretation $\mathcal{I}_2$ is a global Σ-bisimulation S between $\mathcal{I}_1$ and $\mathcal{I}_2$ such that S is a function from $\Delta^{\mathcal{I}_1}$ to $\Delta^{\mathcal{I}_2}$. A class $\mathcal{K}$ of interpretations is *preserved under inverse bounded Σ-morphisms* if whenever there is a bounded Σ-morphism from an interpretation $\mathcal{I}_1$ to some $\mathcal{I}_2 \in \mathcal{K}$, then $\mathcal{I}_1 \in \mathcal{K}$. The following lemma provides the fundamental property of bounded morphisms:

Lemma 1. *Suppose $f : \mathcal{I}_1 \to \mathcal{I}_2$ is a bounded Σ-morphism, where $\mathcal{I}_2$ is a model of an $\mathcal{ALC}$ TBox $\mathcal{T}$ and $\mathsf{sig}_R(\mathcal{T}) \subseteq \Sigma$. Then there is $\mathcal{J}_1 \models \mathcal{T}$ such that $\mathcal{J}_1 =_\Sigma \mathcal{I}_1$.*

Proof. We define $\mathcal{J}_1$ in the same way as $\mathcal{I}_1$ except that $B^{\mathcal{J}_1} := f^{-1}(B^{\mathcal{I}_2})$ for all concept names $B \in \mathsf{sig}(\mathcal{T}_1) \setminus \Sigma$. Then f is a bounded $\mathsf{sig}(\mathcal{T})$-morphism from $\mathcal{J}_1$ to $\mathcal{I}_2$. Thus, $\mathcal{J}_1$ is a model of $\mathcal{T}$ since $\mathcal{I}_1$ is a model of $\mathcal{T}$. ❑

An interpretation $\mathcal{I}$ is a *directed tree interpretation* if $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$, for $r \neq s$, and the directed graph with nodes $\Delta^{\mathcal{I}}$ and edges E defined by setting $(d, d') \in E$ iff $(d, d') \in \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}}$ is a directed tree. We start our investigation with the observation that $\mathcal{ALCQ}$-conservative $\mathcal{ALCQ}$-to-$\mathcal{ALC}$ rewritability can be regarded as a principled approximation of model-conservative rewritability:

Lemma 2. *An $\mathcal{ALC}$ TBox $\mathcal{T}'$ is an $\mathcal{ALCQ}$-conservative rewriting of an $\mathcal{ALCQ}$ TBox $\mathcal{T}$ iff $\mathcal{T}'$ is a model-conservative rewriting of $\mathcal{T}$ over the class of directed tree interpretations of finite outdegree.*

Suppose we want to find an $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewriting of an $\mathcal{ALCQ}$ TBox $\mathcal{T}$. Without loss of generality, we assume that $\mathcal{T}$ is of the form $\{\top \sqsubseteq C_{\mathcal{T}}\}$ and that $C_{\mathcal{T}}$ is built using $\neg$, $\sqcap$, $(\geqslant n\ r\ C)$ only. Construct a TBox $\mathcal{T}^\dagger$ as follows. Take fresh concept names $B_D, B_1^D, \ldots, B_n^D$ for every $D = (\geqslant n\ r\ C) \in \mathsf{sub}(\mathcal{T})$. We use Σ to denote $\mathsf{sig}(\mathcal{T})$ extended with all fresh concept names of the form B_i^D. For each $C \in \mathsf{sub}(\mathcal{T})$, $C^\sharp$ denotes the $\mathcal{ALC}$-concept that results from C by replacing all top-most occurrences of any $D = (\geqslant n\ r\ C)$ in $\mathcal{T}$ with B_D. Now, define $\mathcal{T}^\dagger$ to be the infinite TBox that consists of the following inclusions:

- $\top \sqsubseteq C_{\mathcal{T}}^\sharp$,
- $B_D \sqsubseteq \exists r.(C^\sharp \sqcap B_1^D) \sqcap \cdots \sqcap \exists r.(C^\sharp \sqcap B_n^D)$,
- $B_i^D \sqsubseteq \neg B_j^D$, for $i \neq j$, and
- for all $\mathcal{ALC}$-concepts $C_1, \ldots, C_n$ in Σ and all $D = (\geqslant n\ r\ C) \in \mathsf{sub}(\mathcal{T})$,

$$\bigsqcap_{1 \leq i \leq n} (\exists r.(C^\sharp \sqcap C_i^\sharp \sqcap \bigsqcap_{j \neq i} \neg C_j^\sharp)) \sqsubseteq B_D.$$

The next theorem characterizes $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewritability.

Theorem 4. *An $\mathcal{ALCQ}$ TBox $\mathcal{T}$ is $\mathcal{ALCQ}$-conservatively $\mathcal{ALC}$-rewritable iff $\mathcal{T}$ is preserved under inverse bounded $\mathsf{sig}(\mathcal{T})$-morphisms. Moreover, if $\mathcal{T}$ is $\mathcal{ALCQ}$-conservatively $\mathcal{ALC}$-rewritable, then $\mathcal{T}^\dagger$ is an (infinite) rewriting.*

The semantic characterization of Theorem 4 can be employed to prove the following complexity result using a type elimination argument. We assume that numbers in number restrictions are given in unary.

Theorem 5. *For $\mathcal{ALCQ}$ TBoxes, $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewritability is decidable in 2*ExpTime.

It follows that, given an $\mathcal{ALCQ}$ TBox $\mathcal{T}$, one can first decide $\mathcal{ALCQ}$-conservative $\mathcal{ALC}$-rewritability and then, in case of a positive answer, effectively construct a rewriting by going through the finite subsets of $\mathcal{T}^\dagger$ in a systematic way until a finite $\mathcal{T}' \subseteq \mathcal{T}^\dagger$ with $\mathcal{T}' \models \mathcal{T}$ is reached. By compactness, such a set $\mathcal{T}'$ exists.

We finally show that every model-conservatively $\mathcal{ALC}$-rewritable $\mathcal{ALCQ}$ TBox is equivalently $\mathcal{ALC}$-rewritable.

Theorem 6. *An $\mathcal{ALCQ}$ TBox is model-conservatively $\mathcal{ALC}$-rewritable iff it is equivalently $\mathcal{ALC}$-rewritable, which is decidable in 2*ExpTime.

4 $\mathcal{ALCI}$-to-*DL-Lite*$_{horn}$ and $\mathcal{ALC}$-to-$\mathcal{EL}_\bot$ Rewritability

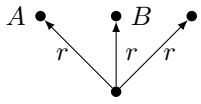
We first observe that all notions of rewritability introduced in this paper coincide in the case of $\mathcal{ALCI}$-to-*DL-Lite*$_{horn}$ rewritability. Deciding rewritability is ExpTime-complete in all cases since deciding equivalent $\mathcal{ALCI}$-to-*DL-Lite*$_{horn}$ rewritability is ExpTime-complete [17]:

Theorem 7. *For $\mathcal{ALCI}$ TBoxes, equivalent DL-Lite$_{horn}$-rewritability, model-conservative DL-Lite$_{horn}$-rewritability, and $\mathcal{ALCI}$-conservative DL-Lite$_{horn}$-rewritability coincide and are* ExpTime*-complete.*

We now provide separating examples for all three notions of $\mathcal{ALC}$-to-$\mathcal{EL}_\bot$ rewritability and then prove decidability of model-conservative $\mathcal{EL}_\bot$-rewritability. While we have not yet been able to find purely model-theoretic characterizations of model- and $\mathcal{ALC}$-conservative $\mathcal{EL}_\bot$-rewritability, we then give necessary model-theoretic conditions for these two notions of rewritability.

Equivalent $\mathcal{ALC}$-to-$\mathcal{EL}_\bot$ rewritability has been characterized in [17] in terms of preservation under products and global equisimulations. A *simulation* between interpretations $\mathcal{I}$ and $\mathcal{J}$ is a relation $S \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{J}$ such that, for any $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$ and $(d_1, d_2) \in S$, if $d_1 \in A^{\mathcal{I}_1}$ then $d_2 \in A^{\mathcal{I}_2}$, and if $(d_1, e_1) \in r^\mathcal{I}$ then there exists e_2 with $(e_1, e_2) \in S$ and $(d_2, e_2) \in r^\mathcal{J}$. $(\mathcal{I}, d)$ is *simulated* by $(\mathcal{J}, e)$ if there is a simulation S between $\mathcal{I}$ and $\mathcal{J}$ such that $(d, e) \in S$. Interpretations $\mathcal{I}$ and $\mathcal{J}$ are *globally equisimilar* if, for any $d \in \Delta^\mathcal{I}$, there exists $e \in \Delta^\mathcal{J}$ such that $(\mathcal{I}, d)$ is simulated by $(\mathcal{J}, e)$ and $(\mathcal{J}, e)$ is simulated by $(\mathcal{I}, d)$. According to [17, Theorem 17], an $\mathcal{ALC}$ TBox is equivalently $\mathcal{EL}_\bot$-rewritable if its models are preserved under products and global equisimulations.

Example 4. The TBox $\mathcal{T} = \{\exists r.A \sqcap \exists r.B \sqcap \forall r.(A \sqcup B) \sqsubseteq E \sqcup F,\ A \sqcap B \sqsubseteq \bot\}$ is not equivalently $\mathcal{EL}_\bot$-rewritable because its models are not preserved under global equisimulations. Indeed, the interpretation $\mathcal{I}$ shown below is clearly a model of $\mathcal{T}$. However, by removing the rightmost r-arrow from $\mathcal{I}$, we obtain an interpretation which is globally equisimilar to $\mathcal{I}$ but not a model of $\mathcal{T}$.

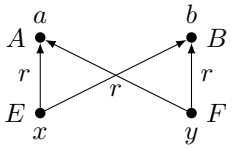


On the other hand, the $\mathcal{EL}_\bot$ TBox

$$\{\exists r.A \sqcap \exists r.B \sqsubseteq \exists r.G,\ \exists r.(G \sqcap A) \sqsubseteq E,\ \exists r.(G \sqcap B) \sqsubseteq F,\ A \sqcap B \sqsubseteq \bot\}$$

is easily seen to be an $\mathcal{ALC}$-conservative $\mathcal{EL}_\bot$ rewriting of $\mathcal{T}$. We now show that $\mathcal{T}$ is not model-conservatively $\mathcal{EL}_\bot$-rewritable. For suppose $\mathcal{T}$ has such a rewriting $\mathcal{T}'$ given in standard normal form (with inclusions of the form $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, $\exists r.B \sqsubseteq A$, or $A \sqsubseteq \exists r.B$ where $A_1, \ldots, A_n, A, B \in \mathsf{N_C} \cup \{\bot\}$). Consider the model

$\mathcal{I}$ of $\mathcal{T}$ depicted below, and let $\mathcal{I}'$ be a model of $\mathcal{T}'$ such that $\mathcal{I} =_{\mathsf{sig}(\mathcal{T})} \mathcal{I}'$.



Let $\mathcal{J}$ be the same as $\mathcal{I}'$ except that $x, y \in M^{\mathcal{J}}$ iff both $x \in M^{\mathcal{I}'}$ and $y \in M^{\mathcal{I}'}$, for every $M \in \mathsf{N_C}$. Since $x \notin E^{\mathcal{J}}$ and $y \notin F^{\mathcal{J}}$, $\mathcal{J}$ is not a model of $\mathcal{T}'$. Since the restriction of $\mathcal{I}'$ to $\{a, b\}$ is a model of $\mathcal{T}'$, and the restrictions of $\mathcal{I}'$ to $\{a, b, x\}$ and $\{a, b, y\}$ coincide, there is $(C \sqsubseteq D) \in \mathcal{T}'$ such that $x, y \in C^{\mathcal{J}}$ but $x, y \notin D^{\mathcal{J}}$. As $\mathcal{I}'$ is a model of $\mathcal{T}'$, which is in standard normal form, and by the definition of $\mathcal{J}$, D must be a concept name. Since clearly $x, y \in C^{\mathcal{I}'}$, we must also have $x, y \in D^{\mathcal{I}'}$, and so $x, y \in D^{\mathcal{J}}$, which is a contradiction.

The following modified version of $\mathcal{T}$

$$\mathcal{T}_m \;=\; \{\exists r.A \sqcap \exists r.B \sqcap \forall r.(A \sqcup B) \sqsubseteq \exists r.(A \sqcap E) \sqcup \exists r.(B \sqcap F),\; A \sqcap B \sqsubseteq \bot\}$$

is not equivalently $\mathcal{EL}_\bot$-rewritable, but has a model-conservative $\mathcal{EL}_\bot$-rewriting

$$\begin{aligned}\mathcal{T}'_m \;=\; \{&\exists r.A \sqcap \exists r.B \sqsubseteq \exists r.M,\; \exists r.(M \sqcap A) \sqsubseteq \exists r.(M \sqcap E),\\ &\exists r.(M \sqcap B) \sqsubseteq \exists r.(M \sqcap F),\; A \sqcap B \sqsubseteq \bot\}.\end{aligned}$$

The difference from the previous example is that if d is an instance of $\exists r.A \sqcap \exists r.B$, then we can place the 'marker' M onto an r-successor of d which is either in $A \sqcap E$ or in $B \sqcap F$, whereas in the previous example the decision on where to put the 'marker' G was not determined by the r-successors of d but by d itself.

We now prove that if there exists an $\mathcal{EL}_\bot$-rewriting of an $\mathcal{ALC}$ TBox $\mathcal{T}$, then there is one without any 'recursion' for the newly introduced symbols. Let $\Sigma = \mathsf{sig}(\mathcal{T})$. We say that an $\mathcal{EL}_\bot$ TBox $\mathcal{T}'$ is in *Σ-layered form of depth n* if there are mutually disjoint sets $\Gamma_0, \ldots, \Gamma_n$ of concept names such that $\Gamma_i \cap \Sigma = \emptyset$ $(0 \leq i \leq n)$ and the inclusions of $\mathcal{T}'$ take the following form, where $r \in \Sigma$:

level i atom inclusions: $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$, for $A_1, \ldots, A_n, B \in \Sigma \cup \Gamma_i \cup \{\bot\}$,
level i right-atom inclusions: $\exists r.A \sqsubseteq B$ for $A \in \Sigma \cup \Gamma_{i+1}$, $B \in \Sigma \cup \Gamma_i \cup \{\bot\}$,
level i left-atom inclusions: $A \sqsubseteq \exists r.B$, for $A \in \Sigma \cup \Gamma_i$, $B \in \Sigma \cup \Gamma_{i+1} \cup \{\bot\}$.

The *depth* of a concept C is the maximal number of nestings of existential restrictions in C. The *depth* of a TBox is the maximal depth of its concepts.

Lemma 3. *If an $\mathcal{ALC}$ TBox $\mathcal{T}$ of depth n is model- (or $\mathcal{ALC}$-) conservatively $\mathcal{EL}_\bot$-rewritable, then there exists a model- (respectively, $\mathcal{ALC}$-) conservative $\mathcal{EL}_\bot$-rewriting $\mathcal{T}'$ of $\mathcal{T}$ in $\mathsf{sig}(\mathcal{T})$-layered form of depth n.*

We use Lemma 3 to prove decidability of model-conservative $\mathcal{EL}_\bot$-rewritability. An $\mathcal{ALC}$ ABox $\mathcal{A}$ is a finite set of assertions of the form $C(a)$ and $r(a, b)$, where C is an $\mathcal{ALC}$ concept and a, b are *individual names*. The set of individual names

that occur in an ABox $\mathcal{A}$ is denoted by $\mathsf{ind}(\mathcal{A})$. When interpreting ABoxes, we adopt the *standard name assumption*: $a^{\mathcal{I}} = a$, for all $a \in \mathsf{ind}(\mathcal{A})$.

Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox of depth $n > 0$ (the case $n = 0$ is trivial). By $\mathsf{sub}^{n-1}(\mathcal{T})$ we denote the closure under single negation of the set of subconcepts of concepts in $\mathcal{T}$ of depth at most $n-1$. By $\Theta^{n-1}(\mathcal{T})$ we denote the set of maximal subsets $\boldsymbol{t}$ of $\mathsf{sub}^{n-1}(\mathcal{T})$ that are satisfiable in a model of $\mathcal{T}$. A *$\mathcal{T}$-ABox* is an ABox such that $\boldsymbol{t}_{\mathcal{A}}(a) = \{D \mid D(a) \in \mathcal{A}\} \in \Theta^{n-1}(\mathcal{T})$ for all $a \in \mathsf{ind}(\mathcal{A})$. Let $\mathcal{A}$ be a directed tree ABox of depth at most n (that is, all nodes in it are at distance $\leq n$ from the root). We say that $\mathcal{A}$ is *n-strongly satisfiable* w.r.t. $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ such that the $r^{\mathcal{I}}$-successors of $a^{\mathcal{I}}$, for every $a \in \mathsf{ind}(\mathcal{A})$ of depth $< n$ in $\mathcal{A}$, coincide with the r-successors of a in $\mathcal{A}$.

We now define inductively *$(\mathcal{T}, i)$-bisimilarity relations* $\sim_{i,\mathcal{T}}$ between pairs $(\mathcal{A}_1, a_1)$ and $(\mathcal{A}_2, a_2)$, where the $\mathcal{A}_i$ are $\mathcal{T}$-ABoxes and $a_i \in \mathsf{ind}(\mathcal{A}_i)$:

- $(\mathcal{A}_1, a_1) \sim_{0,\mathcal{T}} (\mathcal{A}_2, a_2)$ if $\boldsymbol{t}_{\mathcal{A}_1}(a_1) = \boldsymbol{t}_{\mathcal{A}_2}(a_2)$;
- $(\mathcal{A}_1, a_1) \sim_{i+1,\mathcal{T}} (\mathcal{A}_2, a_2)$ if $(\mathcal{A}_1, a_1) \sim_{0,\mathcal{T}} (\mathcal{A}_2, a_2)$ and, for every $r \in \mathsf{sig}(\mathcal{T})$, if $r(d_1, e_1) \in \mathcal{A}_1$ then there is $r(d_2, e_2) \in \mathcal{A}_2$ such that $(\mathcal{A}_1, e_1) \sim_{i,\mathcal{T}} (\mathcal{A}_2, e_2)$, and vice versa.

For every $i \geq 0$, one can determine a finite set AT_i of finite directed tree $\mathcal{T}$-ABoxes $\mathcal{A}$ with root $\rho_{\mathcal{A}}$ and of depth $\leq i$ such that:

- for every $\mathcal{I} \models \mathcal{T}$ and every $d \in \Delta^{\mathcal{I}}$, $(\mathcal{I}, d)$ is $(\mathcal{T}, i)$-bisimilar to exactly one $(\mathcal{A}, \rho_{\mathcal{A}}) \in \mathsf{AT}_i$;[4]
- every $\mathcal{A} \in \mathsf{AT}_i$ is strongly i-satisfiable w.r.t. $\mathcal{T}$.

We assume that all ABoxes in $\mathsf{AT}_0, \dots, \mathsf{AT}_n$ have mutually distinct roots. We define the *canonical ABox* $\mathcal{A}_{\mathcal{T}}$ with individuals $\{\rho_{\mathcal{A}} \mid \mathcal{A} \in \mathsf{AT}_i, i \leq n\}$ as follows:

- for $\mathcal{A}_i \in \mathsf{AT}_i$, $\mathcal{A}_{i+1} \in \mathsf{AT}_{i+1}$ and $r \in \mathsf{sig}(\mathcal{T})$, we have $r(\rho_{\mathcal{A}_{i+1}}, \rho_{\mathcal{A}_i}) \in \mathcal{A}_{\mathcal{T}}$ if there exists $r(\rho_{\mathcal{A}_{i+1}}, b) \in \mathcal{A}_{i+1}$ such that the subtree of $\mathcal{A}_{i+1}$ rooted at b is $(i, \mathcal{T})$-bisimilar to $\mathcal{A}_i$;
- for $\mathcal{A}_i \in \mathsf{AT}_i$ and $A \in \mathsf{sig}(\mathcal{T})$, we have $A(\rho_{\mathcal{A}_i}) \in \mathcal{A}_{\mathcal{T}}$ iff $A(\rho_{\mathcal{A}_i}) \in \mathcal{A}_i$.

Note that $\mathcal{A}_{\mathcal{T}}$ is acyclic (but not a directed tree ABox).

Lemma 4. *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox of depth n. An $\mathcal{EL}_\bot$ TBox $\mathcal{T}'$ in $\mathsf{sig}(\mathcal{T})$-layered form of depth n is a model-conservative $\mathcal{EL}_\bot$-rewriting of $\mathcal{T}$ iff*

- *$\mathcal{T}' \models \mathcal{T}$ and*
- *there exists $\mathcal{A}' =_{\mathsf{sig}(\mathcal{T})} \mathcal{A}_{\mathcal{T}}$ such that, for all $i = 0, \dots, n$, $\mathcal{A}'$ satisfies all level i inclusions in $\mathcal{T}'$ at all $\rho_{\mathcal{A}_i}$ with $\mathcal{A}_i \in \mathsf{AT}_{n-i}$.*

Theorem 8. *Model-conservative $\mathcal{EL}_\bot$-rewritability of $\mathcal{ALC}$ TBoxes is decidable.*

Proof. Given an $\mathcal{ALC}$ TBox $\mathcal{T}$, we first construct the canonical ABox $\mathcal{A}_{\mathcal{T}}$. If an $\mathcal{EL}_\bot$ TBox $\mathcal{T}'$ in Σ-layered form of depth n satisfies the conditions of Lemma 4, then there exists such a TBox with at most $2^{|\mathcal{A}_{\mathcal{T}}|}$ distinct fresh concept names. As the number of such $\mathcal{EL}_\bot$ TBoxes is finite, one can check for each of them whether the conditions of Lemma 4 are satisfied. ❏

[4] Here we identify $\mathcal{I}$ with the ABox with assertions $r(a, b)$, for $(a, b) \in r^{\mathcal{I}}$, and $D(a)$, for $D \in \mathsf{sub}^{n-1}(\mathcal{T})$ and $a \in D^{\mathcal{I}}$.

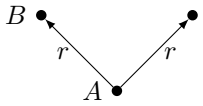
We now give necessary conditions for $\mathcal{ALC}$-conservative $\mathcal{EL}_\bot$-rewritability of $\mathcal{ALC}$ TBoxes. First, we still have the preservation under products:

Theorem 9. *Every $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable $\mathcal{ALC}$ TBox is preserved under products.*

Theorem 9 can be used to show that TBoxes such as $\{A \sqsubseteq B \sqcup E\}$ are not $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable. To separate equivalently rewritable TBoxes from $\mathcal{ALC}$-conservatively rewritable TBoxes, we generalize the construction of Example 4. In that case, we removed an r-arrow (d_0, d) from a tree-shaped model $\mathcal{I}$ of $\mathcal{T}$ and obtained a model that is globally equisimilar to the original model but not a model of $\mathcal{T}$. It turns out that $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable $\mathcal{ALC}$ TBoxes of depth 1 are preserved under the inverse of this operation. We say that $(\mathcal{I}, d)$ is $\subseteq_1$-*simulated by* $(\mathcal{J}, e)$ if (i) $d \in A^\mathcal{I}$ iff $e \in A^\mathcal{J}$, for all $A \in \mathsf{N_C}$; (ii) for all $r \in \mathsf{N_R}$, if $(e, e') \in r^\mathcal{J}$ then there exists d' with $(d, d') \in r^\mathcal{I}$ and, for all $A \in \mathsf{N_C}$, if $e' \in A^\mathcal{J}$ then $d' \in A^\mathcal{I}$; (iii) for all $r \in \mathsf{N_R}$, if $(d, d') \in r^\mathcal{I}$ then there exists e' with $(e, e') \in r^\mathcal{J}$ and, for all $A \in \mathsf{N_C}$, we have $d' \in A^\mathcal{I}$ iff $e' \in A^\mathcal{J}$. Say that $\mathcal{I}$ is *globally* $\subseteq_1$-*simulated* by $\mathcal{J}$ if, for every $e \in \Delta^\mathcal{J}$, there exists $d \in \Delta^\mathcal{I}$ such that $(\mathcal{I}, d)$ is $\subseteq_1$-simulated by $(\mathcal{J}, e)$. An $\mathcal{ALC}$ TBox is *preserved under* $\subseteq_1$-*simulations* if every interpretation that globally $\subseteq_1$-simulates a model of $\mathcal{T}$ is a model of $\mathcal{T}$.

Theorem 10. *Every $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable $\mathcal{ALC}$ TBox of depth 1 is preserved under global $\subseteq_1$-simulations.*

This result can be used to show, for example, that $\mathcal{T} = \{A \sqsubseteq \forall r.B\}$ is not $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable. For the interpretation below is *not* a model



of $\mathcal{T}$, but by removing from it the rightmost r-arrow, we obtain an interpretation which is globally $\subseteq_1$-simulated by $\mathcal{J}$ and is a model of $\mathcal{T}$. It remains open whether preservation under products and global $\subseteq_1$-simulations is sufficient for an $\mathcal{ALC}$ TBox of depth 1 to be $\mathcal{ALC}$-conservatively $\mathcal{EL}_\bot$-rewritable.

5 Conclusion

Conservative rewritings of ontologies provide more flexibility than equivalent rewritings and are more natural in practice. However, they are also technically much more challenging to analyse. For future work, we are particularly interested in better understanding conservative rewritings to $\mathcal{EL}$ and related logics. For example, can we find transparent model-theoretic characterizations and explicit axiomatizations of the rewritten TBoxes? The results in Section 4 should provide a good starting point. Another challenging problem could be to investigate rewritability to OWL 2 QL—essentially *DL-Lite*$_{core}$ extended with role inclusions—which preserves answers to conjunctive queries over all possible ABoxes. (Recall [6] that conjunctive query inseparability for OWL 2 QL TBoxes is ExpTime-complete.)

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research 36, 1–69 (2009)
2. Baader, F.: A formal definition for the expressive power of terminological knowledge representation languages. Journal of Logic and Computation 6(1), 33–54 (1996)
3. Baader, F.: The description logic handbook: Theory, implementation, and applications. Cambridge University Press, Cambridge (2007)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: Proceedings of IJCAI. pp. 364–369 (2005)
5. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Transactions of Database Systems 39(4), 33 (2014)
6. Botoeva, E., Kontchakov, R., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Query inseparability for description logic knowledge bases. In: Proceedings of KR (2014)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)
8. Carral, D., Feier, C., Grau, B.C., Hitzler, P., Horrocks, I.: *EL*-ifying ontologies. In: Proceedings of IJCAR. pp. 464–479 (2014)
9. Carral, D., Feier, C., Romero, A.A., Grau, B.C., Hitzler, P., Horrocks, I.: Is your ontology as hard as you think? Rewriting ontologies into simpler DLs. In: Proceedings of DL. pp. 128–140 (2014)
10. De Giacomo, G.: Decidability of Class-Based Knowledge Representation Formalisms. Ph.D. thesis, Università di Roma (1995)
11. Ding, Y., Haarslev, V., Wu, J.: A new mapping from ALCI to ALC. In: Proceedings of DL (2007)
12. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proceedings of KR. pp. 187–197 (2006)
13. Kaminski, M., Grau, B.C.: Sufficient conditions for first-order and datalog rewritability in $\mathcal{ELU}$. In: Proceedings of DL. pp. 271–293 (2013)
14. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Proceedings of IJCAI. pp. 2040–2045 (2009)
15. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularisation. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, pp. 25–66 (2009)
16. Konev, B., Lutz, C., Walther, D., Wolter, F.: Model-theoretic inseparability and modularity of description logic ontologies. Artificial Intelligence 203, 66–103 (2013)
17. Lutz, C., Piro, R., Wolter, F.: Description logic TBoxes: Model-theoretic characterizations and rewritability. In: Proceedings of IJCAI (2011)
18. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. Journal of Symbolic Computation pp. 194–228 (2010)
19. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of IJCAI. pp. 989–995 (2011)

Exact Learning Description Logic Ontologies from Data Retrieval Examples

Boris Konev, Ana Ozaki, and Frank Wolter

Department of Computer Science, University of Liverpool, UK

Abstract. We investigate the complexity of learning description logic ontologies in Angluin et al.'s framework of exact learning via queries posed to an oracle. We consider membership queries of the form "is individual a a certain answer to a data retrieval query q in a given ABox and the unkown target TBox?" and equivalence queries of the form "is a given TBox equivalent to the unknown target TBox?". We show that (i) DL-Lite TBoxes with role inclusions and $\mathcal{ELI}$ concept expressions on the right-hand side of inclusions and (ii) $\mathcal{EL}$ TBoxes without complex concept expressions on the right-hand side of inclusions can be learned in polynomial time. Both results are proved by a non-trivial reduction to learning from subsumption examples. We also show that arbitrary $\mathcal{EL}$ TBoxes cannot be learned in polynomial time.

1 Introduction

Building an ontology is prone to errors, time consuming, and costly. The research communities has addressed this problem in many different ways, for example, by supplying tool support for editing ontologies [15, 4, 9], developing reasoning support for debugging ontologies [18], supporting modular ontology design [17], and by investigating automated ontology generation from data or text [8, 6, 5, 14]. One major problem when building an ontology is the fact that domain experts are rarely ontology engineering experts and that, conversely, ontology engineers are typically not familiar with the domain of the ontology. An ontology building project therefore often relies on the successful communication between an ontology engineer (familiar with the semantics of ontology languages) and a domain expert (familiar with the domain of interest). In this paper, we consider a simple model of this communication process and analyse, within this model, the computational complexity of reaching a correct domain ontology. We assume that

- the domain expert knows the domain ontology and its vocabulary without being able to formalize or communicate this ontology;
- the domain expert is able to communicate the vocabulary of the ontology and shares it with the the ontology engineer. Thus, the domain expert and ontology engineer have a common understanding of the vocabulary of the ontology. The ontology engineer knows nothing else about the domain.
- the ontology engineer can pose queries to the domain expert which the domain expert answers truthfully. Assuming that the domain expert can interpret data in her area of expertise, the main queries posed by the ontology engineer are based on instance retrieval examples:

- assume a data instance $\mathcal{A}$ and a query $q(x)$ are given. Is the individual a a certain answer to query $q(x)$ in $\mathcal{A}$ and the ontology $\mathcal{O}$?

In addition, we require a way for the ontology engineer to find out whether she has reconstructed the target ontology already and, if this is not the case, to request an example illustrating the incompleteness of the reconstruction. We abstract from defining a communication protocol for this, but assume for simplicity that the following query can be posed by the ontology engineer:

- Is this ontology $\mathcal{H}$ complete? If not, return a data instance $\mathcal{A}$, a query $q(x)$, and an individual a such that a is a certain answer to $q(x)$ in $\mathcal{A}$ and the ontology $\mathcal{O}$ and it is not a certain answer to $q(x)$ in $\mathcal{A}$ and the ontology $\mathcal{H}$.

Given this model, our question is whether the ontology engineer can learn the target ontology $\mathcal{O}$ and which computational resources are required for this depending on the ontology language in which the ontology $\mathcal{O}$ and the hypothesis ontologies $\mathcal{H}$ are formulated. Our model obviously abstracts from a number of fundamental problems in building ontologies and communicating about them. In particular, it makes the assumption that the domain expert knows the domain ontology and its vocabulary (without being able to formalize it) despite the fact that finding an appropriate vocabulary for a domain of interest is a major problem in ontology design [8]. We make this assumption here in order to isolate the problem of communication about the logical relationships between known vocabulary items and its dependence on the ontology language within which the relationships can be formulated.

The model described above is an instance of Angluin et al.'s framework of exact learning via queries to an oracle [1]. The queries using instance retrieval examples can be regarded as membership queries posed by a learner to an oracle and the completeness query based on a hypothesis $\mathcal{H}$ can be regarded as an equivalence query by the learner to the oracle. Formulated in Angluin's terms we are thus interested in whether there exists a deterministic learning algorithm that poses membership and equivalence queries of the above form to an oracle and that learns an arbitrary ontology over a given ontology language in polynomial time. We consider polynomial learnability in three distinct DLs: we show that DL-Lite ontologies with role inclusions and arbitrary $\mathcal{ELI}$ concepts on the right-hand side of concept inclusions can be learned in polynomial time if database queries in instance retrieval examples are $\mathcal{ELI}$ instance queries (or, equivalently, acyclic conjunctive queries). We call this DL DL-Lite$^{\exists}_{\mathcal{R}}$ and note that it is the core of the web ontology language profile OWL2 QL. We also note that *without* complex $\mathcal{ELI}$ concepts on the right-hand side of concept inclusions, polynomial learnability would be trivial as only finitely many non-equivalent such TBoxes exist over a given vocabulary of concept and role names. The second DL we consider is $\mathcal{EL}$ which is the logic underpinning the web ontology language profile OWL2 EL. We show that $\mathcal{EL}$ TBoxes cannot be learned in polynomial time using the protocol above if the database queries in instance retrieval examples are $\mathcal{EL}$ instance queries. We then consider the fragment $\mathcal{EL}_{\mathsf{lhs}}$ of $\mathcal{EL}$ without complex concepts on the right-hand side of concept inclusions and prove that it can be learned in polynomial time using the above protocol with instance retrieval examples. The proofs of the positive learning results are by reduction to polynomial time learnability results for DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ for the case in which *concept subsumptions* rather than instance retrieval examples are used in the communication between the learner and the

oracle [12]. Our move from concept subsumptions to data retrieval examples is motivated by the observation that domain experts are often more familiar with querying data in their domain than with the logical notion of subsumption between complex concepts. Detailed proofs are provided at `http://csc.liv.ac.uk/~frank/publ/publ.html`.

2 Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of *concept* and *role* names, respectively. The dialect DL-Lite$^{\exists}_{\mathcal{R}}$ of DL-Lite is defined as follows [7]. A *role* is a role name or an inverse role r^- with $r \in \mathsf{N_R}$. A *role inclusion (RI)* is of the form $r \sqsubseteq s$, where r and s are roles. A *basic concept* is either a concept name or of the form $\exists r.\top$, with r a role. A DL-Lite$^{\exists}_{\mathcal{R}}$ concept inclusion (CI) is of the form $B \sqsubseteq C$, where B is a basic concept expression and C is an $\mathcal{ELI}$ concept expression, that is, C is formed according to the rule $C, D := A \mid \top \mid C \sqcap D \mid \exists r.C \mid \exists r^-.C$ where A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. A DL-Lite$^{\exists}_{\mathcal{R}}$ TBox is a finite set of DL-Lite$^{\exists}_{\mathcal{R}}$ CIs and RIs. As usual, an $\mathcal{EL}$ *concept expression* is an $\mathcal{ELI}$ concept expression that does not use inverse roles, an $\mathcal{EL}$ *concept inclusion* has the form $C \sqsubseteq D$ with C and D $\mathcal{EL}$ concept expressions, and a *(general) $\mathcal{EL}$ TBox* is a finite set of $\mathcal{EL}$ concept inclusions [2]. We also consider the restriction $\mathcal{EL}_{\mathsf{lhs}}$ of general $\mathcal{EL}$ TBoxes where only concept names are allowed on the right-hand side of concept inclusions. The *size* of a concept expression C, denoted with $|C|$, is the length of the string that represents it, where concept names and role names are considered to be of length one. A TBox *signature* is the set of concept and role names occurring in the TBox. The *size* of a TBox $\mathcal{T}$, denoted with $|\mathcal{T}|$, is $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

Let $\mathsf{N_I}$ be a countably infinite set of *individual names*. An ABox $\mathcal{A}$ is a finite non-empty set containing *concept name assertions* $A(a)$ and *role assertions* $r(a, b)$, where a, b are individuals in $\mathsf{N_I}$, A is a concept name and r is a role. $\mathsf{Ind}(\mathcal{A})$ denotes the set of individuals that occur in $\mathcal{A}$. $\mathcal{A}$ is a *singleton* ABox if it contains only one ABox assertion. Assertions of the form $C(a)$ and $r(a, b)$, where $a, b \in \mathsf{N_I}$, C an $\mathcal{ELI}$ concept expression, and $r \in \mathsf{N_R}$, are called *instance assertions*. Note that instance assertions of the form $C(a)$ with C not a concept name nor $C = \top$ do not occur in ABoxes. The semantics of description logic is defined as usual [3]. We write $\mathcal{I} \models \alpha$ to say that an inclusion or assertion α is true in $\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model* of a KB $(\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. $(\mathcal{T}, \mathcal{A}) \models \alpha$ means that $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$.

A *learning framework* $\mathfrak{F}$ is a triple $(X, \mathcal{L}, \mu)$, where X is a set of *examples* (also called domain or instance space), $\mathcal{L}$ is a set of *learning concepts*[1] and μ is a mapping from $\mathcal{L}$ to 2^X. The *subsumption* learning framework $\mathfrak{F}_{\mathcal{S}}$, studied in [12], is defined as $(X_{\mathcal{S}}, \mathcal{L}, \mu_{\mathcal{S}})$, where $\mathcal{L}$ is the set of all TBoxes that are formulated in a given DL; $X_{\mathcal{S}}$ is the set of *subsumption examples* of the form $C \sqsubseteq D$, where C, D are concept expressions of the DL under consideration; and $\mu_{\mathcal{S}}(\mathcal{T})$ is defined as $\{C \sqsubseteq D \in X_{\mathcal{S}} \mid \mathcal{T} \models C \sqsubseteq D\}$, for every $\mathcal{T} \in \mathcal{L}$. It should be clear that $\mu_{\mathcal{S}}(\mathcal{T}) = \mu_{\mathcal{S}}(\mathcal{T}')$ if, and only if, the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ entail the same set of inclusions, that is, they are logically equivalent.

[1] In the learning literature (e.g., [1]), the term 'learning concept' is often defined as a set of examples. We do not distinguish between learning concepts and their representations and only consider representable learning concepts to emphasize on the task of identifying a TBox that is logically equivalent to the target TBox.

We study the *data retrieval* learning framework $\mathfrak{F}_{\mathcal{D}}$ defined as $(X_{\mathcal{D}}, \mathcal{L}, \mu_{\mathcal{D}})$, where $\mathcal{L}$ is same as in $\mathfrak{F}_{\mathcal{S}}$; X is the set of *data retrieval examples* of the form $(\mathcal{A}, D(a))$, where $\mathcal{A}$ is an ABox, $D(a)$ is a concept assertion of the DL under consideration, and $a \in \mathsf{Ind}(\mathcal{A})$; and $\mu(\mathcal{T}) = \{(\mathcal{A}, D(a)) \in X_{\mathcal{D}} \mid (\mathcal{T}, \mathcal{A}) \models D(a)\}$. As in the case of learning from subsumptions, $\mu_{\mathcal{S}}(\mathcal{T}) = \mu_{\mathcal{S}}(\mathcal{T}')$ if, and only if, the TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are logically equivalent.

Given a learning framework $\mathfrak{F} = (X, \mathcal{L}, \mu)$, we are interested in the exact identification of a *target* learning concept $l \in \mathcal{L}$ by posing queries to oracles. Let $\mathsf{MEM}_{l,X}$ be the oracle that takes as input some $x \in X$ and returns 'yes' if $x \in \mu(l)$ and 'no' otherwise. We say that x is a *positive example* for l if $x \in \mu(l)$ and a *negative example* for l if $x \notin \mu(l)$. Then a *membership query* is a call to the oracle $\mathsf{MEM}_{l,X}$. Similarly, for every $l \in \mathcal{L}$, we denote by $\mathsf{EQ}_{l,X}$ the oracle that takes as input a *hypothesis* learning concept $h \in \mathcal{L}$ and returns 'yes', if $\mu(h) = \mu(l)$, or a *counterexample* $x \in \mu(h) \oplus \mu(l)$ otherwise, where $\oplus$ denotes the symmetric set difference. An *equivalence query* is a call to the oracle $\mathsf{EQ}_{l,X}$.

We say that a learning framework $(X, \mathcal{L}, \mu)$ is *exact learnable* if there is an algorithm A such that for any target $l \in \mathcal{L}$ the algorithm A always halts and outputs $l' \in \mathcal{L}$ such that $\mu(l) = \mu(l')$ using membership and equivalence queries answered by the oracles $\mathsf{MEM}_{l,X}$ and $\mathsf{EQ}_{l,X}$, respectively. A learning framework $(X, \mathcal{L}, \mu)$ is *polynomially* exact learnable if it is exact learnable by an algorithm A such that at every step[2] of computation the time used by A up to that step is bounded by a polynomial $p(|l|, |x|)$, where l is the target and $x \in X$ is the largest counterexample seen so far[3]. As argued in the introduction, for learning subsumption and data retrieval learning frameworks we additionally assume that the signature of the target TBox is always known to the learner.

An important class of learning algorithms—in particular, all algorithms presented in [12, 10, 16] fit in this class—always make equivalence queries with hypotheses h which are polynomial in the size of l and such that $\mu(h) \subseteq \mu(l)$, so that counterexamples returned by the $\mathsf{EQ}_{l,X}$ oracles are always positive. We say that such algorithms use *positive bounded equivalence queries*.

3 Polynomial Time Learnability

In this section we prove polynomial time exact learnability of the DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ data retrieval learning frameworks. These frameworks are instances of the general definition given above, where the concept expression D in a data retrieval example $(\mathcal{A}, D(a))$ is an $\mathcal{ELI}$ concept expression in the DL-Lite$^{\exists}_{\mathcal{R}}$ framework and an $\mathcal{EL}$ concept expression in the $\mathcal{EL}_{\mathsf{lhs}}$ framework, respectively.

The proof is by reduction to learning from subsumptions. We illustrate its idea for $\mathcal{EL}_{\mathsf{lhs}}$. To learn a TBox from data retrieval examples we run a learning from subsumptions algorithm as a 'black box'. Every time the learning from subsumptions algorithm makes a membership or an equivalence query we rewrite the query into the data setting and pass it on to the data retrieval oracle. The oracle's answer, rewritten back to the subsumption

[2] We count each call to an oracle as one step of computation.

[3] We assume some natural notion of a length of an example x and a learning concept l, denoted $|x|$ and $|l|$, respectively.

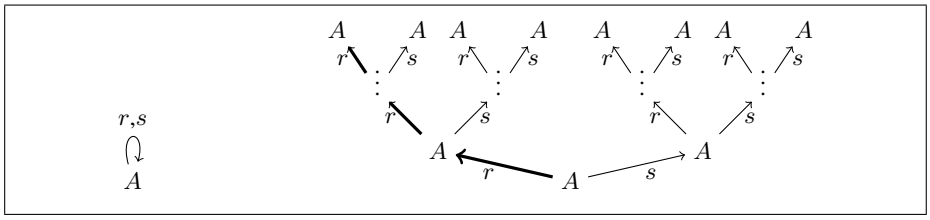


Fig. 1: An ABox $\mathcal{A} = \{r(a,a), s(a,a), A(a)\}$ and its unravelling up to level n.

setting, is given to the learning from subsumptions algorithm. When the learning from subsumptions algorithm terminates we return the learnt TBox. This reduction is made possible by the close relationship between data retrieval and subsumption examples. For every TBox $\mathcal{T}$ and inclusions $C \sqsubseteq D$, one can interpret a concept expression C as a labelled tree and encode this tree as an ABox $\mathcal{A}_C$ with root ρ_C such that $\mathcal{T} \models C \sqsubseteq D$ iff $(\mathcal{T}, \mathcal{A}_C) \models D(\rho_C)$.

Then, membership queries in the subsumption setting can be answered with the help of a data retrieval oracle due to the relation between subsumptions and instance queries described above. An inclusion $C \sqsubseteq D$ is a (positive) subsumption example for some target TBox $\mathcal{T}$ if, and only if, $(\mathcal{A}_C, D(\rho_C))$ is a (positive) data retrieval example for the same target $\mathcal{T}$. To handle equivalence queries, we need to be able to rewrite data retrieval counterexamples returned by the data retrieval oracle into the subsumption setting. For every TBox $\mathcal{T}$ and data retrieval query $(\mathcal{A}, D(a))$ one can construct a concept expression $C_{\mathcal{A}}$ such that $(\mathcal{T}, \mathcal{A}) \models D(a)$ iff $\mathcal{T} \models C_{\mathcal{A}} \sqsubseteq D$. Such a concept expression $C_{\mathcal{A}}$ can be obtained by unravelling $\mathcal{A}$ into a tree-shaped ABox and representing it as a concept expression. This unravelling, however, can increase the ABox size exponentially. Thus, to obtain a polynomial bound on the running time of the learning process, $C_{\mathcal{A}} \sqsubseteq D$ cannot be simply returned as an answer to a subsumption equivalence query. For example, for a target TBox $\mathcal{T} = \{\exists r^n.A \sqsubseteq B\}$ and a hypothesis $\mathcal{H} = \emptyset$ the data retrieval query $(\mathcal{A}, B(a))$, where $\mathcal{A} = \{r(a,a), s(a,a), A(a)\}$, is a positive counterexample. The tree-shaped unravelling of $\mathcal{A}$ up to level n is a full binary tree of depth n, as shown in Fig. 1. On the other hand, the non-equivalence of $\mathcal{T}$ and $\mathcal{H}$ can already be witnessed by $(\mathcal{A}', B(a))$, where $\mathcal{A}' = \{r(a,a), A(a)\}$. The unravelling of $\mathcal{A}'$ up to level n produces a linear size ABox $\{r(a,a_2), r(a_2,a_3), \ldots, r(a_{n-1},a_n), A(a), A(a_2), \ldots, A(a_n)\}$, corresponding to the left-most path in Fig. 1, which, in turn, is linear-size w.r.t. the target inclusion $\exists r^n.A \sqsubseteq B$. Notice that $\mathcal{A}'$ is obtained from $\mathcal{A}$ by removing the $s(a,a)$ edge and checking, using membership queries, whether $(\mathcal{T}, \mathcal{A}') \models q$ still holds. In other words, one might need to ask further membership queries in order to rewrite answers to data retrieval equivalence queries given by the data retrieval oracle into the subsumption setting.

We address the need of rewriting counterexamples by introducing an abstract notion of reduction between different exact learning frameworks. To simplify notation, we assume that both learning frameworks use the same set of learning concepts $\mathcal{L}$ and only consider positive bounded equivalence queries. This definition of reduction can be easily extended to arbitrary learning frameworks and arbitrary queries.

We say that a learning framework $\mathfrak{F} = (X, \mathcal{L}, \mu)$ *polynomially reduces* to $\mathfrak{F}' = (X', \mathcal{L}, \mu')$ if for some polynomials $p_1(\cdot)$, $p_2(\cdot)$ and $p_3(\cdot,\cdot)$ there exist a function $f : X' \to X$ and a partial function $g : \mathcal{L} \times \mathcal{L} \times X \to X'$, defined for every (l, h, x) such that $|h| = p_1(|l|)$, $\mu(h) \subseteq \mu(l)$ and $x \in X$, for which the following conditions hold.

- For all $x' \in X'$ we have $x' \in \mu'(l)$ if, and only if, $f(x') \in \mu(l)$;
- For all $x \in X$ we have $x \in \mu(l) \setminus \mu(h)$ if, and only if, $g(l, h, x) \in \mu'(l) \setminus \mu'(h)$;
- $f(x')$ is computable in time $p_2(|x'|)$;
- $g(l, h, x)$ is computable in time $p_3(|l|, |x|)$ and l can only be accessed by calls to the membership oracle $\mathsf{MEM}_{l,X}$.

As in the case of learning algorithms, we consider every call to the oracle as one step of computation. Notice also that even though g takes h as input, the polynomial time bound on computing $g(l, h, x)$ does not depend on the size of h as g is only defined for h polynomial in the size of l.

Theorem 1. *Let $(X, \mathcal{L}, \mu)$ and $(X', \mathcal{L}, \mu')$ be learning frameworks. If there exists a polynomial reduction from $(X, \mathcal{L}, \mu)$ to $(X', \mathcal{L}, \mu')$ and a polynomial learning algorithm for $(X', \mathcal{L}, \mu')$ that uses membership queries and positive bounded equivalence queries then $(X, \mathcal{L}, \mu)$ is polynomially exact learnable.*

We use Theorem 1 to prove that DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes can be learned in polynomial time from data retrieval examples. We employ the following result:

Theorem 2 ([12]). *The DL-Lite$^{\exists}_{\mathcal{R}}$ and $\mathcal{EL}_{\mathsf{lhs}}$ subsumption learning frameworks are polynomial time exact learnable with membership and positive bounded equivalence queries.*

As the existence of f is guaranteed by the following lemma, in what follows we prove the existence of g and establish the corresponding time bounds.

Lemma 1. *Let $L \in \{$DL-Lite$^{\exists}_{\mathcal{R}}, \mathcal{EL}_{\mathsf{lhs}}\}$ and let $C \sqsubseteq D$ be an L concept inclusion. Then $(\mathcal{T}, \mathcal{A}_C) \models D(\rho_C)$ if, and only if, $\mathcal{T} \models C \sqsubseteq D$.*

Polynomial Reduction for DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes We show for any target $\mathcal{T}$ and hypothesis $\mathcal{H}$ polynomial in the size of $\mathcal{T}$ that Algorithm 1 transforms every positive counterexample in polynomial time to a positive counterexample with a singleton ABox (i.e., of the form $\{A(a)\}$ or $\{r(a, b)\}$). Using the equivalences $(\mathcal{T}, \{A(a)\}) \models C(a)$ iff $\mathcal{T} \models A \sqsubseteq C$ and $(\mathcal{T}, \{r(a, b)\}) \models C(a)$ iff $\mathcal{T} \models \exists r.\top \sqsubseteq C$, we then obtain a positive subsumption counterexample, so $g(l, h, x)$ is computable in polynomial time.

Given a positive data retrieval counterexample $(\mathcal{A}, C(a))$, Algorithm 1 exhaustively applies the *role saturation* and *parent-child merging* rules introduced in [12]. We say that an instance assertion $C(a)$ is *role saturated* for $(\mathcal{T}, \mathcal{A})$ if $(\mathcal{T}, \mathcal{A}) \not\models C'(a)$ whenever C' is the result of replacing a role r by some role $s \in \mathsf{N_R} \cap \Sigma_{\mathcal{T}}$ with $\mathcal{T} \not\models r \sqsubseteq s$ and $\mathcal{T} \models s \sqsubseteq r$, where $\Sigma_{\mathcal{T}}$ is the signature of the target TBox $\mathcal{T}$ known to the learner. To define parent/child merging, we identify each $\mathcal{ELI}$ concept C with a finite tree T_C whose nodes are labeled with concept names and edges are labeled with roles in the standard way. For example, if $C = \exists t.(A \sqcap \exists r.\exists r^-.\exists r.B) \sqcap \exists s.\top$ then Fig. 2a illustrates

Algorithm 1 Reducing the positive counterexample

1: Let $C(a)$ be an instance assertion such that $(\mathcal{H}, \mathcal{A}) \not\models C(a)$ and $(\mathcal{T}, \mathcal{A}) \models C(a)$
2: **function** REDUCECOUNTEREXAMPLE($\mathcal{A}$, $C(a)$)
3: Find a role saturated and parent/child merged $C(a)$ (membership queries)
4: **if** $C = C_0 \sqcap ... \sqcap C_n$ **then**
5: Find C_i, $0 \leq i \leq n$, such that $(\mathcal{H}, \mathcal{A}) \not\models C_i(a)$
6: $C := C_i$
7: **if** $C = \exists r.C'$ and there is $r(a, b) \in \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models C'(b)$ **then**
8: REDUCECOUNTEREXAMPLE($\mathcal{A}$, $C'(b)$)
9: **else**
10: Find a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$ but
11: $(\mathcal{H}, \mathcal{A}') \not\models C(a)$ (membership queries)
12: **return** ($\mathcal{A}'$,$C(a)$)

T_C. Now, we say that an instance assertion $C(a)$ is *parent/child merged* for $\mathcal{T}$ and $\mathcal{A}$ if for nodes n_1, n_2, n_3 in T_C such that n_2 is an r-successor of n_1, n_3 is an s-successor of n_2 and $\mathcal{T} \models r^- \equiv s$ we have $(\mathcal{T}, \mathcal{A}) \not\models C'(a)$ if C' is the concept that results from identifying n_1 and n_3. For instance, the concept in Fig. 2c is the result of identifying the leaf labeled with B in Fig. 2b with the parent of its parent.

We present a run of Algorithm 1 for $\mathcal{T} = \{A \sqsubseteq \exists s.B, s \sqsubseteq r\}$ and $\mathcal{H} = \{s \sqsubseteq r\}$. Assume the oracle gives as counterexample $(\mathcal{A}, C(a))$, where $\mathcal{A} = \{t(a, b), A(b), s(a, c)\}$ and $C(a) = \exists t.(A \sqcap \exists r.\exists r^-.\exists r.B) \sqcap \exists s.\top(a)$ (Fig. 2a). Role saturation produces $C(a) = \exists t.(A \sqcap \exists s.\exists s^-.\exists s.B) \sqcap \exists s.\top(a)$ (Fig. 2b). Then, applying parent/child merging twice we obtain $C(a) = \exists t.(A \sqcap \exists s.B) \sqcap \exists s.\top(a)$ (Fig. 2c and 2d).

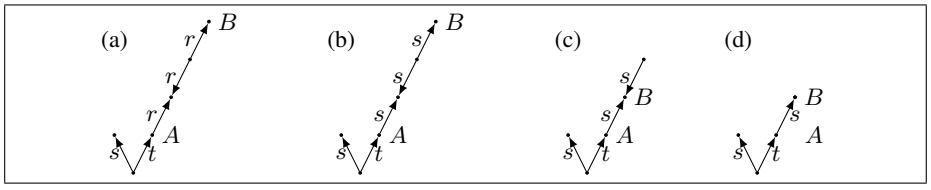


Fig. 2: Concept C being role saturated and parent/child merged.

Since $(\mathcal{H}, \mathcal{A}) \not\models \exists t.(A \sqcap \exists s.B)(a)$, after Lines 4-6, Algorithm 1 updates C by choosing the conjunct $\exists t.(A \sqcap \exists s.B)$. As C is of the form $\exists t.C'$ and there is $t(a, b) \in \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models C'(b)$, the algorithm recursively calls the function "ReduceCounterExample" with $A \sqcap \exists s.B(b)$. Now, since $(\mathcal{H}, \mathcal{A}) \not\models \exists s.B(b)$, after Lines 4-6, C is updated to $\exists s.B$. Finally, C is of the form $\exists t.C'$ and there is no $t(b, c) \in \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models C'(c)$. So the algorithm proceeds to Lines 11-12, where it chooses $A(b) \in \mathcal{A}$. Since $(\mathcal{T}, \{A(b)\}) \models \exists s.B(b)$ and $(\mathcal{H}, \{A(b)\}) \not\models \exists s.B(b)$ we have that $\mathcal{T} \models A \sqsubseteq \exists s.B$ and $\mathcal{H} \not\models A \sqsubseteq \exists s.B$.

Lemma 2. *Let $(\mathcal{A}, C(a))$ be a positive counterexample. Then the following holds:*

1. *if C is a basic concept then there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$;*

Algorithm 2 Minimizing an ABox $\mathcal{A}$

1: Let $\mathcal{A}$ be an ABox such that $(\mathcal{T}, \mathcal{A}) \models A(a)$ but $(\mathcal{H}, \mathcal{A}) \not\models A(a)$, for $A \in \mathsf{N_C}$, $a \in \mathsf{Ind}(\mathcal{A})$.
2: **function** MINIMIZEABOX($\mathcal{A}$)
3: Concept saturate $\mathcal{A}$ with $\mathcal{H}$
4: **for** every $A \in \mathsf{N_C} \cap \Sigma_\mathcal{T}$ and $a \in \mathsf{Ind}(\mathcal{A})$ such that
5: $(\mathcal{T}, \mathcal{A}) \models A(a)$ and $(\mathcal{H}, \mathcal{A}) \not\models A(a)$ **do**
6: Domain Minimize $\mathcal{A}$ with $A(a)$
7: Role Minimize $\mathcal{A}$ with $A(a)$
8: **return** ($\mathcal{A}$)

2. *if C is of the form $\exists r.C'$ (or $\exists r^-.C'$) and C is role saturated and parent/child merged then either there is $r(a, b) \in \mathcal{A}$ (or $r(b, a) \in \mathcal{A}$) such that $(\mathcal{T}, \mathcal{A}) \models C'(b)$ or there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$.*

Lemma 3. *For any target DL-Lite$^\exists_\mathcal{R}$ TBox $\mathcal{T}$ and hypothesis DL-Lite$^\exists_\mathcal{R}$ TBox $\mathcal{H}$ given a positive data retrieval counterexample $(\mathcal{A}, C(a))$, Algorithm 1 computes in time polynomial in $|\mathcal{T}|$, $|\mathcal{H}|$, $|\mathcal{A}|$ and $|C|$ a counterexample $C'(b)$ such that $(\mathcal{T}, \mathcal{A}') \models C'(b)$, where $\mathcal{A}' \subseteq \mathcal{A}$ is a singleton ABox.*

Proof. (Sketch) Let $(\mathcal{A}, C(a))$ be the input of "ReduceCounterExample". The number of membership queries in Line 3 is polynomial in $|C|$ and $|\mathcal{T}|$. If C has more than one conjunct then it is updated in Lines 4-6, so C becomes either (1) a basic concept or (2) of the form $\exists r.C'$ (or $\exists r^-.C'$). By Lemma 2 in case (1) there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$, computed by Line 11 of Algorithm 1. In case (2) either there is a singleton $\mathcal{A}' \subseteq \mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}') \models C(a)$, computed by Line 11 of Algorithm 1, or we obtain a counterexample with a refined C. Since the size of the refined counterexample is strictly smaller after every recursive call of "ReduceCounterExample", the total number of calls is bounded by $|C|$. ❑

Using Theorem 2 and Theorem 1 we obtain:

Theorem 3. *The DL-Lite$^\exists_\mathcal{R}$ data retrieval framework is polynomially exact learnable.*

Polynomial Reduction for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes In this section we give a polynomial algorithm computing g for $\mathcal{EL}_{\mathsf{lhs}}$. First we note that the concept assertion in data retrieval counterexamples $(\mathcal{A}, D(a))$ can always be made atomic. Let $\Sigma_\mathcal{T}$ be the signature of the target TBox $\mathcal{T}$.

Lemma 4. *If $(\mathcal{A}, D(a))$ is a positive counterexample then by posing polynomially many membership queries one can find a concept name $A \in \Sigma_\mathcal{T}$ and an individual $b \in \mathsf{Ind}(\mathcal{A})$ such that $(\mathcal{A}, A(b))$ is also a counterexample.*

Thus it suffices to show that given a positive counterexample $(\mathcal{A}, D(a))$ with $D \in \mathsf{N_C}$, one can compute an $\mathcal{EL}$ concept expression C bounded in size by $|\mathcal{T}|$ such that $(\mathcal{T}, \{C(b)\}) \models A(b)$ and $(\mathcal{H}, \{C(b)\}) \not\models A(b)$, where $A \in \mathsf{N_C}$. As $(\mathcal{T}, \{C(b)\}) \models A(b)$ if and only if $\mathcal{T} \models C \sqsubseteq A$, we obtain a positive subsumption counterexample. Our algorithm for computing g is based on two operations: minimization, computed by

Algorithm 3 Computing a tree shaped ABox

function FINDTREE($\mathcal{A}$)
  MINIMIZEABOX($\mathcal{A}$)
  while there is a cycle c in $\mathcal{A}$ **do**
    Unfold $a \in \mathsf{Ind}(\mathcal{A})$ in cycle c
    MINIMIZEABOX($\mathcal{A}$)
  Let C be the concept expression corresponding to $\mathcal{A}$ with counterexample $A(a)$.
  return $(C(a),A(a))$

Algorithm 2, and unfolding. Algorithm 2 *minimizes* a given ABox with the following rules.

(Concept saturate $\mathcal{A}$ with $\mathcal{H}$) If $A(a) \notin \mathcal{A}$ and $(\mathcal{H}, \mathcal{A}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A} \cup \{A(a)\}$, where $A \in \mathsf{N_C} \cap \Sigma_{\mathcal{T}}$ and $a \in \mathsf{Ind}(\mathcal{A})$.

(Domain Minimize $\mathcal{A}$ with $A(a)$) If $A(a)$ is a counterexample and $(\mathcal{T}, \mathcal{A}^{-b}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A}^{-b}$, where $\mathcal{A}^{-b}$ is the result of removing from $\mathcal{A}$ all ABox assertions in which b occurs.

(Role Minimize $\mathcal{A}$ with $A(a)$) If $A(a)$ is a counterexample and $(\mathcal{T}, \mathcal{A}^{-r(b,c)}) \models A(a)$ then replace $\mathcal{A}$ by $\mathcal{A}^{-r(b,c)}$, where $\mathcal{A}^{-r(b,c)}$ be obtained by removing a role assertion $r(b, c)$ from $\mathcal{A}$.

Lemma 5. *Given a positive counterexample $(\mathcal{A}, D(a))$ with $D \in \mathsf{N_C}$, Algorithm 2 computes in polynomially many steps with respect to $|\mathcal{A}|$, $|\mathcal{H}|$, and $|\mathcal{T}|$ an ABox $\mathcal{A}'$ such that $|\mathsf{Ind}(\mathcal{A}')| \leq |\mathcal{T}|$ and $(\mathcal{A}', A(b))$ is a positive counterexample, for some $A \in \mathsf{N_C}$ and $b \in \mathsf{Ind}(\mathcal{A}')$.*

It remains to show that $\mathcal{A}$ can be made tree-shaped. We say that $\mathcal{A}$ has an (undirected) cycle if there is a finite sequence $a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_k$ such that (i) $a_0 = a_k$ and (ii) there are mutually distinct assertions of the form $r_{i+1}(a_i, a_{i+1})$ or $r_{i+1}(a_{i+1}, a_i)$ in $\mathcal{A}$, for $0 \leq i < k$. The *unfolding* of a cycle $c = a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_k$ in a given ABox $\mathcal{A}$ is obtained by replacing c by the cycle $c' = a_0 \cdot r_1 \cdot a_1 \cdot \ldots \cdot r_k \cdot a_{k-1} \cdot r_k \cdot \widehat{a}_0 \cdot r_1 \cdots \widehat{a}_{k-1} \cdot r_k \cdot a_0$, where $\widehat{a}_i$ are fresh individual names, $0 \leq i \leq k-1$, in such a way that (i) if $r(a_i, d) \in \mathcal{A}$, for an individual d not in the cycle, then $r(\widehat{a}_i, d) \in \mathcal{A}$; and (ii) if $A(a_i) \in \mathcal{A}$ then $A(\widehat{a}_i) \in \mathcal{A}$.

We prove in the full version that after every unfolding-minimisation step in Algorithm 3 the ABox $\mathcal{A}$ on the one hand becomes strictly larger and on the other does not exceed the size of the target TBox $\mathcal{T}$. Thus Algorithm 3 terminates after a polynomial number of steps yielding a tree-shaped counterexample.

Lemma 6. *Algorithm 3 computes a minimal tree shaped ABox $\mathcal{A}$ with size polynomial in $|\mathcal{T}|$ and runs in polynomially many steps in $|\mathcal{T}|$ and $|\mathcal{A}|$.*

Using Theorem 2 and Theorem 1 we obtain:

Theorem 4. *The $\mathcal{EL}_{\mathsf{lhs}}$ data retrieval framework is polynomially exact learnable.*

4 Limits of Polynomial Time Learnability

Our proof of non-polynomial learnability of general $\mathcal{EL}$ TBoxes from data retrieval examples extends previous results on non-polynomial learnability of $\mathcal{EL}$ TBoxes from

subsumptions [12]. We start by giving a brief overview of the construction in [12], show that it fails in the data retrieval setting and then demonstrate how it can be modified.

The non-learnability proof in [12] proceeds as follows. A learner tries to exactly identify one of the possible target TBoxes $\{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$, for a superpolynomial in n set $\mathfrak{L}_n$ defined below. At every stage of computation the oracle maintains a set of TBoxes S, which the learner is unable to distinguish based on the answers given so far. Initially $S = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. It has been proved that for any $\mathcal{EL}$ inclusion $C \sqsubseteq D$ either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ or the number of $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ does not exceed $|C|$. When a polynomial learner asks a membership query $C \sqsubseteq D$ the oracle answers 'yes' if $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ and 'no' otherwise. In the latter case the oracle removes polynomially many $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq D$ from S. Similarly, for any equivalence query with hypothesis $\mathcal{H}$ asked by a polynomial learning algorithm there exists a polynomial size inclusion $C \sqsubseteq D$, which can be returned as a counterexample and that excludes only polynomially many TBoxes from S. Thus, every query to the oracle reduces the size of S at most polynomially in n, but then the learner cannot distinguish between the remaining TBoxes of our initial superpolynomial set S.

The set of indices $\mathfrak{L}_n$ and the target TBoxes $\mathcal{T}_L$ are defined as follows. Fix two role names r and s. An *n-tuple* L is a sequence of role sequences $(\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, where every $\boldsymbol{\sigma}_i$ is a sequence of role names r and s, that is $\boldsymbol{\sigma}_i = \sigma_i^1 \sigma_i^2 \dots \sigma_i^n$ with $\sigma_i^j \in \{r, s\}$. Then $\mathfrak{L}_n$ is a set of n-tuples such that for every $L, L' \in \mathfrak{L}_n$ with $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$, if $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}'_j$ then $L = L'$ and $i = j$. There are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathfrak{L}_n$. For every $n > 0$ and every n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0 = \{X_i \sqsubseteq \exists r.X_{i+1} \sqcap \exists s.X_{i+1} \mid 0 \leq i < n\}$ and the following inclusions:

$$\begin{array}{ccc} A_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & & A_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \\ B_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & \cdots & B_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \end{array}$$

$$A \equiv X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M.$$

where the expression $\exists \boldsymbol{\sigma}.C$ stands for $\exists \sigma^1.\exists \sigma^2 \dots \exists \sigma^n.C$, M is a concept name that 'marks' a designated path given by $\boldsymbol{\sigma}$ and $\mathcal{T}_0$ generates a full binary tree whose edges are labelled with the role names r and s and with X_0 at the root, X_1 at level 1 and so on.

In contrast to the subsumption framework, every $\mathcal{T}_L$ *can* be exactly identified using data retrieval queries. For example, as $X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A \in \mathcal{T}_L$, a learning from data retrieval queries algorithm can learn all the sequences in the n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, by defining an ABox $\mathcal{A} = \{X_0(a_1), r(a_1, a_2), s(a_1, a_2), \dots, r(a_{n-1}, a_n), s(a_{n-1}, a_n), M(a_n)\}$ and then proceeding with unfolding and minimizing $\mathcal{A}$ via membership queries of the form $(\mathcal{T}_L, \mathcal{A}) \models A(a_1)$.

To show the non-tractability for data retrieval queries, we first modify S in such a way that the concept expression which 'marks' the sequences in $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ is now given by the set $\mathfrak{B}_n$ of all conjunctions $F_1 \sqcap \dots \sqcap F_n$, where $F_i \in \{E_i, \bar{E}_i\}$, for $1 \leq i \leq n$. Intuitively, every member of $\mathfrak{B}_n$ encodes a binary string of length n with E_i encoding 1 and $\bar{E}_i$ encoding 0. For every $L \in \mathfrak{L}_n$ and every $\mathbf{B} \in \mathfrak{B}_n$ we define $\mathcal{T}_L^{\mathbf{B}}$ as the union of $\mathcal{T}_0$ and the concept inclusions defined above with $\mathbf{B}$ replacing M.

Then for any sequence $\boldsymbol{\sigma}$ of length n there exists at most one $L \in \mathfrak{L}_n$, at most one $1 \leq i \leq n$ and at most one $\mathbf{B} \in \mathfrak{B}_n$ such that $\mathcal{T}_L^{\mathbf{B}} \models A_i \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$ and $\mathcal{T}_L^{\mathbf{B}} \models$

$B_i \sqsubseteq \exists \boldsymbol{\sigma}.\mathbf{B}$. Notice that the size of each $\mathcal{T}_L^{\mathbf{B}}$ is polynomial in n and so $\mathfrak{L}_n$ contains superpolynomially many n-tuples in the size of each $\mathcal{T}_L^{\mathbf{B}}$, with $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$. Every $\mathcal{T}_L^{\mathbf{B}}$ entails, among other inclusions, $\bigsqcap_{i=1}^{n} C_i \sqsubseteq A$, where every C_i is either A_i or B_i. Let Σ_n be the signature of the TBoxes $\mathcal{T}_L^{\mathbf{B}}$ and consider a TBox $\mathcal{T}^*$ defined as the following set of concept inclusions:

$$\exists r.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1),\ (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists r.(E_1 \sqcap \bar{E}_1),$$
$$\exists s.(E_1 \sqcap \bar{E}_1) \sqsubseteq (E_1 \sqcap \bar{E}_1),\ (E_1 \sqcap \bar{E}_1) \sqsubseteq \exists s.(E_1 \sqcap \bar{E}_1),$$
$$(E_i \sqcap \bar{E}_i) \sqsubseteq A \quad \text{for every } 1 \leq i \leq n \text{ and every } A \in \Sigma_n \cap \mathsf{N_C}$$

The basic idea of extending our TBoxes with $\mathcal{T}^*$ is that if $a \in (E_i \sqcap \bar{E}_i)^{\mathcal{I}_\mathcal{A}}$, for an ABox $\mathcal{A}$ and individual $a \in \mathsf{Ind}(\mathcal{A})$, then for all $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, we have $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(b)$, where D is any $\mathcal{EL}$ concept expression over Σ_n and $b \in \mathsf{Ind}(\mathcal{A})$ is any successor or predecessor of a (or a itself). This means that for each individual in $\mathcal{A}$ at most one $\mathbf{B}$ of the 2^n binary strings in $\mathfrak{B}_n$ can be distinguished by data retrieval queries. The following lemma enables us to respond to membership queries without eliminating too many $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ used to encode $\mathcal{T}_L^{\mathbf{B}}$ in the set of TBoxes that the learner cannot distinguish.

Lemma 7. *For any ABox $\mathcal{A}$, any $\mathcal{EL}$ concept assertion $D(a)$ over Σ_n, and any $a \in \mathsf{Ind}(\mathcal{A})$, if there is $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$ such that $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ then:*

- *either $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$, for every $L \in \mathfrak{L}_n$ and $\mathbf{B} \in \mathfrak{B}_n$, or*
- *$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|D|$ elements $L \in \mathfrak{L}_n$, or*
- *$(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$ for at most $|\mathcal{A}|$ elements $\mathbf{B} \in \mathfrak{B}_n$.*

The next lemma is immediate from Lemma 15 presented in [12]. It shows how the oracle can answer equivalence queries eliminating at most one $L \in \mathfrak{L}_n$ used to encode $\mathcal{T}_L^{\mathbf{B}}$ in the set S of TBoxes that the learner cannot distinguish.

Lemma 8. *For any $n > 1$ and any $\mathcal{EL}$ TBox $\mathcal{H}$ in Σ_n with $|\mathcal{H}| < 2^n$, there exists an ABox $\mathcal{A}$, an individual $a \in \mathsf{Ind}(\mathcal{A})$ and an $\mathcal{EL}$ concept expression D over Σ_n such that (i) the size of $\mathcal{A}$ plus the size of D does not exceed $6n$ and (ii) if $(\mathcal{H}, \mathcal{A}) \models D(a)$ then $(\mathcal{T}_L^{\mathbf{B}}, \mathcal{A}) \models D(a)$ for at most one $L \in \mathfrak{L}_n$ and if $(\mathcal{H}, \mathcal{A}) \not\models D(a)$ then for every $L \in \mathfrak{L}_n$ we have $(\mathcal{T}_L^{\mathbf{B}} \cup \mathcal{T}^*, \mathcal{A}) \models D(a)$.*

Then, by Lemmas 7 and 8, we have that: (i) any polynomial size membership query can distinguish at most polynomially many TBoxes from S; and (ii) if the learner's hypothesis is polynomial size then there exists a polynomial size counterexample that the oracle can give which distinguishes at most polynomially many TBoxes from S.

Theorem 5. *The $\mathcal{EL}$ data retrieval framework is not polynomially exact learnable.*

5 Future Work

We plan to consider an extension of the learning protocol in which arbitrary conjunctive queries are admitted in queries to the domain expert/oracle. We then still have polynomial time learnability for $\mathcal{EL}_{\mathsf{lhs}}$ but conjecture non-polynomial time learnability for DL-Lite$_{\mathcal{R}}^{\exists}$. Another extension is exact learnability for the Horn-extension of DL-Lite$_{\mathcal{R}}^{\exists}$ for which we conjecture that polynomial time learnability still holds.

Bibliography

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

[2] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369. Professional Book Center, 2005.

[3] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. Oiled: a reason-able ontology editor for the semantic web. In *KI 2001: Advances in Artificial Intelligence*, pages 396–408. Springer, 2001.

[5] D. Borchmann and F. Distel. Mining of $\mathcal{EL}$-GCIs. In *The 11th IEEE International Conference on Data Mining Workshops*, Vancouver, Canada, 11 December 2011. IEEE Computer Society.

[6] P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated reasoning*, 39(3):385–429, 2007.

[8] P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)*, 24:305–339, 2005.

[9] J. Day-Richter, M. A. Harris, M. Haendel, S. Lewis, et al. Obo-edit an ontology editor for biologists. *Bioinformatics*, 23(16):2198–2200, 2007.

[10] M. Frazier and L. Pitt. Learning From Entailment: An Application to Propositional Horn Sentences. In *ICML*, pages 120–127, 1993.

[11] B. Konev, M. Ludwig, D. Walther, and F. Wolter. The logical difference for the lightweight description logic EL. *J. Artif. Intell. Res. (JAIR)*, 44:633–708, 2012.

[12] B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 2014.

[13] C. Lutz, R. Piro, and F. Wolter. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, pages 983–988, 2011.

[14] Y. Ma and F. Distel. Learning formal definitions for snomed CT from text. In *Artificial Intelligence in Medicine - 14th Conference on Artificial Intelligence in Medicine, AIME 2013, Murcia, Spain, May 29 - June 1, 2013. Proceedings*, pages 73–77, 2013.

[15] M. A. Musen. Protégé ontology editor. *Encyclopedia of Systems Biology*, pages 1763–1765, 2013.

[16] C. Reddy and P. Tadepalli. Learning First-Order Acyclic Horn Programs from Entailment. In *in Proceedings of the 15th International Conference on Machine Learning; (and Proceedings of the 8th International Conference on Inductive Logic Programming*, pages 23–37. Morgan Kaufmann, 1998.

[17] H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.

[18] H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging OWL-DL ontologies: A heuristic approach. In *The Semantic Web–ISWC 2005*, pages 745–757. Springer, 2005.

Semantics of SPARQL under OWL 2 Entailment Regimes

Egor V. Kostylev and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford

Abstract. We study the semantics of SPARQL queries with optional matching features under entailment regimes. We argue that the normative semantics may lead to answers that are in conflict with the intuitive meaning of optional matching, where unbound variables naturally represent unknown information. We propose an extension of the SPARQL algebra that addresses these issues and is compatible with any entailment regime satisfying the minimal requirements given in the normative specification. We then study the complexity of query evaluation and show that our extension comes at no cost for regimes with an entailment relation of reasonable complexity. Finally, we show that our semantics preserves the known properties of optional matching that are commonly exploited for static analysis and optimisation.

1 Introduction

SPARQL became the standard language for querying RDF in 2008 [1]. Since then, the theoretical properties of SPARQL have been the subject of intensive research efforts and are by now relatively well-understood [2–7]. At the same time, SPARQL has become a core technology in practice, and most RDF-based applications rely on SPARQL endpoints for query formulation and processing.

The functionality of many such applications is enhanced by OWL 2 ontologies [8], which are used to provide background knowledge about the application domain, and to enrich query answers with implicit information. A new version of SPARQL, called SPARQL 1.1, was released in 2013 [9]. This new version captures the capabilities of OWL 2 by means of the so-called *entailment regimes* [10]: a flexible mechanism for extending SPARQL query answering to the W3C standards layered on top of RDF. A regime specifies which RDF graphs and SPARQL queries are legal (i.e., admissible) for the regime, as well as an entailment relation that unambiguously defines query answers for all legal queries and graphs.

The semantics of SPARQL under entailment regimes is specified for the conjunctive fragment, where queries are represented as *basic graph patterns* (i.e., sets of RDF triples with variables) and query answers are directly provided by the entailment relation of the regime. Roughly speaking, to check whether a mapping from variables of the query to nodes in the RDF graph is an answer to the query, one first transforms the query itself into an RDF graph by substituting each variable with the corresponding value, and then checks whether this graph is entailed in the regime by the original data graph [10–12].

When one goes beyond the basic fragment of SPARQL the language becomes considerably more complicated, but the effect of entailment regimes on the query semantics remains circumscribed to basic graph patterns [13, 14]. To evaluate a query one must first evaluate its component basic patterns using the relevant regime and then compose the results by means of the SPARQL algebra operations.

Of particular interest from both a theoretical and a practical perspective is the extension of the basic fragment of SPARQL with the *optional matching feature*, which is realised in the language by means of the `OPTIONAL` operator (abbreviated by OPT in this paper). This feature allows the optional information to be added to query answers only when the information is available in the RDF data graph: if the optional part of the query does not match the data, then the relevant variables are left unbounded in query answers.

One of the main motivations behind optional matching in SPARQL is to deal with the "lack of regular, complete structures in RDF graphs" (see [9] Section 6) and hence with the inherent incompleteness of information in RDF data sources where only partial information about the relevant Web resources is typically available. In this setting, an unbound variable in an answer mapping is naturally interpreted as a "null" value, meaning that there might exist a binding for this variable if we consider other information elsewhere on the Web, but none is currently available in the RDF graph at hand. Another (and slightly different) motivation for optional matching was to introduce a mechanism for "not rejecting solutions because some part of the query pattern does not match" [1]; in this sense, one would naturally expect optional matching to either extend solutions with the optional information, or to leave solutions unchanged. Both readings of optional matching coincide if we focus just on RDF, and they are faithfully captured by the normative semantics. In this paper we argue that they naturally diverge once we consider more sophisticated entailment regimes. Furthermore, the differences that arise can have a major impact on expected answers.

To make this discussion concrete, let us briefly discuss a simple example of an RDF graph representing the direct train lines between UK cities as well as ferry boat transfers from UK cities to international destinations. Let this graph be exhaustive in its description of rail connections, but much less so in what concerns ferry transfers. We may exploit optional matching to retrieve all direct train connections between cities X and Y, extended with ferry transfers from Y to other cities Z whenever possible. Under the normative semantics of SPARQL we may obtain answers $(\mathit{London}, \mathit{Oxford}, -)$ and $(\mathit{London}, \mathit{Holyhead}, -)$ provided the graph has information about direct train lines from *London* to both *Oxford* and *Holyhead*, but no matching can be found in the graph for ferry connections starting from *Oxford* or *Holyhead* to other cities. Suppose next that the data graph is extended to a graph corresponding to an OWL 2 ontology in which it is stated that inland cities do not have ferry connections, and that *Oxford* is an inland city. The ontology establishes a clear distinction between *Oxford* and *Holyhead*: whereas the former is inland and cannot have ferry connections, the latter may still well be (and indeed is) a coastal city offering a number of transfers to international destinations. The normative OWL 2 direct semantics

entailment regime, however, does not distinguish between the case of *Holyhead* (where the information about ferry connections is still unknown) and *Oxford* (where the information is certain), and both answers would be returned. In this way, the normative semantics adopts the reading of optional matching where the optional information is used to complete (but never discard) query answers. In contrast, under the reading of unbounded variables as placeholders for unknown information, one would naturally expect the answer on *Oxford* to be ruled out. Indeed, if our goal were to find rail to ferry transfers starting from *London* and terminating in *Dublin* by first querying this graph and then looking for the missing information elsewhere on the Web, discarding cities like *Oxford* on the first stage would significantly facilitate our task.

In this paper, we propose an alternative semantics for the OPT operator which adopts the aforementioned reading of optional matching as an incomplete "null". We call our semantics *strict*, which reflects the fact that it rules out those answers in which unbound variables in the optional part cannot be matched to *any* consistent extension of the input graph. Our semantics is given as an extension of the SPARQL algebra and hence satisfies the expected compositionality properties of algebraic query languages. Furthermore, it is backwards-compatible with the normative semantics for regimes in which all legal graphs are consistent, such as the RDF regime [10]. We also study the complexity of query evaluation and show that our extension comes at no cost for regimes in which entailment is not harder than query evaluation under normative semantics for the RDF regime. Finally, we show that our semantics preserves the known properties of optional matching that are commonly exploited for static analysis and optimisation.

This paper is an updated version of the work [15].

2 SPARQL 1.1 under Entailment Regimes

In this section, we formalise the syntax and normative semantics of a core fragment of SPARQL 1.1 with optional matching under entailment regimes. Our formalisation is based on the normative specification documents [9–11] and builds on the well-known foundational works on SPARQL [2, 3, 6].

2.1 Syntax

Let $\mathbf{I}$, $\mathbf{L}$, and $\mathbf{B}$ be infinite sets of *IRIs*, *literals*, and *blank nodes*, respectively. The set of *RDF terms* $\mathbf{T}$ is $\mathbf{I} \cup \mathbf{L} \cup \mathbf{B}$. An *RDF triple* is a triple $(s\ p\ o)$ from $\mathbf{T} \times \mathbf{I} \times \mathbf{T}$, with s called *subject*, p *predicate*, and o *object*. An *(RDF) graph* is a finite set of RDF triples. Assume additionally the existence of a countably infinite set $\mathbf{V}$ of variables disjoint from $\mathbf{T}$. A *triple pattern* is a tuple from $(\mathbf{T} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{T} \cup \mathbf{V})$. A *basic graph pattern (BGP)* is a finite set of triple patterns. *Built-in conditions* are conditions of the form $bound(?X)$, $?X = c$, and $?X = ?Y$ for $?X, ?Y \in \mathbf{V}$ and $c \in \mathbf{T}$, and their Boolean combinations.

Complex graph patterns are constructed from BGPs using a range of available operators that are applicable to graph patterns and built-in conditions. We

focus on the AND-OPT-FILTER fragment (i.e., we consider neither union nor projection), which is widely accepted to be the fundamental core of SPARQL [2]. In this setting, *graph patterns* are inductively defined as follows (e.g., see [11]):

1. every BGP is a graph pattern;
2. if P_1 and P_2 are graph patterns that share no blank nodes then $(P_1 \,\mathsf{AND}\, P_2)$ and $(P_1 \,\mathsf{OPT}\, P_2)$ are graph patterns (called AND and OPT *patterns*); and
3. if P is a graph pattern and R is a built-in condition, then $(P \,\mathsf{FILTER}\, R)$ is a graph pattern (called FILTER *pattern*).

In what follows $\mathsf{vars}(P)$ (respectively $\mathsf{triples}(P)$) denotes all the variables from $\mathbf{V}$ (respectively all triple patterns) that appear in a graph pattern P.

We conclude with the definition of a special class of graph patterns with intuitive behaviour [2]. A graph pattern is *well-designed* if and only if for each of its OPT subpatterns $(P_1 \,\mathsf{OPT}\, P_2)$ the pattern P_1 mentions all the variables of P_2 which appear outside this subpattern. Note that all graph patterns in the examples of this paper are well-designed.

2.2 Semantics of BGPs under Entailment Regimes

The semantics of graph patterns is defined in terms of *mappings*; that is, partial functions from variables $\mathbf{V}$ to terms $\mathbf{T}$. The *domain* $\mathsf{dom}(\mu)$ of a mapping μ is the set of variables on which μ is defined. The set of triples obtained from a BGP P by replacing each $?X$ from $\mathsf{dom}(\mu)$ by $\mu(?X)$ is denoted by $\mu(P)$.

Two mappings μ_1 and μ_2 are *compatible* (written as $\mu_1 \sim \mu_2$) if $\mu_1(?X) = \mu_2(?X)$ for all variables $?X$ which are in both $\mathsf{dom}(\mu_1)$ and $\mathsf{dom}(\mu_2)$. If $\mu_1 \sim \mu_2$, then we write $\mu_1 \cup \mu_2$ for the mapping obtained by extending μ_1 with μ_2 on variables undefined in μ_1. A mapping μ_1 is *subsumed* by a mapping μ_2 (written $\mu_1 \sqsubseteq \mu_2$) iff $\mu_1 \sim \mu_2$ and $\mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu_2)$. Finally, a set of mappings Ω_1 is *subsumed* by a set of mappings Ω_2 (written $\Omega_1 \sqsubseteq \Omega_2$) iff for each $\mu_1 \in \Omega_1$ there exists $\mu_2 \in \Omega_2$ such that $\mu_1 \sqsubseteq \mu_2$.

Based on [10], an *(entailment) regime* $\mathfrak{R}$ is a tuple $(\mathbf{R}, \mathcal{G}, \mathcal{P}, \mathcal{C}, [\![\cdot]\!])$, where

1. $\mathbf{R}$ is a set of *reserved* IRIs from $\mathbf{I}$;
2. $\mathcal{G}$ is the set of *legal* graphs;
3. $\mathcal{P}$ is the set of *legal* BGPs;
4. $\mathcal{C}$ is the set of *consistent* graphs, such that $\mathcal{C} \subseteq \mathcal{G}$; and
5. $[\![\cdot]\!]$ is the *query answering* function, that takes a graph G from $\mathcal{G}$ and a BGP P from $\mathcal{P}$ and returns either a set $[\![P]\!]_G$ of mappings μ such that $\mathsf{dom}(\mu) = \mathsf{vars}(P)$, if $P \in \mathcal{C}$; or Err, otherwise.

As in most theoretical works on SPARQL [2, 3, 6, 16], we assume that the query answering function returns a set of mappings, rather than a multiset.

The definitions of query answering and consistency in a regime are based on an entailment relation [10], which is also specified as part of the regime. We do not model the entailment relation explicitly, but assume two conditions that capture the effects of any reasonable entailment relation on legality and consistency. All regimes mentioned in the normative specification satisfy these properties and in this paper we consider only regimes that do so.

(C1) If graphs G, G_1 and G_2 are legal, and there is $h : \mathbf{T} \to \mathbf{T}$, preserving $\mathbf{R}$, such that $h(G_1 \cup G_2) \subseteq G$ then $G_1 \cup G_2$ is legal; if, in addition, G is in $\mathcal{C}$ then $G_1 \cup G_2$ is also in $\mathcal{C}$.

(C2) If a BGP P is in $\mathcal{P}$ then $\mu(P)$ is in $\mathcal{G}$ for any (total) $\mu : \mathbf{V} \to (\mathbf{T} \setminus \mathbf{R})$, such that $\mu(P)$ is a graph; if also $\mu(P)$ is in $\mathcal{C}$ then $\mu \in [\![P]\!]_{\mu(P)}$.

Condition (C1) formalises (a weak form of) the monotonicity of legality and consistency: an illegal graph that is a union of legal ones cannot be made legal by identifying and renaming of non-reserved terms or adding triples to it; moreover, a similar property holds for consistency. Condition (C2) guarantees, that "freezing" variables of a legal BGP to non-reserved terms gives us a legal graph, and, moreover, if such a graph is consistent, then the answer of the BGP on this graph contains the mapping corresponding to the "freezing".

The notions introduced in the remainder of this paper are parameterised with a regime $\mathfrak{R}$, which is not mentioned explicitly for brevity.

2.3 Normative Semantics under Entailment Regimes

Following [2], now we show how the query answering function $[\![\cdot]\!]$ extends to complex graph patterns (we refer to [2] for details). A mapping μ *satisfies* a built-in condition R, denoted $\mu \models R$, if one of the following holds:

1. R is $bound(?X)$ and $?X \in \mathsf{dom}(\mu)$; or
2. R is $?X = c$, $?X \in \mathsf{dom}(\mu)$, and $\mu(?X) = c$; or
3. R is $?X = ?Y$, $?X \in \mathsf{dom}(\mu)$, $?Y \in \mathsf{dom}(\mu)$, and $\mu(?X) = \mu(?Y)$; or
4. R is an evaluating to true Boolean combination of other built-in conditions.

The *join*, *difference*, and *left outer join* of sets of mappings Ω_1, Ω_2 are as follows:

$$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ such that } \mu_1 \sim \mu_2\},$$
$$\Omega_1 \setminus \Omega_2 = \{\mu_1 \mid \mu_1 \in \Omega_1, \text{ there is no } \mu_2 \in \Omega_2 \text{ such that } \mu_1 \sim \mu_2\},$$
$$\Omega_1 \mathbin{⟕} \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2).$$

A graph pattern is *legal* for a regime $\mathfrak{R}$ if all the BGPs it contains are legal. The *normative query answering* function $[\![\cdot]\!]^n$ is inductively defined for all legal graph patterns P on the base of $[\![\cdot]\!]$ as follows. For graphs G from $\mathcal{C}$ we have:

1. if P is a BGP then $[\![P]\!]^n_G = [\![P]\!]_G$;
2. if P is $(P_1 \ \mathsf{AND}\ P_2)$ then $[\![P]\!]^n_G = [\![P_1]\!]^n_G \bowtie [\![P_2]\!]^n_G$;
3. if P is $(P_1 \ \mathsf{OPT}\ P_2)$ then $[\![P]\!]^n_G = [\![P_1]\!]^n_G \mathbin{⟕} [\![P_2]\!]^n_G$; and
4. if P is $(P' \ \mathsf{FILTER}\ R)$ then $[\![P]\!]^n_G = \{\mu \mid \mu \in [\![P']\!]^n_G \text{ and } \mu \models R\}$.

If $G \notin \mathcal{C}$ then $[\![P]\!]^n_G = Err$ for any graph pattern P (which again coincides with $[\![P]\!]_G$ when P is a BGP). Note, that by these definitions $\mu \in [\![P]\!]^n_G$ implies that $\mathsf{dom}(\mu) \subseteq \mathsf{vars}(P)$, but this inclusion may be strict if P contains OPT operator.

Two legal patterns P_1 and P_2 are *equivalent (under normative semantics)*, denoted by $P_1 \equiv^n P_2$, if $[\![P_1]\!]^n_G = [\![P_2]\!]^n_G$ for every RDF graph $G \in \mathcal{G}$.

3 On Optional Matching Under the Normative Semantics

One of the main motivations for optional matching in SPARQL was to deal with the "lack of regular, complete structures in RDF graphs" [9]. Indeed, RDF data

is loosely structured, and in many applications it is not satisfactory to reject an answer if some relevant information is missing. For example, if we are interested in retrieving the names, emails, and websites of employees, we may not want to discard a partial answer involving the name and email address of an employee merely because the information on the employee's website is not available in the graph. The normative semantics was designed to deal with such situations: the optional information is included in query answers only when the information is available; otherwise, the relevant variables are left unbounded. An unbound variable in an answer is thus a manifestation of inherent incompleteness of RDF data sources, and the missing information is interpreted as unknown.

This natural interpretation of query results, however, no longer holds if the query is evaluated under certain entailment regimes, as we illustrate next by means of examples. In these and all other examples given later on, we focus on the OWL 2 direct semantics regime. In order for an RDF graph to be legal for this regime, it must correspond to an OWL 2 ontology; similarly, legal BGPs must correspond to an extended ontology in which variables are allowed [10]. Thus, in the examples we express RDF graphs and BGPs in (extended) OWL 2 functional syntax, and use words "ontology" and "graph" interchangeably (we also omit declaration axioms in ontologies and BGPs to avoid clatter).

Example 1. Consider the OWL 2 ontology $\mathcal{O}_1$ consisting of the axioms

$$\mathsf{ClassAssertion}(\mathit{InlandCity}\ \mathit{Oxford}),\quad \mathsf{PropertyAssertion}(\mathit{train}\ \mathit{London}\ \mathit{Oxford}),$$
$$\mathsf{ClassAssertion}(\mathit{CoastalCity}\ \mathit{Holyhead}),\mathsf{PropertyAssertion}(\mathit{train}\ \mathit{London}\ \mathit{Holyhead}),$$
$$\mathsf{PropertyDomain}(\mathit{ferry}\ \mathit{CoastalCity}),\quad \mathsf{DisjointClasses}(\mathit{CoastalCity}\ \mathit{InlandCity}).$$

Consider also the following graph pattern P_1, which we wish to evaluate over $\mathcal{O}_1$:

$$\mathsf{PropertyAssertion}(\mathit{train}\quad ?X\quad ?Y)\ \mathsf{OPT}\ \mathsf{PropertyAssertion}(\mathit{ferry}\quad ?Y\quad ?Z).$$

Intuitively, solutions to P_1 provide direct train lines from city X to city Y as well as, optionally, the ferry transfers from Y to other cities Z. Under the normative semantics, the BGPs in P_1 are evaluated separately. In particular, the optional BGP is evaluated to the empty set, and $[\![P_1]\!]^n_{\mathcal{O}_1} = \{\mu_1, \mu_2\}$, where

$$\mu_1 = \{?X \mapsto \mathit{London}, ?Y \mapsto \mathit{Oxford}\} \text{ and } \mu_2 = \{?X \mapsto \mathit{London}, ?Y \mapsto \mathit{Holyhead}\}.$$

In both answers, variable $?Z$ is unbounded and hence we conclude that $\mathcal{O}_1$ contains no relevant information about ferry connections starting from *Oxford* or *Holyhead*. However, the nature of the lack of such information is fundamentally different. On the one hand, the connections from *Holyhead* (e.g., to *Dublin*) are missing from $\mathcal{O}_1$ just by the incompleteness of the information in the graph, which is usual in (and also a feature of) Semantic Web applications. On the other hand, *Oxford* cannot have a ferry connection because it is a landlocked city, and hence the information about its (lack of) ferry connections is certain. Thus, the normative semantics cannot distinguish between unknown and non-existent ferry connections. However, if we adhere to the reading of unbounded variables as incomplete information or "nulls", then μ_1 should not be returned as an answer.

The issues described in this example become even more apparent in cases where the optional part alone cannot be satisfied, as in the following example.

Example 2. Consider the ontology $\mathcal{O}_2$ with the axioms

$$\mathsf{ClassAssertion}(\mathit{Person}\ \mathit{Peter}), \quad \mathsf{DisjointProperties}(\mathit{hasFather}\ \mathit{hasMother}).$$

Furthermore, consider the following pattern P_2:

$$\mathsf{ClassAssertion}(\mathit{Person}\ ?X)\ \mathsf{OPT}\ (\{ \\ \mathsf{PropertyAssertion}(\mathit{hasFather}\ ?X\ ?Y), \mathsf{PropertyAssertion}(\mathit{hasMother}\ ?X\ ?Y)\}).$$

The optional BGP does not match to anything, so $[\![P_2]\!]^n_{\mathcal{O}_2}$ consists of $\{?X \mapsto \mathit{Peter}\}$. However, this BGP is in contradiction with the disjointness axiom: under the OWL 2 regime, no solution to P_2 exists for any ontology with this axiom.

As these examples suggest, if we interpret unbound variables in answers to queries with optional parts as an indication of unknown information in the data graph, then the normative semantics may yield counter-intuitive answers. At the core of this issue is the inability of the normative semantics to distinguish between answers in which it is possible to assign values to the missing optional part (a natural reflection of incompleteness in the data), and those where this is impossible (a reflection that the missing information is incompatible with the answer). This distinction is immaterial for regimes without inconsistencies, but it becomes apparent in more sophisticated regimes, such as those based on OWL 2.

4 Semantics of Strict Optional Matching

In this section, we propose our novel semantics for optional matching under regimes. In a nutshell, our semantics addresses the issues described in Section 3 by ruling out those answer mappings where unbound variables in the optional part cannot be matched to any consistent extension of the input graph. Our semantics is therefore *strict*, in the sense that only answers in which unbound variables are genuine manifestations of incompleteness in the data are returned.

4.1 Definition of Strict Semantics

We start by introducing the notion of a frozen RDF graph for a pattern P and a mapping μ. Roughly speaking, this graph is obtained by taking all the triple patterns in P and transforming them into RDF triples by applying the extension of μ where unbounded variables are "frozen" to arbitrary fresh constants.

Definition 1. *Let $\mathfrak{R} = (\mathbf{R}, \mathcal{G}, \mathcal{P}, \mathcal{C}, [\![\cdot]\!])$ be an entailment regime. Let P be a legal graph pattern, and let μ be a mapping from variables $\mathbf{V}$ to RDF terms $\mathbf{T}$. Then, the* freezing G^P_μ *of P under μ is the RDF graph $\bar{\mu}(\mathsf{triples}(P))$, where $\bar{\mu}$ is the mapping that extends μ by assigning each variable in $\mathsf{vars}(P)$, which is not in $\mathsf{dom}(\mu)$, to a globally fresh IRI from $\mathbf{I}$ not belonging to $\mathbf{R}$.*

The freezing G^P_μ depends only on the candidate mapping μ and the triple patterns occurring in P; thus, it does not depend either on the specific operators used in P, or on the RDF graph over which the query pattern is to be evaluated.

Example 3. For pattern P_1 and mappings μ_1, μ_2 from Example 1 the freezings $G^{P_1}_{\mu_1}$ and $G^{P_1}_{\mu_2}$ have the following form, for fresh IRIs k and ℓ:

$\{\mathsf{PropertyAssertion}(\mathit{train\ London\ Oxford}), \mathsf{PropertyAssertion}(\mathit{ferry\ Oxford\ k})\}$,
$\{\mathsf{PropertyAssertion}(\mathit{train\ London\ Holyhead}), \mathsf{PropertyAssertion}(\mathit{ferry\ Holyhead\ \ell})\}$.

Intuitively, the freezing represents the simplest and most general RDF graph over which all the undefined variables in a given solution mapping could be bounded to concrete values. Thus, if G^P_μ together with the input graph G is not a consistent graph for the relevant regime, we can conclude, using condition (C1) of the regime, that the undefined variables in μ will never be matched to concrete values in any consistent extension of G and hence μ should be ruled out as an answer. On the other hand, if $G \cup G^P_\mu$ is consistent, then such an extension exists and, by condition (C2), the undefined variables can be mapped in this extension.

Definition 2. *Let $\mathfrak{R} = (\mathbf{R}, \mathcal{G}, \mathcal{P}, \mathcal{C}, [\![\cdot]\!])$ be an entailment regime. A mapping μ is* $\mathfrak{R}$-admissible *for a graph $G \in \mathcal{C}$ and legal graph pattern P if $G \cup G^P_\mu$ is a graph belonging to $\mathcal{C}$. The set of all $\mathfrak{R}$-admissible mappings for a consistent graph G and a legal graph pattern P is denoted as $Adm(G, P)$.*

Example 4. Clearly, $\mathcal{O}_1 \cup G^{P_1}_{\mu_1}$ is inconsistent since ferries only depart from coastal cities, but Oxford is an inland city. In contrast, $\mathcal{O}_1 \cup G^{P_1}_{\mu_2}$ is consistent. Thus, we have $\mu_1 \notin Adm(\mathcal{O}_1, P)$, but $\mu_2 \in Adm(\mathcal{O}_1, P)$.

We are now ready to formalise our semantics.

Definition 3. *Given an entailment regime $\mathfrak{R} = (\mathbf{R}, \mathcal{G}, \mathcal{P}, \mathcal{C}, [\![\cdot]\!])$, the* strict query answering *function $[\![\cdot]\!]^s$ is defined for legal graph patterns P and $G \in \mathcal{C}$ as follows:*

1. *if P is a BGP then $[\![P]\!]^s_G = [\![P]\!]_G$;*
2. *if P is P_1 AND P_2 then $[\![P]\!]^s_G = ([\![P_1]\!]^s_G \bowtie [\![P_2]\!]^s_G) \cap Adm(G, P)$;*
3. *if P is P_1 OPT P_2 then $[\![P]\!]^s_G = ([\![P_1]\!]^s_G \bowtie [\![P_2]\!]^s_G) \cap Adm(G, P)$; and*
4. *if P is P' FILTER R then $[\![P]\!]^s_G = \{\mu \mid \mu \in [\![P']\!]^s_G \text{ and } \mu \models R\}$,*

where $\cap$ denotes the standard set-theoretic intersection. If $G \notin \mathcal{C}$ then $[\![P]\!]^s_G = Err$ for any graph pattern P. Finally, legal patterns P_1 and P_2 are equivalent (under strict semantics)*, written $P_1 \equiv^s P_2$, if $[\![P_1]\!]^s_G = [\![P_2]\!]^s_G$ for any legal G.*

Example 5. The strict semantics behaves as expected for our examples: $[\![P_1]\!]^s_{\mathcal{O}_1} = \{\mu_1\}$ holds for $\mathcal{O}_1$ and P_1 from Example 1, while $[\![P_2]\!]^s_{\mathcal{O}_2} = \emptyset$ holds for Example 2.

The strict and normative semantics coincide in two limit cases. First, if the entailment regime does not allow for inconsistent graphs (i.e., if $\mathcal{C} = \mathcal{G}$) as is the case for the RDF regime [10], then $[\![P]\!]^s_G = [\![P]\!]^n_G$ for every legal pattern P and graph G. Second, if the relevant pattern P is OPT-free then the freezing for

every candidate answer mapping contains no fresh IRIs and is $\mathfrak{R}$-entailed by G; thus, we again have $[\![P]\!]^s_G = [\![P]\!]^n_G$ for every legal graph G.

Thus, the difference between the normative semantics $[\![\cdot]\!]^n$ and strict semantics $[\![\cdot]\!]^s$ manifests only for regimes that admit inconsistency, and is circumscribed to the presence of OPT in graph patterns, where non-admissible mappings are excluded in the case of the strict semantics. Note, however, that even if a mapping μ_1 (respectively μ_2) is admissible for a subpattern P_1 (respectively P_2) containing OPT, it is possible for $\mu_1 \cup \mu_2$ not to be admissible for the joined pattern $P = P_1$ AND P_2. Thus, the admissibility restriction is also explicitly reflected in the semantics of AND given in Definition 3. This is illustrated in the example given next.

Example 6. Consider ontology $\mathcal{O}_3$, consisting of the axioms

SubClassOf(
  IntersectionOf(SomeValuesFrom(*husband* Thing) SomeValuesFrom(*wife* Thing))
  Nothing),
ClassAssertion(*Person Mary*).

The first axiom establishes that a person cannot have both a husband and a wife. Consider also the following well-designed graph pattern P_3:

(ClassAssertion(*Person* ?*X*) OPT (PropertyAssertion(*husband* ?*X* ?*Y*))) AND
(ClassAssertion(*Person* ?*X*) OPT (PropertyAssertion(*wife* ?*X* ?*Z*))).

Clearly, $\mu = \{?X \mapsto \mathit{Mary}\}$ belongs to the strict answer to each of the OPT subpatterns of P_3 since each of them independently can match to a consistent extension of $\mathcal{O}_3$. However, μ is not admissible for P_3 since *Mary* has both a husband and a wife in $G^{P_3}_\mu$, and hence $\mathcal{O}_3 \cup G^{P_3}_\mu$ is inconsistent. Thus, $[\![P_3]\!]^s_{\mathcal{O}_3} = \emptyset$.

4.2 Comparing the Normative and Strict Semantics

Our previous examples support the expected behaviour of our semantics, namely that its effect is circumscribed to filtering out problematic answers returned under the normative semantics. We next formally show that our semantics behaves as expected *in general*, provided that we restrict ourselves to well-designed patterns and negation-free FILTER expressions (which are rather mild restrictions).

It is known that patterns which are not well-designed easily lead to unexpected answers, even under the normative semantics (we refer to [2] for a detailed discussion). Therefore, it comes at no surprise that the intuitive behaviour of our semantics is only guaranteed under this assumption.

Theorem 1. *Let* $\mathfrak{R} = (\mathbf{R}, \mathcal{G}, \mathcal{P}, \mathcal{C}, [\![\cdot]\!])$ *be an entailment regime. The inclusion* $[\![P]\!]^s_G \sqsubseteq [\![P]\!]^n_G$ *holds for any graph* G *from* $\mathcal{C}$ *and any legal well-designed graph pattern* P *which does not use negation in* FILTER *expressions.*

Note that Theorem 1 is formulated in terms of subsumption, instead of set-theoretic containment. The rationale behind this formulation is clarified next.

Example 7. Consider the ontology $\mathcal{O}_1'$, which is obtained from $\mathcal{O}_1$ in Example 1 by removing all axioms involving *Holyhead*, and adding the axiom

$$\mathsf{PropertyAssertion}(\mathit{bus}\ \mathit{Canterbury}\ \mathit{London}).$$

Consider also the following graph pattern P_1':

$$\mathsf{PropertyAssertion}(\mathit{bus}\ ?U\ ?X)\ \mathsf{OPT}\ (\mathsf{PropertyAssertion}(\mathit{train}\ ?X\ ?Y)\ \mathsf{OPT}\ \mathsf{PropertyAssertion}(\mathit{ferry}\ ?Y\ ?Z)).$$

The mapping $\mu = \{?U \mapsto \mathit{Canterbury}, ?X \mapsto \mathit{London}, ?Y \mapsto \mathit{Oxford}\}$ is returned by the normative semantics. As already discussed, Oxford is an inland city and hence cannot have ferry connections; thus, μ is not returned under strict semantics. However, it may be possible to reach a ferry connection from London (although none is given), and hence the answer $\mu' = \{?U \mapsto \mathit{Canterbury}, ?X \mapsto \mathit{London}\}$ is returned instead of μ under strict semantics. Clearly, μ' is not a normative answer and $[\![P_1']\!]^s_{\mathcal{O}_1'} \not\subseteq [\![P_1']\!]^n_{\mathcal{O}_1'}$; however, $\mu' \sqsubseteq \mu$ and $[\![P_1']\!]^s_{\mathcal{O}_1'} \sqsubseteq [\![P_1']\!]^n_{\mathcal{O}_1'}$.

5 Computational Properties and Static Optimisation

In this section we first study the computational properties of our semantics. We show that the complexity of graph pattern evaluation under strict and normative semantics coincide, provided that consistency checking is feasible in PSPACE for the regime at hand. Then we focus on static query analysis, and in particular on pattern equivalence. We show that the key equivalence-preserving transformation rules that have been proposed for static optimisation of SPARQL queries continue to hold if we consider equivalence under strict semantics.

5.1 Complexity of Strict Graph Pattern Evaluation

Recall that the graph pattern evaluation is the key problem in SPARQL. In the context of entailment regimes, it is defined as follows, where x is either n or s.

Graph Pattern Evaluation	
Input:	Regime $\mathfrak{R}$, legal graph G, legal graph pattern P, and mapping μ.
Question:	Is $\mu \in [\![P]\!]^x_G$ under the regime $\mathfrak{R}$?

Here, when we say that regime $\mathfrak{R}$ is a part of the input, we mean that it includes two oracle functions checking consistency of legal graphs and evaluating legal BGPs over legal graphs, respectively. In what follows, we refer to the problem as Normative if $x = n$, and as Strict if $x = s$.

It is known that the normative graph pattern evaluation problem is in PSPACE for the RDF regime [2]. We next argue that membership in PSPACE holds in general for any regime satisfying the basic properties discussed in Section 2 and for both normative and strict versions of the problem, provided that the complexity of both oracles of the regime is in PSPACE.

Theorem 2. NORMATIVE *and* STRICT GRAPH PATTERN EVALUATION *problems are in* PSPACE*, provided the oracles associated to input regimes are in* PSPACE*.*

Consequently, the use of our strict semantics does not increase the computational complexity for reasonable regimes. In particular, it follows directly from Theorem 2 that the evaluation problem is in PSPACE under both semantics for the tractable entailment regimes associated to the OWL 2 profiles [17].

It is also known that graph pattern evaluation under normative semantics is PSPACE-hard for the RDF regime [2]. To formulate a general hardness result that holds for *any* regime we would need to require additional properties for a regime to qualify as "reasonable". In order not to unnecessarily complicate the presentation, we simply point out that PSPACE-hardness holds for all the regimes in the specification under both normative and strict semantics [10].

5.2 Static Analysis and Optimisation

Static analysis and optimisation of SPARQL queries has received significant attention in recent years [4, 6, 18–20]. A key ingredient for optimisation is the availability of a comprehensive catalog of equivalence-preserving transformation rules for SPARQL patterns. A rich set of such equivalences for normative semantics and RDF regime is established in [2] and [4]. Some of these equivalences, such as idempotence, commutativity, and associativity of the AND operator, hold without any restrictions (for our core fragment of SPARQL). However, those that involve OPT are more intricate and hold only for well-designed patterns. The claim of this section is that these equivalences continue to hold for any entailment regime, under both normative and strict semantics.

Theorem 3. *The following equivalences hold for any entailment regime, provided the graph patterns on both sides are legal and well-designed, for* $x \in \{n, s\}$*:*

$$(P_1 \,\mathsf{OPT}\, P_2) \,\mathsf{FILTER}\, R \equiv^x (P_1 \,\mathsf{FILTER}\, R) \,\mathsf{OPT}\, P_2,$$
$$P_1 \,\mathsf{AND}\, (P_2 \,\mathsf{OPT}\, P_3) \equiv^x (P_1 \,\mathsf{AND}\, P_2) \,\mathsf{OPT}\, P_3,$$
$$(P_1 \,\mathsf{OPT}\, P_2) \,\mathsf{OPT}\, P_3 \equiv^x (P_1 \,\mathsf{OPT}\, P_3) \,\mathsf{OPT}\, P_2.$$

6 Conclusion

In this paper, we have proposed a novel semantics for optional matching in SPARQL under entailment regimes where unbound variables in answer mappings are naturally interpreted as "null" values. Our *strict* semantics has been designed to deal in a faithful way with the "lack of regular, complete structures in RDF graphs" and hence with the fundamental incompleteness of information on the Semantic Web [1]. We believe that both strict and normative semantics are valid, but one may be more appropriate than the other in certain applications. Both semantics are compatible at a fundamental level and it would be possible to exploit them in the same application by letting users commit to one or the other explicitly when posing queries. Integrating them in a clean way from a syntactic point of view is more tricky, and it is something we leave for future investigation.

References

1. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (2008) Available at http://www.w3.org/TR/rdf-sparql-query/.
2. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3) (2009)
3. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: ISWC. (2008) 114–129
4. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: ICDT. (2010) 4–33
5. Arenas, M., Pérez, J.: Querying semantic web data with SPARQL. In: PODS. (2011) 305–316
6. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. ACM Trans. Database Syst. **38**(4) (2013) 25
7. Polleres, A.: From SPARQL to rules (and back). In: WWW. (2007) 787–796
8. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-style Syntax. W3C Recommendation (2012) Available at http://www.w3.org/TR/owl2-syntax/.
9. W3C SPARQL Working Group: SPARQL 1.1 Query language. W3C Recommendation (2013) Available at http://www.w3.org/TR/sparql11-query/.
10. Glimm, B., Ogbuji, C.: SPARQL 1.1 Entailment Regimes. W3C Recommendation (2013) Available at http://www.w3.org/TR/sparql11-entailment/.
11. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: ISWC. (2010) 241–256
12. Kollia, I., Glimm, B.: Optimizing SPARQL query answering over OWL ontologies. J. Artif. Intell. Res. **48** (2013) 253–303
13. Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: ISWC. (2014) 552–567
14. Arenas, M., Gottlob, G., Pieris, A.: Expressive languages for querying the semantic web. In: PODS. (2014) 14–26
15. Kostylev, E.V., Cuenca Grau, B.: On the semantics of SPARQL queries with optional matching under entailment regimes. In: ISWC. (2014)
16. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: ISWC. (2006) 30–43
17. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles (27 October 2009) Available at http://www.w3.org/TR/owl2-profiles/.
18. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: SPARQL query containment under RDFS entailment regime. In: IJCAR. (2012) 134–148
19. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: SPARQL query containment under $\mathcal{SHI}$ axioms. In: AAAI. (2012)
20. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: Evaluating and benchmarking SPARQL query containment solvers. In: ISWC. (2013) 408–423

Towards Expressive Metamodelling with Instantiation

Petra Kubincová, Ján Kľuka, and Martin Homola

Comenius University in Bratislava, Faculty of Mathematics, Physics, and Informatics,
Mlynská dolina, 842 48 Bratislava, Slovakia
petra.kubincova@gmail.com, {kluka,homola}@fmph.uniba.sk

Abstract. In metamodelling we allow concepts and roles to be classified into "meta" concepts and to be reasoned with as if they were individuals. This is useful in modelling of certain complex domains or in reasoning *about* ontology entities for sake of verification of methodological constraints. What many proposed metamodelling languages lack is the ability to model with instantiation – the relation between an instance and a concept it belongs to – similar to the ability to express restrictions on rdf:type in the undecidable OWL Full. We investigate a variant of higher-order description logics combining various desired metamodelling features, including: (a) a fixly interpreted instanceOf role connecting instances with their concepts, freely usable in modelling; (b) promiscuous concepts that may have individuals, other concepts, and roles as instances at the same time; but also (c) strictly typed concepts allowing only a certain type of instances. We show the decidability of two expressive fragments by means of reduction.

Keywords: Description logics, higher-order logic, metamodelling.

1 Introduction

Metamodelling allows to model with extensional entities (concepts and roles) as if they were individuals: they may be instances of other concepts, and they may be related to other entities with roles. This is useful in some complex domains with a high number of extensional entities where it makes sense to further categorize them into meta-concepts, and use meta-roles to express relations among them. For example, in the biological taxonomy the rank Species contains the taxon Giraffa camelopardalis. Taxa are regular concepts (they classify specimens); ranks such as Species and Genus are thus meta-concepts. Ranks may be further classified into meta-meta-concepts, such as Rank. Other applications of metamodelling include reasoning *about* ontology entities for verification of methodological constraints. More details can be found in the literature [3,4,10,21].

Finding an agreement on how a DL suitable for metamodelling should look like is hard. Logics proposed for this purpose include works of Motik [16] and De Giacomo et al. [3] who take an approach similar to RDF [8] and allow all entities to have threefold semantics – of individual, concept, and role – simultaneously; also concepts and roles are promiscuous – the same concept (role) may contain (connect) any kind of entity. These works rely on HiLog-style semantics developed originally for higher-order Prolog [2]. It is essentially first-order and corresponds to Ullmann's meaning triangle [22]: Each entity name is interpreted as an intension – internal meaning with no known structure, thus technically an object. Concept and role extensions are subsequently assigned

to intensions. De Giacomo et al. interpret via intensions also concept and role expressions, more in the spirit of original HiLog. Homola et al. [11,10] proposed a HiLog-style higher-order DL that is strictly typed, i.e., the entities of different types (and orders) are disjoint. On the other hand, works of Pan et al. [18,19] and Motz et al. [17] rely on the stronger, Henkin's truly higher-order semantics [9] and are strictly typed. While Homola et al. and Pan et al. sort the higher order entities beforehand, Motz et al. introduce a new kind of axioms ($=_{\mathsf{m}}$) by which one identifies a concept with an individual when needed. Distinctly, Glimm et al. [4] propose an axiomatization of metamodelling within a DL, relying on companion individuals for all regular concepts, which can then be asserted into meta-concepts as needed. This work supports only the second order, and the orders are separated.

Besides, in OWL 2 there is the option of punning [5], i.e., using the same name in different contexts, which is essentially equivalent to the π-semantics investigated by Motik [16] who showed that there is no semantic relation between the contextually different uses. Finally, OWL Full [20] allows almost arbitrary metamodelling, including restrictions on the language constructs, such as instantiation, subconcept relation, or even restrictions themselves. OWL Full is undecidable and Motik [16] showed that it remains so even if the underlying DL is downgraded to $\mathcal{ALC}$. Interestingly enough, Glimm et al. [4] axiomatized a metamodel of instantiation and, to some extent, also of subsumption in $\mathcal{SROIQ}$.

We propose a new variant of higher-order DL for metamodelling with HiLog-style semantics, which combines and unifies some of the approaches listed above. We argue for, and implement the following features:

- Basic level of separation between individuals, concepts, and roles is needed, to avoid confusion, and to provide basic sanity checks for the modeller. Therefore individuals, corresponding to particular objects in the world, have no extensions. Concepts have only concept extensions and roles have only role extensions.
- Both promiscuous and strictly typed concept and role extensions coexist in the framework and may be freely chosen by the modeller. This allows, e.g., to require that the concept Person has individuals as instances only (it does not make sense for people to have extensions), while in the same ontology the concept Deprecated can have any instances (anything may become deprecated).
- In most of the former proposals, meta-levels can be viewed as additional layers in the ontology. Constraints may be modelled inside each layer but they cannot equally penetrate to different layers and create complex inter-layer dependencies. This is possible by modelling with instantiation. In our approach, a fixed role instanceOf connects instances with the concepts they belong to (it has equal semantics to rdf:type) and it may be freely modelled with. We are able to require, say, that for each species defined by von Linné there is specimen (instance of the species) in the British Museum.

The paper starts with brief preliminaries in Sect. 2. Section 3.1 builds $\mathcal{HI}(\mathcal{L})$, a promiscuous higher-order variant of $\mathcal{L}$ with instantiation modelling in which roles cannot be classified. Section 3.2 builds $\mathcal{HIR}(\mathcal{L})$ which adds the option to classify roles as well. Consequently, Sect. 3.3 shows how strict types are axiomatized in these languages, and Sect. 3.4 reviews the basic properties of these logics. Conclusions follow in Sect. 4.

Table 1. Syntax and Semantics of $\mathcal{SROIQ}$

Cons'tor	Syntax	Semantics		Syn.	Semantics
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	inv. role	R^-	$\{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}$
exist. restr.	$\exists R.C$	$\{x \mid \exists y \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	univ. role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
card. restr.	$\geqslant n R.C$	$\{x \mid \sharp\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$	chain (w)	$S \cdot Q$	$S^{\mathcal{I}} \circ Q^{\mathcal{I}}$
self restr.	$\exists R.\mathsf{Self}$	$\{x \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$			

Axiom	Syntax	Semantics		Syntax	Semantics
conc. incl. (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	concept assert.	$a\colon C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role incl. (RIA)	$w \sqsubseteq R$	$w^{\mathcal{I}} \subseteq R^{\mathcal{I}}$	role assertion	$a, b\colon R$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
reflexivity assert.	$\mathrm{Ref}(R)$	$R^{\mathcal{I}}$ is reflexive	neg. role assert.	$a, b\colon \neg R$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \notin R^{\mathcal{I}}$
role disjointness	$\mathrm{Dis}(P, R)$	$P^{\mathcal{I}} \cap R^{\mathcal{I}} = \emptyset$			

2 Preliminaries

2.1 $\mathcal{SROIQ}$ Description Logic

$\mathcal{SROIQ}$ DL [12] currently constitutes a generally accepted standard for a very expressive description logic. It uses a DL vocabulary $N = N_{\mathrm{C}} \uplus N_{\mathrm{R}} \uplus N_{\mathrm{I}}$ with countable sets of *atomic concepts* (N_{C}), *atomic roles* (N_{R}), and *individuals* (N_{I}). *Complex concepts* (*complex roles*) are defined as the smallest sets containing all concepts and roles that can be inductively constructed using the concept (role) constructors[1] in Table 1, where C, D are concepts, P, R are atomic or inverse roles, S, Q are any roles (including role chains), a, b are individuals, and n is a positive integer. A $\mathcal{SROIQ}$ *knowledge base* (KB) is a finite set $\mathcal{K}$ of *axioms* of the types shown in Table 1.

A DL *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$ where $\Delta^{\mathcal{I}} \neq \emptyset$ and $\cdot^{\mathcal{I}}$ is a function such that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for all $a \in N_{\mathrm{I}}$, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in N_{\mathrm{C}}$, and $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $R \in N_{\mathrm{R}}$. Interpretation of complex concepts and roles is inductively defined as given in Table 1. An axiom φ is *satisfied* by $\mathcal{I}$ (denoted $\mathcal{I} \models \varphi$) if $\mathcal{I}$ satisfies the respective semantic constraint listed in Table 1, and $\mathcal{I}$ is a *model* of $\mathcal{K}$ (denoted $\mathcal{I} \models \mathcal{K}$) if it satisfies all axioms of $\mathcal{K}$. A concept C is *satisfiable w.r.t.* $\mathcal{K}$ if there is a model of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. A formula φ is *entailed by* $\mathcal{K}$ (denoted $\mathcal{K} \models \varphi$) if $\mathcal{I} \models \varphi$ for all models $\mathcal{I}$ of $\mathcal{K}$. The two reasoning tasks of *concept satisfiability* and *concept subsumption* entailment are inter-reducible [12]. Reasoning in $\mathcal{SROIQ}$ is decidable if further syntactic restrictions are applied [12], which are based on the following notions.

A role R is *simple* if (a) it is atomic and it does not occur on the right-hand side of any RIA; (b) it is an inverse of a simple role S; or (c) it occurs on the right-hand side of a RIA, but each such RIA is of the form $S \sqsubseteq R$ where S is a simple role.

A regular order on roles $\prec$ is a strict partial order (transitive and irreflexive binary relation) on roles (including inverse roles) such that $S \prec R$ iff $S^- \prec R$ for any roles S, R. Given a regular order on roles $\prec$, a RIA is $\prec$-*regular* if it has one of the following forms: (a) $R \cdot R \sqsubseteq R$; (b) $R^- \sqsubseteq R$; (c) $S_1 \cdot \cdots \cdot S_n \sqsubseteq R$ and $S_i \prec R$ for $1 \leq i \leq n$; (d) $R \cdot S_1 \cdot \cdots \cdot S_n \sqsubseteq R$ and $S_i \prec R$ for $1 \leq i \leq n$; (e) $S_1 \cdot \cdots \cdot S_n \cdot R \sqsubseteq R$ and $S_i \prec R$ for $1 \leq i \leq n$.

[1] There are additional $\mathcal{SROIQ}$ constructors and axioms all of which, including individual equality (=), are reducible to the core constructs listed in Table 1.

Table 2. Syntax and ν-semantics of $\mathcal{SHOIQ}mm$

Cons'tor	Syntax	Semantics		Syn.	Semantics
intersection	$D_1 \sqcap D_2$	$C^{\mathcal{I}}(D_1) \cap C^{\mathcal{I}}(D_2)$	complement	$\neg D$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}(D)$
exist. restr.	$\exists S.D$	$\{ x \mid \exists y \langle x, y\rangle \in R^{\mathcal{I}}(S) \wedge y \in C^{\mathcal{I}}(D) \}$	nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
card. restr.	$\geqslant n S.D$	$\{ x \mid \sharp\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}(S) \wedge y \in C^{\mathcal{I}}(D)\} \geq n \}$			

Axiom	Syntax	Semantics		Syntax	Semantics
conc. incl. (GCI)	$D_1 \sqsubseteq D_2$	$C^{\mathcal{I}}(D_1) \subseteq C^{\mathcal{I}}(D_2)$			
role incl. (RIA)	$S_1 \sqsubseteq_{\mathrm{R}} S_2$	$R^{\mathcal{I}}(S_1) \subseteq R^{\mathcal{I}}(S_2)$	transitivity	$S_1 \cdot S_1 \sqsubseteq_{\mathrm{R}} S_1$	$R^{\mathcal{I}}(S_1)$ is transitive
concept assertion	$a\colon D_1$	$a^{\mathcal{I}} \in C^{\mathcal{I}}(D_1)$	role assert.	$a, b\colon S_1$	$\langle a^{\mathcal{I}}, b^{I}\rangle \in R^{\mathcal{I}}(S_1)$

Finally, the restrictions placed on $\mathcal{SROIQ}$: (a) only simple roles may appear in cardinality restrictions; self restriction, and role disjointness axioms; (b) there is a regular order of roles $\prec$ such that all RIAs in the KB are $\prec$-regular; (c) the universal role may not appear in any RIA, role reflexivity, nor disjointness axiom.

Under these restrictions, satisfiability (and thus subsumption) is decidable [12] and it is N2ExpTime complete [14].

2.2 DLs with Metamodelling and the ν-Semantics

Motik [16] has proposed DLs for metamodelling with HiLog-based ν-semantics. In DL applications of HiLog semantics [2], names (and sometimes expressions) are first assigned domain objects called *intensions*, which, in turn, are assigned *extensions*: sets of intensions for concepts, or sets of pairs of intensions for roles. When treated as a concept instance or a role actor, the semantics of a name is its intension. When treated as a concept or a role, the extension of the name's intension is considered.

Motik's DL $\mathcal{SHOIQ}$ *with metamodelling* (abbreviated to $\mathcal{SHOIQ}mm$) uses a *set of names* $N = N_a \cup \{ n^- \mid n \in N_a \}$ where N_a is a *set of atomic names*. The set of *concepts* is the smallest superset of N containing all concepts that can be inductively constructed using the constructors in Table 2, where D_1 and D_2 are concepts, $a, S \in N$, and n is a non-negative integer. A $\mathcal{SHOIQ}mm$ *knowledge base* is a set $\mathcal{K}$ of axioms of the forms listed in Table 2 where $S_1, S_2, a, b \in N$, and D_1, D_2 are concepts. *Simple names* are defined as $\mathcal{SROIQ}$ simple roles (using $\sqsubseteq_{\mathrm{R}}$ instead of $\sqsubseteq$). A KB $\mathcal{K}$ employs the *unique role assumption* (URA) if $(\{S_1\} \sqsubseteq \neg\{S_2\}) \in \mathcal{K}$ for each two distinct names S_1 and S_2 occurring as roles in $\mathcal{K}$.

A ν*-interpretation* is a quadruple $\mathcal{I} = \langle \Delta^{I}, \cdot^{\mathcal{I}}, C^{\mathcal{I}}, R^{\mathcal{I}}\rangle$ where $\Delta^{\mathcal{I}} \neq \emptyset$ is a *domain set*, $\cdot^{\mathcal{I}} \colon N \to \Delta^{\mathcal{I}}$ is a *name interpretation function*, $C^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \to 2^{\Delta^{\mathcal{I}}}$ is an *atomic concept extension function* and $R^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \to 2^{\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}}$ is a *role extension function*. The extension of $C^{\mathcal{I}}$ to concepts and ν*-satisfaction* of axioms ($\mathcal{I} \models \varphi$) are specified in Table 2. The notions of ν-model and ν-entailment are defined analogously to their $\mathcal{SROIQ}$ counterparts. A concept D is ν*-satisfiable w.r.t.* $\mathcal{K}$ if $C^{\mathcal{I}}(D) \neq \emptyset$ in some ν-model $\mathcal{I}$ of $\mathcal{K}$.

Decidability of ν-satisfiability in NExpTime was proved by Motik [16] if (a) only simple names appear in cardinality restrictions, and (b) the KB employs URA, or contains no nominals and no cardinality restrictions.

Table 3. Syntax and semantics of $\mathcal{HI}(\mathcal{SROIQ})$ and $\mathcal{HIR}(\mathcal{SHOIQ})$

Syntax	Extensions in $\mathcal{HI}(\mathcal{SROIQ})$	Extensions in $\mathcal{HIR}(\mathcal{SHOIQ})$
R_0	$R_0^{\mathcal{IE}}$	$R_0^{\mathcal{IE}}$
R^-	$\{ (y,x) \mid (x,y) \in R^{\mathcal{E}} \}$	$\{ (y,x) \mid (x,y) \in R^{\mathcal{E}} \}$
U	$(\Delta_{\mathrm{I}}^{\mathcal{I}} \uplus \Delta_{\mathrm{C}}^{\mathcal{I}}) \times (\Delta_{\mathrm{I}}^{\mathcal{I}} \uplus \Delta_{\mathrm{C}}^{\mathcal{I}})$	–
$S \cdot Q$	$S^{\mathcal{E}} \circ Q^{\mathcal{E}}$	$R^{\mathcal{E}} \circ R^{\mathcal{E}}$ for $S = R, Q = R$
instanceOf	$\{ (x,y) \mid x \in \Delta_{\mathrm{I}}^{\mathcal{I}} \uplus \Delta_{\mathrm{C}}^{\mathcal{I}} \wedge y \in \Delta_{\mathrm{C}}^{\mathcal{I}} \wedge x \in y^{\mathcal{E}} \}$	$\{ (x,y) \mid x \in \Delta^{\mathcal{I}} \wedge y \in \Delta_{\mathrm{C}}^{\mathcal{I}} \wedge x \in y^{\mathcal{E}} \}$
A	$A^{\mathcal{IE}}$	$A^{\mathcal{IE}}$
$\neg C$	$(\Delta_{\mathrm{I}}^{\mathcal{I}} \uplus \Delta_{\mathrm{C}}^{\mathcal{I}}) \setminus C^{\mathcal{E}}$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{E}}$
$C \sqcap D$	$C^{\mathcal{E}} \cap D^{\mathcal{E}}$	$C^{\mathcal{E}} \cap D^{\mathcal{E}}$
$\{B\}$	$\{B^{\mathcal{I}}\}$	$\{B^{\mathcal{I}}\}$
$\exists R.C$	$\{ x \mid \exists y.(x,y) \in R^{\mathcal{E}} \wedge y \in C^{\mathcal{E}} \}$	$\{ x \mid \exists y.(x,y) \in R^{\mathcal{E}} \wedge y \in C^{\mathcal{E}} \}$
$\geqslant n R.C$	$\{ x \mid \#\{ y \mid (x,y) \in R^{\mathcal{E}} \wedge y \in C^{\mathcal{E}} \} \geq n \}$	$\{ x \mid \#\{ y \mid (x,y) \in R^{\mathcal{E}} \wedge y \in C^{\mathcal{E}} \} \geq n \}$
$\exists R.\mathsf{Self}$	$\{ x \mid (x,x) \in R^{\mathcal{E}} \}$	–
$C \sqsubseteq D$	$C^{\mathcal{E}} \subseteq D^{\mathcal{E}}$	$C^{\mathcal{E}} \subseteq D^{\mathcal{E}}$
$B : C$	$B^{\mathcal{I}} \in C^{\mathcal{E}}$	$B^{\mathcal{I}} \in C^{\mathcal{E}}$
$B_1, B_2 : R$	$(B_1^{\mathcal{I}}, B_2^{\mathcal{I}}) \in R^{\mathcal{E}}$	$(B_1^{\mathcal{I}}, B_2^{\mathcal{I}}) \in R^{\mathcal{E}}$
$B_1, B_2 : \neg R$	$(B_1^{\mathcal{I}}, B_2^{\mathcal{I}}) \notin R^{\mathcal{E}}$	–
$w \sqsubseteq R$	$w^{\mathcal{E}} \subseteq R^{\mathcal{E}}$	$w^{\mathcal{E}} \subseteq R^{\mathcal{E}}$ for $w = P$ or $w = R \cdot R$
$\mathrm{Dis}(P,R)$	$P^{\mathcal{E}} \cap R^{\mathcal{E}} = \emptyset$	–

3 Higher-Order DLs with Promiscuous Concepts and Instantiation Modelling

The logics $\mathcal{HI}(\mathcal{L})$ and $\mathcal{HIR}(\mathcal{L})$ defined in this section for an underlying first-order DL $\mathcal{L}$ rely on HiLog-style semantics. They feature *promiscuous* higher-order concepts with different types of entities allowed as instances. In addition, the fixed instanceOf role metamodels the instantiation relation, and it is usable in axioms like any other role. The letter $\mathcal{H}$ stands for higher-order, $\mathcal{I}$ for instanceOf, and $\mathcal{R}$ for metamodelling of roles.

3.1 Promiscuous Concepts and Instance Modelling

We first define $\mathcal{HI}(\mathcal{SROIQ})$, a higher-order variant of $\mathcal{SROIQ}$ allowing individuals and concepts to be classified, and featuring the instanceOf role. Roles relate both individuals and concepts with one another, but they cannot be classified. $\mathcal{HI}(\mathcal{L})$ for a fragment $\mathcal{L}$ of $\mathcal{SROIQ}$ is the corresponding fragment of $\mathcal{HI}(\mathcal{SROIQ})$.

In some HiLog-based DLs [16,3], names designate intensions and are fully interchangeable as individuals, concepts, and roles, depending on their occurrence. We believe that such an approach is less suitable in the realm of DLs. The individual-concept-role distinction is fundamental from the ontological standpoint (see, e.g., [6, Pt. II], [7, Ch. 4, 6, 7], [21]) and dates back to Aristotle [1, 2a11, 6a37]. This distinction also saves users accustomed to first-order DLs from surprises, and provides basic sanity checks.

Let us now define the syntax of $\mathcal{HI}(\mathcal{SROIQ})$. Note that the instanceOf role can occur anywhere where any regular atomic role can.

Definition 1. *Let $N = N_\mathrm{I} \uplus N_\mathrm{C} \uplus N_\mathrm{R}$ be a DL vocabulary such that* $\mathsf{instanceOf} \in N_\mathrm{R}$. $\mathcal{HI}(\mathcal{SROIQ})$ role expressions *are inductively defined as the smallest set containing the expressions listed in Table 3 (upper part), where $R_0 \in N_\mathrm{R} \setminus \{\mathsf{instanceOf}, \mathsf{U}\}$, R is an atomic or inverse role, S and Q are role expressions.* $\mathcal{HI}(\mathcal{SROIQ})$ descriptions *are inductively defined as the smallest set containing the expressions listed in Table 3 (middle part), where $A \in N_\mathrm{C}$, $B \in N_\mathrm{I} \uplus N_\mathrm{C}$, C and D are descriptions, and R is an atomic or inverse role. A* $\mathcal{HI}(\mathcal{SROIQ})$ knowledge base $\mathcal{K}$ *is a finite set of* axioms *of the forms listed in Table 3 (bottom part), where $B, B_1, B_2 \in N_\mathrm{I} \uplus N_\mathrm{C}$, C and D are descriptions, P and R are atomic or inverse roles, and w is a role chain.*

$\mathcal{HI}(\mathcal{SROIQ})$ easily models the taxonomic example from the Introduction. Taxons are classified to meta-concepts of ranks ($\mathsf{Species}$, Genus, …), ranks to the meta-meta-concept Rank (1). Metaconcepts are axiomatized just as concepts (2).

$$\mathsf{G.\,camelopardalis} : \mathsf{Species} \qquad \mathsf{Giraffa} : \mathsf{Genus} \qquad \mathsf{Species} : \mathsf{Rank} \qquad \mathsf{Genus} : \mathsf{Rank} \tag{1}$$

$$\mathsf{Species} \sqcup \mathsf{Genus} \sqcup \cdots \sqsubseteq \mathsf{Taxon} \qquad \mathsf{Species} \sqcap \mathsf{Genus} \sqsubseteq \bot \tag{2}$$

Metaroles with concepts on one or both sides of the relationship allow specifying that a taxon or rank was $\mathsf{definedBy}$ a person (3), and that one species is an evolutionary $\mathsf{successorOf}$ of another (4). We can then express complex meta-concepts such as $\mathsf{LinneanSpecies}$ (5).

$$\exists \mathsf{definedBy}.\top \sqsubseteq \mathsf{Taxon} \sqcup \mathsf{Rank} \quad \top \sqsubseteq \forall \mathsf{definedBy}.\mathsf{Person} \quad \mathsf{Giraffa, M.\,T.\,Brünnich} : \mathsf{definedBy} \tag{3}$$

$$\exists \mathsf{successorOf}.\top \sqsubseteq \mathsf{Species} \qquad \top \sqsubseteq \forall \mathsf{successorOf}.\mathsf{Species} \tag{4}$$

$$\mathsf{LinneanSpecies} \equiv \mathsf{Species} \sqcap \exists \mathsf{definedBy}.\{\mathsf{vonLinné}\} \tag{5}$$

First-order modelling still works as in $\mathcal{SROIQ}$: Individual organisms are classified to taxons and particular species are subsumed by their respective genera (6). Roles record that a specimen (a studied example individual of a species) was $\mathsf{describedBy}$ a person, and is $\mathsf{locatedIn}$ a museum (7).

$$\mathsf{Zarafa} : \mathsf{G.\,camelopardalis} \quad \mathsf{G.\,camelopardalis} \sqsubseteq \mathsf{Giraffa} \quad \mathsf{Specimen} \sqsubseteq \mathsf{Organism} \tag{6}$$

$$\exists \mathsf{describedBy}.\top \sqcup \exists \mathsf{locatedIn}.\top \sqsubseteq \mathsf{Specimen} \quad \top \sqsubseteq \forall \mathsf{describedBy}.\mathsf{Person} \sqcap \forall \mathsf{locatedIn}.\mathsf{Museum} \tag{7}$$

$\mathcal{HI}(\mathcal{SROIQ})$ employs HiLog-based semantics: each entity is denoted to a domain element (the intension) using the intension function $\cdot^\mathcal{I}$. The intensions for individuals, concepts, and roles are disjoint. Intensions of concepts (roles) are assigned concept (role) extensions. Only one extension function $\cdot^\mathcal{E}$ is therefore needed, unlike in the ν-semantics. The $\mathsf{instanceOf}$ role has fixed semantics: it connects an instance with the intension of each concept it belongs to – i.e., just like $\mathsf{rdf{:}type}$ in RDF [8].

Definition 2. *An* $\mathcal{HI}$-interpretation *of a DL vocabulary N with* $\mathsf{instanceOf} \in N_\mathrm{R}$ *is a triple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}, \cdot^\mathcal{E})$ such that:*

1. $\Delta^\mathcal{I} = \Delta^\mathcal{I}_\mathrm{I} \uplus \Delta^\mathcal{I}_\mathrm{C} \uplus \Delta^\mathcal{I}_\mathrm{R}$ *where $\Delta^\mathcal{I}_\mathrm{I}, \Delta^\mathcal{I}_\mathrm{C}, \Delta^\mathcal{I}_\mathrm{R}$ are pairwise disjoint,*
2. $a^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{I}$ *for each $a \in N_\mathrm{I}$, $A^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{C}$ for each $A \in N_\mathrm{C}$, $R^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{R}$ for each $R \in N_\mathrm{R}$,*
3. $c^\mathcal{E} \subseteq \Delta^\mathcal{I}_\mathrm{I} \uplus \Delta^\mathcal{I}_\mathrm{C}$ *for each $c \in \Delta^\mathcal{I}_\mathrm{C}$, $r^\mathcal{E} \subseteq (\Delta^\mathcal{I}_\mathrm{I} \uplus \Delta^\mathcal{I}_\mathrm{C}) \times (\Delta^\mathcal{I}_\mathrm{I} \uplus \Delta^\mathcal{I}_\mathrm{C})$ for each $r \in \Delta^\mathcal{I}_\mathrm{R}$.*

Extensions of role expressions $R^\mathcal{E}$ and of descriptions $C^\mathcal{E}$ are inductively defined according to Table 3.

The semantics of axioms (and of nominals above) is defined so that when a name is treated as a concept instance or a role actor, its intension is considered.

Definition 3. *An axiom φ is* satisfied *by a $\mathcal{HI}$-interpretation $\mathcal{I}$ ($\mathcal{I} \models \varphi$) if $\mathcal{I}$ satisfies the respective semantic constraints from the lower part of Table 3. A $\mathcal{HI}$-interpretation $\mathcal{I}$ is a* model *of $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) if $\mathcal{I}$ satisfies every axiom $\varphi \in \mathcal{K}$. A concept C is* satisfiable w.r.t. *$\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. An axiom φ is* entailed *by $\mathcal{K}$ ($\mathcal{K} \models \varphi$) if $\mathcal{I} \models \varphi$ holds for each $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$.*

The semantics of instanceOf should now be apparent. The fixed interpretation of this role allows to "move accross" meta-layers in modelling: Restrictions on instanceOf can select instances of concepts satisfying various meta-criteria, e.g., specimens of Linnean species described by someone else than von Linné (8). Conversely, restrictions on $\mathsf{instanceOf}^-$ select concepts whose instances satisfy complex criteria, e.g., species with specimens located in the British Museum. Liberal treatment of the instanceOf role allows creating its subroles, e.g., hasType assigning a prototypical specimen to each species (10), and using them in number restrictions, e.g., to assert that each species has exactly one holotype (the "most notable" specimen) and it is located in a major museum (11). While we could have created the meta-role hasType anyway, without relating it to instanceOf we could not assure that it connects each species with one of its instances.

$$\mathsf{Specimen} \sqcap \exists \mathsf{instanceOf}.(\mathsf{Species} \sqcap \exists \mathsf{definedBy}.\{\mathsf{vonLinné}\}) \sqcap \exists \mathsf{describedBy}.\neg\{\mathsf{vonLinné}\} \quad (8)$$

$$\mathsf{Species} \sqcap \exists \mathsf{instanceOf}^-.(\mathsf{Specimen} \sqcap \exists \mathsf{locatedIn}.\{\mathsf{britishMusem}\}) \quad (9)$$

$$\mathsf{hasType} \sqsubseteq \mathsf{instanceOf}^- \qquad \exists \mathsf{hasType}.\top \sqsubseteq \mathsf{Species} \qquad \top \sqsubseteq \forall \mathsf{hasType}.\mathsf{Specimen} \quad (10)$$

$$\mathsf{Species} \sqsubseteq (\leqslant 1 \mathsf{hasType}.\mathsf{Holotype}) \sqcap = 1 \mathsf{hasType}.(\mathsf{Holotype} \sqcap \exists \mathsf{locatedIn}.\mathsf{MajorMusem}) \quad (11)$$

We will now show how $\mathcal{HI}(\mathcal{SROIQ})$ can be reduced to first-order $\mathcal{SROIQ}$. The reduction, fully defined below, is based on ideas by Glimm et al. [4]. For each concept A, a new individual name c_A is introduced to represent A's intension. These new names are instances of a new concept $\top_{\mathrm{C}}$ of concept intensions. The relationship between the extension A and the intension c_A is expressed through the instanceOf role in the InstSync axioms. In the reduced knowledge base, instanceOf is an ordinary, axiomatized role.

Definition 4 (First-Order Reduction). *A DL vocabulary N with* $\mathsf{instanceOf} \in N_{\mathrm{R}}$ *is reduced into a DL vocabulary $N^1 := (N^1_{\mathrm{C}}, N^1_{\mathrm{R}}, N^1_{\mathrm{I}})$ where $N^1_{\mathrm{C}} = N_{\mathrm{C}} \uplus \{\top_{\mathrm{C}}\}$, $N^1_{\mathrm{R}} = N_{\mathrm{R}}$, $N^1_{\mathrm{I}} = N_{\mathrm{I}} \uplus \{\mathsf{c}_A \mid A \in N_{\mathrm{C}}\}$ for fresh names $\top_{\mathrm{C}}$ and c_A for all $A \in N_{\mathrm{C}}$. A given $\mathcal{HI}(\mathcal{SROIQ})$ KB $\mathcal{K}$ in N is reduced into a $\mathcal{SROIQ}$ KB $\mathcal{K}^1 :=$ Bound($\mathcal{K}$) $\cup$ InstSync($\mathcal{K}$) $\cup$ Typing($\mathcal{K}$) in N^1 where* Bound($\mathcal{K}$) *is obtained from $\mathcal{K}$ by replacing each occurence of $A \in N_{\mathrm{C}}$ in a nominal or in the left-hand side of a concept or (negative) role assertion by c_A.* InstSync($\mathcal{K}$) *consists of axioms $A \equiv \exists \mathsf{instanceOf}.\{\mathsf{c}_A\}$ for all $A \in N_{\mathrm{C}}$.* Typing($\mathcal{K}$) *consists of axioms $\top \sqsubseteq \forall \mathsf{instanceOf}.\top_{\mathrm{C}}$, $a\colon \neg\top_{\mathrm{C}}$ and $\mathsf{c}_A\colon \top_{\mathrm{C}}$ for all $a \in N_{\mathrm{I}}$ and $A \in N_{\mathrm{C}}$.*

The following theorem asserts that $\mathcal{K}^1$ is just as strong as $\mathcal{K}$. The more involved part of its proof is finding a $\mathcal{HI}(\mathcal{SROIQ})$ model $\mathcal{I}$ of $\mathcal{K}$ for a first-order model $\mathcal{J}$ of $\mathcal{K}^1$. We define the set of concept intensions $\Delta^{\mathcal{I}}_{\mathrm{C}}$ as $\top^{\mathcal{J}}_{\mathrm{C}}$, and let the intension of each atomic concept A be $\mathsf{c}^{\mathcal{J}}_A$. We then define for each concept intension $c \in \Delta^{\mathcal{I}}_{\mathrm{C}}$ the extension $c^{\mathcal{E}}$ as the set of all xs related to it by $\mathsf{instanceOf}^{\mathcal{J}}$. This ensures $\mathsf{instanceOf}^{\mathcal{E}} = \mathsf{instanceOf}^{\mathcal{J}}$. Moreover, since the interpretation of instanceOf is constrained by the InstSync axioms, extension of each atomic concept $A^{\mathcal{IE}}$ is equal to its first-order interpretation $A^{\mathcal{J}}$. A more detailed proof of the theorem can be found in the extended version of this paper [15].

Theorem 1. *For any $\mathcal{HI}(\mathcal{SROIQ})$ KB $\mathcal{K}$ and any axiom φ in a common vocabulary N, we have $\mathcal{K} \models \varphi$ iff $\mathcal{K}^1 \models \mathrm{Bound}(\varphi)$.*

Observe that $\mathcal{K}^1$ is linear in the size (string-length) of $\mathcal{K}$ (assuming N consists exactly of all symbols occurring in $\mathcal{K}$). If $\mathcal{K}$ satisfies, for all roles including instanceOf, the $\mathcal{SROIQ}$ restrictions (cf. Sect. 2.1), so does $\mathcal{K}^1$.

Corollary 1. *Concept satisfiability and subsumption in $\mathcal{HI}(\mathcal{SROIQ})$ conforming to $\mathcal{SROIQ}$ restrictions on roles including* instanceOf *are decidable in N2ExpTime.*

In general, $\mathcal{HI}(\mathcal{L})$ reduces to $\mathcal{LO}$ if the DL $\mathcal{L}$ admits GCIs, existential restriction, and complement. The decidability and complexity of standard reasoning tasks for $\mathcal{HI}(\mathcal{L})$ are then the same as for $\mathcal{LO}$.

A natural next step after metamodelling instantiation is metamodelling subsumption. For a singleton meta-concept C, the meta-concept $\forall \mathsf{instanceOf}^-.\exists \mathsf{instanceOf}.C$, due to Glimm et al. [4], classifies subconcepts of the single instance of C. We can thus express that all species of genus Giraffa except G. camelopardalis are extinct (12). Glimm et al. [4] axiomatized a subClassOf role over atomic concepts, but, unlike instanceOf, it does not extend to unnamed concept intensions. Metamodelling of subsumption with the intended semantics $\{(x, y) \mid x, y \in \Delta^{\mathcal{I}}_{\mathrm{C}} \wedge x^{\mathcal{E}} \subseteq y^{\mathcal{E}}\}$ is a part of our near-future work.

$$\mathsf{Species} \sqcap \forall \mathsf{instanceOf}^-.\exists \mathsf{instanceOf}.\{\mathsf{Giraffa}\} \sqcap \neg\{\mathsf{G.\,camelopardalis}\} \sqsubseteq \mathsf{Extinct} \qquad (12)$$

3.2 Classification of Roles

We now define another variant of higher-order DL with a notable addition: allowing concepts to contain and roles to connect also atomic role names. We define the logic $\mathcal{HIR}(\mathcal{SHOIQ})$, on top of the $\mathcal{SHOIQ}$ DL [13] disallowing some $\mathcal{SROIQ}$ constructs (we say that an expression is allowed if its extension is defined in the right column of Table 3). $\mathcal{HIR}(\mathcal{L})$ for a fragment $\mathcal{L}$ of $\mathcal{SHOIQ}$ is the corresponding fragment of $\mathcal{HIR}(\mathcal{SHOIQ})$. $\mathcal{HIR}$ uses the same vocabulary as $\mathcal{HI}$.

The most notable syntactic difference for $\mathcal{HIR}(\mathcal{L})$ compared to $\mathcal{HI}(\mathcal{L})$ is the option to use atomic roles as individuals. The full syntax is as follows.

Definition 5. $\mathcal{HIR}(\mathcal{SHOIQ})$ role expressions *are inductively defined as the smallest set containing the allowed expressions listed in the upper part of Table 3, where $R_0 \in N_{\mathrm{R}} \setminus \{\mathsf{instanceOf}\}$, R is an atomic or inverse role.* $\mathcal{HIR}(\mathcal{SHOIQ})$ descriptions *are inductively defined as the smallest set containing the allowed expressions listed in the middle part of Table 3, where $A \in N_{\mathrm{C}}$, $B \in N$, C and D are descriptions and R is an atomic or inverse role. A* $\mathcal{HIR}(\mathcal{SHOIQ})$ knowledge base $\mathcal{K}$ *is a finite set of allowed axioms of the forms listed in the bottom part of Table 3, where B, B_1, $B_2 \in N$, C and D are descriptions, P and R are atomic or inverse roles.*

In contrast to $\mathcal{HI}$, $\mathcal{HIR}$ extensions range over all intensions, and hence concepts in $\mathcal{HIR}(\mathcal{L})$ are fully promiscuous, though about each instance we are able to say if it is an individual, concept, or role. In many cases such promiscuity is not desired; we will later learn how to constrain it if needed. However, there are certain concepts such as Deprecated for which it makes sense. Concepts, but as well roles, and even individuals may become deprecated if they are replaced by new names or more refined versions. In

the taxonomy domain, taxa, ranks, end even specimens may become deprecated (e.g., when they are invalidated by further studies). A role with a truly promiscuous domain is, e.g., definedBy; as it is applicable on both taxa and ranks.

Definition 6. *An $\mathcal{HIR}$-*interpretation *of a DL vocabulary N with* instanceOf $\in N_\mathrm{R}$ *is a triple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}, \cdot^\mathcal{E})$ such that*

1. $\Delta^\mathcal{I} = \Delta^\mathcal{I}_\mathrm{I} \uplus \Delta^\mathcal{I}_\mathrm{C} \uplus \Delta^\mathcal{I}_\mathrm{R}$ *(all three sets pairwise disjoint),*
2. $a^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{I}$ *for each* $a \in N_\mathrm{I}$, $A^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{C}$ *for each* $A \in N_\mathrm{C}$, $R^\mathcal{I} \in \Delta^\mathcal{I}_\mathrm{R}$ *for each* $R \in N_\mathrm{R}$,
3. $c^\mathcal{E} \subseteq \Delta^\mathcal{I}$ *for each* $c \in \Delta^\mathcal{I}_\mathrm{C}$, $r^\mathcal{E} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ *for each* $r \in \Delta^\mathcal{I}_\mathrm{R}$,

and the extensions of role expressions and complex descriptions are inductively defined according to Table 3.

Satisfiability, model, and entailment are defined as for $\mathcal{HI}(\mathcal{SROIQ})$. We will now reduce $\mathcal{HIR}(\mathcal{SHOIQ})$ to $\mathcal{SHOIQ}mm$ under ν-semantics. While there are similarities with the previous first-order reduction, the target is now already a HiLog-based higher-order logic. The core idea is to separate concepts, individuals, and roles which are not distinguished by $\mathcal{SHOIQ}mm$. We introduce $\top_\mathrm{I}$, $\top_\mathrm{C}$ and $\top_\mathrm{R}$, representing disjoint top-concepts for individuals, concepts, and roles (their union being $\top'$). The instanceOf role is reduced into an ordinary role with domain $\top'$ and range $\top_\mathrm{C}$, axiomatized through InstSync axioms. As $\mathcal{SHOIQ}mm$ uses a set of names including an inverse name n^- for each atomic name n, we also introduce $\top^-$ to handle all inverse names separately.

Definition 7 (Reduction to $\mathcal{SHOIQ}mm$ with ν-semantics). *A DL vocabulary N with* instanceOf $\in N_\mathrm{R}$ *is reduced into the set of atomic names $N^\nu_a = N_\mathrm{I} \uplus N_\mathrm{C} \uplus N_\mathrm{R} \uplus \{\top_\mathrm{C}, \top_\mathrm{R}, \top_\mathrm{I}, \top', \top^-\}$ for fresh names $\top_\mathrm{C}$, $\top_\mathrm{R}$, $\top_\mathrm{I}$, $\top^-$, and $\top'$.*

$\mathcal{HIR}(\mathcal{SHOIQ})$ KB $\mathcal{K}$ in N is reduced into a $\mathcal{SHOIQ}mm$ KB $\mathcal{K}^\nu$ in N^ν as follows: $\mathcal{K}^\nu =$ Bound$(\mathcal{K}) \cup$ InstSync$(\mathcal{K}) \cup$ Typing$(\mathcal{K})$*, where* InstSync$(\mathcal{K})$ *contains $A \equiv \exists$*instanceOf.$\{A\}$ *for all $A \in N_\mathrm{C}$,* Bound$(\mathcal{K})$ *contains for each $\varphi \in \mathcal{K}$ its transformed version where every occurrence of a description C in the form $\neg D$ is replaced by $\top' \sqcap C$ and every RIA $w \sqsubseteq S$ is replaced by $w \sqsubseteq_\mathrm{R} S$.* Typing$(\mathcal{K})$ *consists of axioms:*

1. $\neg(\{\text{instanceOf}, \top_\mathrm{C}, \top_\mathrm{R}, \top_\mathrm{I}, \top', \top^-\} \sqcup \top^-) \equiv \top' \equiv \top_\mathrm{C} \sqcup \top_\mathrm{R} \sqcup \top_\mathrm{I}$,
2. $\top_\mathrm{C} \sqcap \top_\mathrm{I} \sqsubseteq \bot$, $\top_\mathrm{C} \sqcap \top_\mathrm{R} \sqsubseteq \bot$, $\top_\mathrm{R} \sqcap \top_\mathrm{I} \sqsubseteq \bot$,
3. $\top \sqsubseteq \forall$instanceOf.$\top_\mathrm{C}$, $\exists$instanceOf.$\top \sqsubseteq \top'$,
4. *(i)* $a\colon \top_\mathrm{I}$ *for all* $a \in N_\mathrm{I}$, *(ii)* $A\colon \top_\mathrm{C}$, $A \sqsubseteq \top'$ *for all* $A \in N_\mathrm{C}$, *(iii)* $R\colon \top_\mathrm{R}$, $\top \sqsubseteq \forall R.\top'$, $\exists R.\top \sqsubseteq \top'$ *for all* $R \in N_\mathrm{R} \setminus \{\text{instanceOf}\}$, *(iv)* $n^-\colon \top^-$ *for all* $n \in N^\nu$.

The following theorem states that $\mathcal{K}^\nu$ is just as strong as $\mathcal{K}$. Its proof is similar to the proof of Theorem 1, and is included in the extended version [15].

Theorem 2. *For any $\mathcal{HIR}(\mathcal{SHOIQ})$ KB $\mathcal{K}$ and axiom φ over a common vocabulary N, we have $\mathcal{K} \models \varphi$ iff $\mathcal{K}^\nu \models$* Bound(φ).

The ν-reduction of a $\mathcal{HIR}(\mathcal{SHOIQ})$ KB $\mathcal{K}$ is linear in the size of $\mathcal{K}$. Simple roles are defined as for $\mathcal{SROIQ}$. URA is defined as for $\mathcal{SHOIQ}mm$, but only for names from N_R. We can now state the following corollary. Its assumptions suffice to satisfy the decidability conditions of ν-satisfiability of $\mathcal{K}^\nu$ from Sect. 2.2.

Table 4. Knowledge base with n types

$\top^{X(t)} \sqsubseteq \forall \mathsf{instanceOf}^{-}.\top^{X(t-1)}$ for each t s. t. $t > 0$	$a\colon \top^{\mathrm{I}(1)}$ for each $a \in N_{\mathrm{I}}$
$\top^{X(t)} \sqsubseteq \neg\top^{Y(u)}$ for each t, u s. t. $0 < t, u \leq n \wedge t \neq u$	
$\top^{\mathrm{IR}(1)} \equiv \top^{\mathrm{I}(1)} \sqcup \top^{\mathrm{R}(1)}$	$R\colon \top^{\mathrm{R}(1)}$ for each $R \in N_{\mathrm{R}}$

Corollary 2. *Let a $\mathcal{HIR}(\mathcal{SHOIQ})$ KB $\mathcal{K}$ be such that (a) only simple roles occur in cardinality restrictions, and (b) $\mathcal{K}$ employs URA, or it contains no nominals and no cardinality restrictions. Concept satisfiability and entailment in a $\mathcal{HIR}(\mathcal{SHOIQ})$ KB are then decidable in* NExpTime.

Note that the reduction theorem can be generalized to reducibility of $\mathcal{HIR}(\mathcal{L})$ to $\mathcal{LO}mm$, if $\mathcal{L}$ admits GCIs, existential restriction, and complement.

3.3 Type Hierarchy Strikes Back

In $\mathcal{HI}(\mathcal{L})$ ($\mathcal{HIR}(\mathcal{L})$), concepts are promiscuous – any individuals, concepts (and roles) may become their instances. In case selected concepts need to be strictly typed, this is axiomatized as follows:

Definition 8 (Typing framework). *Given $n \in \mathbb{N}$, a $\mathcal{HI}$* KB *with n* types *adds fresh concepts names $\top^{\mathrm{I}(i)}$ for each i such that $0 < i \leq n$, and contains axioms listed in the upper part of Table 4 for $X = Y = \mathrm{I}$. A $\mathcal{HIR}$* KB *with n* types *adds fresh concept names $\top^{X(i)}$ for each i, $0 < i \leq n$, and each $X \in \{\mathrm{I}, \mathrm{R}, \mathrm{IR}\}$, and contains axioms listed in the upper and lower part of Table 4 for all $X, Y \in \{\mathrm{I}, \mathrm{R}, \mathrm{IR}\}$.*

The $\top^{\mathrm{I}(1)}$ concept classifies precisely all individuals (similarly, $\top^{\mathrm{R}(1)}$ classifies precisely all roles and $\top^{\mathrm{IR}(1)}$ classifies precisely all individuals and roles), $\top^{\mathrm{I}(2)}$ classifies precisely all concepts with only individual instances (analogously, $\top^{\mathrm{R}(2)}$ for concepts of roles, and $\top^{\mathrm{IR}(2)}$ for mixed concepts), etc. We can thus assert some typing in our example:

$$\mathsf{Organism} \sqcup \mathsf{Person} \sqcup \mathsf{Museum} \sqsubseteq \top^{\mathrm{I}(1)} \quad \mathsf{Taxon} \sqsubseteq \top^{\mathrm{I}(2)} \quad \mathsf{Rank} \sqsubseteq \top^{\mathrm{I}(3)}$$

Typing is propagated to subconcepts and instances: $\mathsf{G.\,camelopardalis} \sqsubseteq \top^{\mathrm{I}(1)}$ and $\mathsf{Species} \sqsubseteq \top^{\mathrm{I}(2)}$ is now entailed, and so for other taxa and ranks. Domains and ranges of roles may be typed similarly, e.g.: $\exists \mathsf{successorOf}.\top \sqsubseteq \top^{\mathrm{I}(2)}$ and $\top \sqsubseteq \forall \mathsf{successorOf}.\top^{\mathrm{I}(2)}$. This, though, is also already entailed, since Species was already asserted as the domain and range, and it is already typed.

3.4 Some Properties of $\mathcal{HI}(\mathcal{SROIQ})$ and $\mathcal{HIR}(\mathcal{SHOIQ})$

$\mathcal{HI}(\mathcal{SROIQ})$ and $\mathcal{HIR}(\mathcal{SHOIQ})$ have the basic properties of HiLog-based logics: *intensional regularity* $\mathcal{K}, (X = Y) \models X \equiv Y$, and the lack of *extensionality* $\mathcal{K}, (X \equiv Y) \models X = Y$. Indeed, if $\mathcal{K} \models A = B$ for two concept names A and B, then $\mathcal{K} \models A \equiv B$, since in every model $A^{\mathcal{I}} = B^{\mathcal{I}}$, hence also $A^{\mathcal{IE}} = B^{\mathcal{IE}}$. Both logics are thus intensionally regular for concepts. This is a quite natural requirement for metamodelling. If we assert that an international and a Slovak name denote the same species ($\mathsf{G.\,camelopardalis} = \mathsf{Žirafa}$

štíhla), we expect their extensions to also be equal. $\mathcal{HIR}(\mathcal{SHOIQ})$ is intensionally regular also for roles, but $\mathcal{K}$ never entails $R = S$ under decidability conditions.

A model of a KB such that $\mathcal{K} \models A \equiv B$ can assign A and B distinct intensions $A^{\mathcal{I}} = a \neq b = B^{\mathcal{I}}$ with the same extension, e.g., $a^{\mathcal{E}} = b^{\mathcal{E}} = \{x\}$. Both our logics thus lack extensionality. This enables, e.g., deprecating an old binomial name of a species (e.g., Cervus camelopardalis) without deprecating its newer name (Giraffa camelopardalis) although they classify the same organisms (Giraffa c. ≡ Cervus c.), or modelling of single-species genera such as Sommeromys ≡ S. macrorhinos, where S. macrorhinos : Species and Sommeromys : Genus without contradicting Species ⊓ Genus ⊑ ⊥.

$\mathcal{HI}(\mathcal{SROIQ})$'s expressivity makes it vulnerable to Russel's paradox of naïve set theory. A concept of such concepts which are not instances of themselves is defined as Barber ≡ ¬∃instanceOf.Self. Take any $\mathcal{HI}(\mathcal{SROIQ})$ model $\mathcal{I}$ of $\mathcal{K}$, and let $b :=$ Barber$^{\mathcal{I}} \in \Delta_{\mathrm{C}}$, $B := b^{\mathcal{E}}$, and $S := \exists$instanceOf.Self $= \{ x \mid (x, x) \in \text{instanceOf}^{\mathcal{E}} \} = \{ x \mid x \in \Delta_{\mathrm{C}} \wedge x \in x^{\mathcal{E}} \}$. We have $B = (\Delta_{\mathrm{I}} \uplus \Delta_{\mathrm{C}}) \setminus S = \Delta_{\mathrm{I}} \uplus (\Delta_{\mathrm{C}} \setminus S) = \Delta_{\mathrm{I}} \uplus \{ x \mid x \in \Delta_{\mathrm{C}} \wedge x \notin x^{\mathcal{E}} \}$. Hence the contradiction: $b \in b^{\mathcal{E}}$ iff $b \notin b^{\mathcal{E}}$. Even though $\mathcal{SHOIQ}$ does not admit self-restriction, simpler contradictions involving instanceOf, e.g., x : Y and (x, Y) : ¬instanceOf, can be expressed in $\mathcal{HIR}(\mathcal{SHOIQ})$. However, these examples are actually not specific to $\mathcal{HI}$ or $\mathcal{HIR}$ logics, as they reduce to contradictory $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}mm$ KBs, respectively.

4 Conclusions

We have introduced a higher-order framework $\mathcal{HIR}(\mathcal{L})$ which enriches a DL $\mathcal{L}$ with promiscuous higher-order concepts, and makes the instantiation relation accessible to the modeller in the form of a fixly interpreted role named instanceOf, whose semantics is akin to rdf:type. We showed a number of examples illustrating how such constructs may be useful for metamodelling.

Our work is most closely related to that of Homola et al. [10] which is here extended by promiscuity and modelling with instantiation. The former approach is strictly typed; types are easily constructed in $\mathcal{HIR}(\mathcal{L})$ by axiomatization if needed, and may be used – but they are not strictly enforced. We base many of our constructions on Glimm et al. [4] who, however, do not enable orders higher than the second, meta-roles, nor promiscuity. They also do not provide any higher-order model-theoretic characterization, only an axiomatization in a regular DL. Such characterization is instrumental in showing that the logic has desired properties, which we have discussed in Sect. 3.4.

Computational support for logics up to $\mathcal{HIR}(\mathcal{SHOIQ})$ can be obtained via a reduction to Motik's extension of $\mathcal{SHOIQ}$ with metamodelling (under ν-semantics). For the weaker variant $\mathcal{HI}(\mathcal{L})$, which disallows classification of roles, $\mathcal{HI}(\mathcal{SROIQ})$ is fully reducible to regular $\mathcal{SROIQ}$ by adapting the reduction known from Glimm et al. The question whether the subsumption relation and further logical constructs of the underlying DL can be made accessible for metamodelling is an interesting open problem.

Acknowledgments. This work was funded by project VEGA 1/1333/12. Petra Kubincová received an extraordinary scholarship awarded by Faculty of Math., Physics, and Informatics. We thank Miroslav Vacura and Vojtěch Svátek for valuable suggestions.

References

1. Aristotle: The Categories, On Interpretation, Prior Analytics. The Loeb Classical Library, Harvard University Press, Cambridge, MA (1962), transl. by Tredennick, H., Cook, H.P.
2. Chen, W., Kifer, M., Warren, D.S.: A foundation for higher-order logic programming. J. Logic Programming 15(3), 187–230 (1993)
3. De Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: AAAI (2011)
4. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: ISWC (2010)
5. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 309–322 (2008)
6. Grossmann, R.: The Categorial Structure of the World. Indiana University Press, Bloomington (1983)
7. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Telematics Instituut, Enschede (2005)
8. Hayes, P.J., Patel-Schneider, P.F. (eds.): RDF 1.1 Semantics. W3C Recommendation (25 February 2014)
9. Henkin, L.: Completeness in the theory of types. J. Symbolic Logic 15, 81–91 (1950)
10. Homola, M., Kľuka, J., Svátek, V., Vacura, M.: Typed higher-order variant of $\mathcal{SROIQ}$ – why not? In: DL (2014)
11. Homola, M., Kľuka, J., Svátek, V., Vacura, M.: Towards typed higher-order description logics. In: DL (2013)
12. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: KR (2006)
13. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. Autom. Reasoning 39(3), 249–276 (2007)
14. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: KR. pp. 274–284 (2008)
15. Kubincová, P., Homola, M., Kľuka, J.: Towards expressive metamodelling with instantiation (extended version). Tech. Rep. TR-2015-045, Comenius University (2015), available online: `http://kedrigern.dcs.fmph.uniba.sk/reports/download.php?id=60`
16. Motik, B.: On the properties of metamodeling in OWL. J. Log. Comput 17(4), 617–637 (2007)
17. Motz, R., Rohrer, E., Severi, P.: Reasoning for $\mathcal{ALCQ}$ extended with a flexible metamodelling hierarchy. In: JIST 2014 (2014)
18. Pan, J.Z., Horrocks, I.: RDFS(FA): Connecting RDF(S) and OWL DL. IEEE Trans. Knowl. Data Eng. 19, 192–206 (2007)
19. Pan, J.Z., Horrocks, I., Schreiber, G.: OWL FA: A metamodeling extension of OWL DL. In: OWLED (2005)
20. Schneider, M. (ed.): OWL 2 Web Ontology Language RDF-Based Semantics. W3C Recommendation, 2nd edn. (11 December 2012)
21. Svátek, V., Homola, M., Kľuka, J., Vacura, M.: Metamodeling-based coherence checking of OWL vocabulary background models. In: OWLED (2013)
22. Ullmann, S.: Semantics: An introduction to the science of meaning. Barnes & Noble, Inc., New York (1962)

Mapping Analysis in Ontology-based Data Access: Algorithms and Complexity (Extended Abstract)

Domenico Lembo[1], José Mora[1], Riccardo Rosati[1],
Domenico Fabio Savo[1], Evgenij Thorstensen[2]

[1] DIAG, Sapienza Università di Roma
lastname@dis.uniroma1.it

[2] Dept. of Informatics, University of Oslo
evgenit@ifi.uio.no

1 Introduction

Ontology-based data access (OBDA) is a recent paradigm for accessing *data sources* through an *ontology* that acts as a conceptual, integrated view of the data, and declarative *mappings* that connect the ontology to the data sources [13, 6, 14, 5, 2] .

One important aspect in OBDA concerns the construction of a system specification, i.e., defining the ontology and the mappings over an existing set of data sources. Mappings are indeed the most complex part of an OBDA specification, since they have to capture the semantics of the data sources and express such semantics in terms of the ontology. The first experiences in the application of the OBDA framework in real-world scenarios (e.g., [2, 9]) have shown that the semantic distance between the conceptual and the data layer is often very large, because data sources are mostly application-oriented: this often makes the definition, debugging, and maintenance of mappings a hard and complex task. Such experiences have clearly shown the need of tools for supporting the management of mappings.

The recent work [11] has started providing a theoretical basis for mapping management support in OBDA, focusing on the formal analysis of mappings in ontology-based data access. In particular, the two most important semantic anomalies of mappings have been analyzed: inconsistency and redundancy. Roughly speaking, an inconsistent mapping for an ontology and a source schema is a specification that gives rise to logical contradictions with the ontology and/or the source schema. Then, a mapping $\mathcal{M}$ is redundant with respect to an OBDA specification if adding the mapping $\mathcal{M}$ to the specification does not change its semantics. The work presented in [11] has defined both a *local* notion of mapping inconsistency and redundancy, which focuses on single mapping assertions, and a *global* notion, where inconsistency and redundancy is considered with respect to a whole mapping specification (set of mapping assertions).

In this paper, we study the computational properties of verifying both local and global mapping inconsistency and redundancy in an OBDA specification. We consider a wide range of ontology languages that comprises the description logics underlying OWL 2 and all its profiles (OWL 2 EL, OWL 2 QL, and OWL 2 RL),[1] and examine mapping languages of different expressiveness (the so-called GAV and GLAV mappings [7]) over sources corresponding to relational databases. We provide algorithms and establish tight complexity bounds for the decision problems associated with both local and global mapping inconsistency and mapping redundancy, and for both combined complexity and TBox complexity (which only considers the size of the TBox).

[1] http://www.w3.org/TR/owl2-profiles/

The outcome of our analysis is twofold:

- in our framework, it is possible to define *modular techniques* that are able to reduce the analysis of mappings to the composition of standard reasoning tasks over the ontology (inconsistency, instance checking, query answering) and over the data sources (query answering and containment). This is a non-trivial result, because mappings are formulas combining both ontology and data source elements;
- the above forms of mapping analysis enjoy *nice computational properties*, in the sense that they are not harder than the above mentioned standard reasoning tasks over the ontology and the data sources (see Figure 1 and Figure 2) .

According to the above results, in our OBDA framework, the analysis of mappings is feasible for languages with nice computational properties, like the three OWL profiles.

2 Theoretical background

An OBDA specification is a triple $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{T}$ is a DL TBox, $\mathcal{S}$ is a source schema, and $\mathcal{M}$ is a mapping between the two. In this paper, we consider TBoxes specified through DLs that are the logical basis of the W3C standard OWL and of its profiles, i.e., $\mathcal{SROIQ}$ [8], which underpins OWL 2, *DL-Lite*$_R$ [4], which is the basis of OWL 2 QL, RL [10], a simplified version of OWL 2 RL, and $\mathcal{EL}_\bot$, a slight extension of the DL $\mathcal{EL}$ [3], which is the basis of OWL 2 EL. The source schema is assumed to be relational, and we consider both *simple schemas*, i.e., without integrity constraints, and *FD schemas*, i.e., simple schemas with functional dependencies [1]. The *mapping* is a set of assertions m of the form $\phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$, where $\phi(\boldsymbol{x})$, called the *body of* m, and $\psi(\boldsymbol{x})$, called the *head of* m, are conjunctive queries (CQs) over $\mathcal{S}$ and $\mathcal{T}$, respectively. We use $head(m)$ and $body(m)$ to denote the head and the body of m.

Mappings of the form above are called *GLAV*, and are the most expressive commonly studied mappings [12, 7]. Besides them, we refer also to *GLAV*$_{BE}$ mappings, which are GLAV mappings where $\psi(\boldsymbol{x})$ is a CQ with a bounded number of occurrences of existential variables, and to *GAV* mappings, which are GLAV mappings where $head(m)$ does not admit existential variables.

The semantics of an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is given in terms of first-order interpretations that satisfy both $\mathcal{T}$ and $\mathcal{M}$, given a source instance D legal for $\mathcal{S}$, i.e., an instance for $\mathcal{S}$ that satisfies the constraints of $\mathcal{S}$. We denote with $Mod(\mathcal{J}, D)$ the set of models of $\mathcal{J}$ w.r.t. D We also say that a mapping assertion m *is active on a source instance* D if the evaluation of the query $body(m)$ over D is non-empty. A mapping $\mathcal{M}$ is active on D if all its mapping assertions $m \in \mathcal{M}$ are active on D.

Below we recall the definitions given in [11] that formalize the mapping analysis services that we study in this paper. Given a TBox $\mathcal{T}$, a source schema $\mathcal{S}$, a mapping assertion m, a mapping $\mathcal{M}$, and an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, we have that

- m *is (locally) inconsistent for* $\langle \mathcal{T}, \mathcal{S} \rangle$ if m is head-inconsistent for $\mathcal{T}$, i.e., $\mathcal{T} \models \forall \boldsymbol{x}.(\neg\psi(\boldsymbol{x}))$, or m is body-inconsistent for $\mathcal{S}$, i.e., $\mathcal{S} \models \forall \boldsymbol{x}.(\neg\phi(\boldsymbol{x}))$.
- $\mathcal{M}$ *is globally inconsistent for* $\langle \mathcal{T}, \mathcal{S} \rangle$ if there does not exist a source instance D legal for $\mathcal{S}$ such that $\mathcal{M}$ is active on D and $Mod(\mathcal{J}, D) \neq \emptyset$.
- A mapping $\mathcal{M}'$ *is globally redundant for* $\mathcal{J}$ if, for every source instance D that is legal for $\mathcal{S}$, $Mod(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D) = Mod(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \cup \mathcal{M}' \rangle, D)$.

Local mapping redundancy is a special case of global mapping redundancy in which the mappings $\mathcal{M}$ and $\mathcal{M}'$ are both composed of a single assertion.

	GAV				GLAV			
task	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$
local inc.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NLOGSPACE	=P	=P	=N2EXPTIME
global inc.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NLOGSPACE	=P	=P	=N2EXPTIME
local red.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NP	=NP	=NP	open
global red.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NP	=NP	=NP	open

Fig. 1. TBox compl. of mapping inconsistency and redundancy (for both simple and FD schemas).

	GAV				GLAV$_{\text{BE}}$			
task	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$
local inc.	=NLOGSPACE (SI) =P (FD)	=P	=P	=N2EXPTIME	=NLOGSPACE (SI)* =P (FD)*	=P*	=P*	=N2EXPTIME*
global inc.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	=N2EXPTIME
local red.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	open
global red.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	open

Fig. 2. Combined complexity of mapping inconsistency and redundancy (SI = simple schemas, FD = FD schemas). * The result also holds for arbitrary GLAV mappings.

3 Complexity Results

We summarize below our complexity results. We consider both TBox complexity, i.e., the complexity computed w.r.t. the size of the TBox only, and combined complexity.

For both simple and FD schemas, and for both GAV and GLAV mappings, the TBox complexity of *local mapping inconsistency* turns out to be the same as the TBox complexity of ontology inconsistency. As for the combined complexity, simple and FD schemas behave differently. For simple schemas, it is not necessary to check body-inconsistency (since there are no constraints in $\mathcal{S}$), and thus the combined complexity is the same as for mapping head-inconsistency, which in turn is the same as the combined complexity of ontology inconsistency. For FD schemas we further need to check whether the mapping assertion is body-consistent, which can be done in PTIME. Combining together this result with the above bounds for simple schemas, we obtain the exact bounds for combined complexity shown in Fig. 2.

Global inconsistency can be reduced to checking the consistency of an OBDA specification w.r.t. a (minimal) source database that activates $\mathcal{M}$. In particular we have that for both simple and FD schemas, for both GAV and GLAV mappings, the TBox complexity of global mapping inconsistency is the same as the TBox complexity of ontology inconsistency. As for combined complexity, we devise a non-deterministic algorithm exploiting the above mentioned correspondence of global mapping inconsistency and OBDA inconsistency. This algorithm allows us to prove that for both simple and FD schemas, and for both GAV and GLAV$_{\text{BE}}$ mappings, it holds that: (i) if the ontology language is *DL-Lite*$_R$, RL, or $\mathcal{EL}_\bot$, then the combined complexity of global mapping inconsistency is in NP; (ii) if the ontology language is $\mathcal{SROIQ}$, then it is in N2EXPTIME. We also prove that these bounds are in fact exact.

As for *redundancy*, our investigation shows that both local and global redundancy have the same computational behaviour. The complexity results are obtained with techniques that resemble those used for establishing complexity of global inconsistency. All our complexity results are reported in the tables in Fig. 1 and Fig. 2.

Acknowledgments. This research has been partially supported by the EU under FP7 Large-scale integrating project Optique (grant n. FP7-318338).

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
2. Natalia Antonioli, Francesco Castanò, Spartaco Coletta, Stefano Grossi, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Emanuela Virardi, and Patrizia Castracane. Ontology-based data management for the Italian public debt. In *Proc. of FOIS 2014*, pages 372–385, 2014.
3. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI 2005*, pages 364–369, 2005.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. Diego Calvanese, Martin Giese, Peter Haase, Ian Horrocks, Thomas Hubauer, Yannis E. Ioannidis, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Herald Kllapi, Johan W. Klüwer, Manolis Koubarakis, Steffen Lamparter, Ralf Möller, Christian Neuenstadt, T. Nordtveit, Özgür L. Özçep, Mariano Rodriguez-Muro, Mikhail Roshchin, Domenico Fabio Savo, Michael Schmidt, Ahmet Soylu, Arild Waaler, and Dmitriy Zheleznyakov. Optique: OBDA solution for big data. In *Proc. of ESWC 2013 Satellite Events*, volume 7955 of *LNCS*, pages 293–295. Springer, 2013.
6. Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo. MASTRO STUDIO: Managing ontology-based data access applications. *PVLDB*, 6:1314–1317, 2013.
7. AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
8. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of KR 2006*, pages 57–67, 2006.
9. Evgeny Kharlamov, Martin Giese, Ernesto Jimnez-Ruiz, Martin G. Skjveland, Ahmet Soylu, Dmitriy Zheleznyakov, Timea Bagosi, Marco Console, Peter Haase, Ian Horrocks, Sarunas Marciuska, Christoph Pinkel, Mariano Rodriguez-Muro, Marco Ruzzi, Valerio Santarelli, Domenico Fabio Savo, Kunal Sengupta, Michael Schmidt, Evgenij Thorstensen, Johannes Trame, and Arild Waaler. Optique 1.0: Semantic access to big data: The case of Norwegian petroleum directorate's factpages. In *Proc. of ISWC 2013 Posters & Demos Track*, pages 65–68, 2013.
10. Roman Kontchakov and Michael Zakharyaschev. An introduction to Description Logics and query rewriting. In Manolis Koubarakis, Giorgos B. Stamou, Giorgos Stoilos, Ian Horrocks, Phokion G. Kolaitis, Georg Lausen, and Gerhard Weikum, editors, *RW 2014 Tutorial Lectures*, volume 8714 of *LNCS*, pages 195–244. Springer, 2014.
11. Domenico Lembo, José Mora, Riccardo Rosati, Domenico Fabio Savo, and Evgenij Thorstensen. Towards mapping analysis in ontology-based data access. In *Proc. of RR 2014*, volume 8741 of *LNCS*, pages 108–123. Springer, 2014.
12. Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
13. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
14. Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of ISWC 2013*, volume 8218 of *LNCS*, pages 558–573. Springer, 2013.

The Combined Complexity of Reasoning with Closed Predicates in Description Logics*

Nhung Ngo[1], Magdalena Ortiz[2], and Mantas Šimkus[2]

[1] Free University of Bozen-Bolzano
ngo@inf.unibz.it
[2] Vienna University of Technology
ortiz@kr.tuwien.ac.at|simkus@dbai.tuwien.ac.at

Abstract. We investigate the computational cost of combining the open and closed world assumptions in Description Logics. Unlike previous works, which have considered data complexity and focused mostly on lightweight DLs, we study combined complexity for a wide range of DLs, from lightweight to very expressive ones. From existing results for the standard setting, where all predicates are interpreted under the open world assumption, and the well-known relationship between closed predicates and concept constructors like disjunction and nominals, we infer bounds on the combined complexity of reasoning in the presence of closed predicates. We show that standard reasoning requires exponential time even for weak logics like $\mathcal{EL}$, while answering conjunctive queries becomes hard at least for coNEXPTIME, and in most cases even for 2EXPTIME. An important stepping stone for our results, that is interesting on its own right, is to prove that conjunctive query answering in (plain) $\mathcal{ALCO}$ is hard for coNEXPTIME in combined complexity. This singles out nominals as a previously unidentified source of additional complexity when answering queries over expressive DLs.

1 Introduction

As fragments of classical first-order predicate logic, description logics (DLs) have an *open-world* semantics. That is, knowledge bases (KBs) are interpreted as the set of *all* relational structures that satisfy what is explicitly stated in the KB, and where any statement whose truth is not directly implied by the knowledge base can be interpreted in an arbitrary way. However, this open-world view of DLs is not the most adequate in all cases, and in particular, when DLs are used to describe domain-specific knowledge to be leveraged when querying data sources, but the sources stem from traditional (closed-world) databases that fully describe the instances that are to be included in a relation. For example, when the students enrolled in some course are extracted from a database, this information should be considered complete, and query answering algorithms should exploit this to exclude irrelevant models and infer more query answers.

Combining open and closed world reasoning is not a new topic in DLs [3], but it has received renewed attention in recent years [20,19,7,30]. A prominent proposal for achieving partial closed world reasoning is to use DBoxes instead of ABoxes as the

* This work has been supported by the Austrian Science Fund (FWF) projects T515 and P25207.

assertional component of KBs [30]. Syntactically, a DBox looks just like an ABox, but semantically, it is interpreted like a database: the instances of the concepts and roles in the DBox are given exactly by the assertions it contains, and the *unique name assumption* is made for the *active domain* of the individuals occurring in it. We follow a recent generalization of this setting, where instead of replacing ABoxes by DBoxes, we enrich the terminological component of KBs with a set of concepts and roles that are to be interpreted as *closed predicates* [19]. In this way, some ABox assertions are interpreted under closed semantics, as in DBoxes, while others are considered open, as in ABoxes.

There are not many works studying the complexity of reasoning with closed predicates. For the DL-Lite family and for $\mathcal{EL}$, the *data complexity* of *ontology mediated query answering* has been considered [7,19]. The authors of these works consider a conjunctive query together with a terminological component (a TBox and possibly a set of closed predicates), and study the complexity of answering such a query over an input data instance (an ABox or a DBox). Under the standard open-world semantics for all predicates, this is a central problem that has received great attention in the last decade in the DL community. Most research focuses on the cases where the problem is tractable, or even first-order rewritable. Unfortunately, the tractability of query answering is lost in the presence of DBoxes, even for the core fragments of DL-Lite and $\mathcal{EL}$ [7]. In a nutshell, closed predicates cause the *convexity* property to be lost, allowing a KB to entail a disjunction of facts without entailing any of the disjuncts. An in-depth analysis of this and a careful classification of tractable cases can be found in [19].

In this paper, we take a closer look at the computational complexity of reasoning in the presence of closed predicates. Unlike previous works, we consider the *combined complexity* of reasoning, that is, not only the data is considered as an input, but also the terminological information, and in the case of query answering, the query. Rather than focusing on a few lightweight DLs, we consider a range of logics including very expressive ones, and use existing results in the literature to infer many tight bounds on the computational complexity of query answering. It was shown already in [30] that closed predicates can be simulated, under the standard open world semantics, in any extension of $\mathcal{ALC}$ that supports nominals, and conversely, nominals can be simulated by closed predicates (the latter does not depend on any of the availability of any the constructs of $\mathcal{ALC}$). It is also easy to show that closed predicates suffice to easily express full disjunction and atomic negation in any logic supporting qualified existential restrictions, hence adding closed predicates to plain $\mathcal{EL}$ already results in the full expressiveness of $\mathcal{ALCO}$, and makes standard reasoning require exponential time in the worst case.

For query answering, we build on the reduction from $\mathcal{ALC}$ to $\mathcal{EL}$ to show that the constructors that make query answering 2ExpTime-hard in extensions of $\mathcal{ALC}$ (namely inverses [17], or transitive roles together with role hierarchies [6]), have the same effect in the analogous extensions of $\mathcal{EL}$ in the presence of closed predicates (i.e., $\mathcal{ELI}$ and $\mathcal{EL}\mathcal{H}^{\mathsf{trans}}$). However, since the precise complexity of query answering in $\mathcal{ALCO}$ remains open, we cannot infer tight bounds for the extensions of $\mathcal{EL}$ with closed predicates that do not support these additional constructs. This leads us to the main technical contribution of the paper: we show that conjunctive query answering over $\mathcal{ALCO}$ (with the standard open-world semantics) is coNExpTime-hard. Hence the same holds in the presence of closed predicates for $\mathcal{EL}$ and its extensions. Although we leave a matching upper

bound open for future work, we exhibit nominals (or closed predicates) as a previously unidentified source of increased complexity for query answering in expressive DLs.

2 Preliminaries

We assume the reader is familiar with DLs and, in particular, with $\mathcal{EL}$ and $\mathcal{ALCO}$. We use N_C and N_R for concept names and roles, respectively. The notions of a *TBox* $\mathcal{T}$, an *ABox* $\mathcal{A}$, and an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ are defined in the usual way. The notions of satisfaction $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$ are also as usual. We make the standard name assumption (SNA), i.e., $a^{\mathcal{I}} = a$ for all $\mathcal{I}$ and individuals a. For combining the open- and closed-world semantics, we enrich KBs with a set Σ of *closed predicates*. That is, we consider *knowledge bases (KBs)* $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$, where $\mathcal{T}$ is a TBox, $\Sigma \subseteq N_C \cup N_R$, and $\mathcal{A}$ is an ABox. We call Σ the set of *closed predicates* in $\mathcal{K}$. For such a $\mathcal{K}$ and an interpretation $\mathcal{I}$, we write $\mathcal{I} \models \mathcal{K}$ if
(a) $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$,
(b) for all concept names $A \in \Sigma$, if $e \in A^{\mathcal{I}}$, then $A(e) \in \mathcal{A}$, and
(c) for all roles $r \in \Sigma$, if $(e, e') \in r^{\mathcal{I}}$, then $r(e, e') \in \mathcal{A}$.
In case $\Sigma = \emptyset$, $\mathcal{K}$ is boils down to a usual DL KB and $\mathcal{I} \models \mathcal{K}$ captures the usual notion of satisfaction. In this case, we may simply write $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

Note that in this paper KBs with closed predicates have the semantics as in [19], which relies on the SNA. However, all complexity results of this paper can be recast for the semantics of [7] that employs a weaker form of unique name assumption.

3 Standard Reasoning

Interpreting some predicates as closed allows one to simulate negation, disjunction, and nominals in plain $\mathcal{EL}$, making it as expressive as full $\mathcal{ALCO}$.

Theorem 1. *Assume a consistent $\mathcal{ALCO}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For every nominal $\{a\}$ that appears in $\mathcal{K}$, let D_a be a fresh concept name. Then we can construct in linear time an $\mathcal{EL}$ KB $\mathcal{K}' = (\mathcal{T}', \Sigma, \mathcal{A}')$ with closed predicates such that:*

1. *Every model $\mathcal{I}$ of $\mathcal{K}'$ is a model of $\mathcal{K}$. Moreover, $D_a^{\mathcal{I}} = \{a\}^{\mathcal{I}}$ for every $\{a\}$ in $\mathcal{K}$.*
2. *Every model $\mathcal{I}$ of $\mathcal{K}$ can be transformed into a model of $\mathcal{K}'$ by modifying the interpretation of concept names and roles that do not appear in $\mathcal{K}$.*

Proof. The extension of $\mathcal{EL}$ with atomic negation (i.e., negation is applied to concept names only), denoted $\mathcal{EL}^{\neg}$, is known to be a notational variant of $\mathcal{ALC}$ [1]. Hence we simply show how to construct $\mathcal{K}' = (\mathcal{T}', \Sigma, \mathcal{A}')$ for a given $\mathcal{ELO}^{\neg}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We do this in two steps: first we eliminate nominals and obtain from $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ an $\mathcal{EL}^{\neg}$ KB $\mathcal{K}_1 = (\mathcal{T}_1, \Sigma_1, \mathcal{A}_1)$. Then we transform $\mathcal{K}_1 = (\mathcal{T}_1, \Sigma_1, \mathcal{A}_1)$ into the desired $\mathcal{K}' = (\mathcal{T}', \Sigma, \mathcal{A}')$ in $\mathcal{EL}$. Let

$$\Sigma_1 = \{D_a \mid \{a\} \text{ appears in } \mathcal{K}\}, \qquad \mathcal{A}_1 = \mathcal{A} \cup \{D_a(a) \mid \{a\} \text{ appears in } \mathcal{K}\}.$$

This ensures that $D_a^{\mathcal{I}} = \{a\}^{\mathcal{I}}$ in every model of $\mathcal{A}_1$ where the predicates in Σ_1 are interpreted as closed. Hence we can simply replace each concept $\{a\}$ by D_a in $\mathcal{T}$ to obtain the desired $\mathcal{T}_1$. Next, for eliminating negation from $(\mathcal{T}_1, \Sigma_1, \mathcal{A}_1)$, we let

$\Sigma = \Sigma_1 \cup \{D_\bot, D_1, D_2, D_u\}$ $\quad\quad$ $\mathcal{A}' = \mathcal{A}_1 \cup \{D_u(a_1), D_u(a_2), D_1(a_1), D_2(a_2)\}$

To obtain $\mathcal{T}'$ from $\mathcal{T}_1$, we replace every negated concept name $\neg A$ by a fresh concept name $\bar{A}$, and add the following axioms, where r_A is a fresh role name:

$$A \sqcap \bar{A} \sqsubseteq D_\bot \quad (1) \qquad \exists r_A.D_1 \sqsubseteq A \quad (3)$$
$$\top \sqsubseteq \exists r_A.D_u \quad (2) \qquad \exists r_A.D_2 \sqsubseteq \bar{A} \quad (4)$$

Note that, since there are no assertions of the form $D_\bot(a)$ in $\mathcal{A}'$, we have $D_\bot^\mathcal{I} = \emptyset$ in every model $\mathcal{I}$ of $\mathcal{K}'$, and hence $D_\bot$ behaves as the special concept $\bot$. This and axiom (1) ensure disjointness of A and $\bar{A}$, while axioms (2) – (4) together with the assertions for the closed predicates D_u, D_1 and D_2 ensure that $e \in A^\mathcal{I}$ or $e \in (\bar{A})^\mathcal{I}$ for every $e \in \Delta^\mathcal{I}$. Indeed, let $\mathcal{I}$ be a model of $\mathcal{K}'$ and let $e \in \Delta^\mathcal{I}$ be arbitrary. Then by axiom (2) there exists $e' \in \Delta^\mathcal{I}$ such that $(e, e') \in r^\mathcal{I}$ and $e' \in D_u^\mathcal{I}$. But since D_u is closed, then $e' \in \{a_1, a_2\}$. If $e' = a_1$, then $e' \in D_1^\mathcal{I}$, so $e \in (\exists r_A.D_1)^\mathcal{I}$ and $e \in A^\mathcal{I}$ by axiom (3). Analogously, if $e' = a_2$, we can use axiom (4) to infer that $e \in (\bar{A})^\mathcal{I}$. This ensures that $\bar{A}$ has exactly the same extension as $\neg A$ in every model of $\mathcal{K}$, and the claim follows.

This easy reduction allows us to extend to $\mathcal{EL}$ hardness results known for $\mathcal{ALC}$. We can also infer upper bounds for logics with closed predicates, from known results under the standard open-world semantics, by exploiting the fact that in DLs that contain $\mathcal{ALC}$, closed predicates can be simulated using nominals (see [26] and Prop. 1 in [30]):

Theorem 2. *For every DL KB $(\mathcal{T}, \Sigma, \mathcal{A})$ there exists a logically equivalent KB of the form $(\mathcal{T} \cup \mathcal{T}', \emptyset, \mathcal{A})$, where $\mathcal{T}'$ is a set of $\mathcal{ALCO}$ axioms whose size is polynomially bounded by the size of $\mathcal{A}$.*

This already allows us to obtain an almost complete picture of the landscape for standard reasoning tasks. The following theorem is stated for KB satisfiability, but note that it also applies to other traditional reasoning tasks like subsumption and instance checking since they are mutually reducible to each other in $\mathcal{ALC}$, and hence also in any logic containing $\mathcal{EL}$ with closed predicates.

Corollary 1. *The following bounds for deciding satisfiability of $(\mathcal{T}, \Sigma, \mathcal{A})$ hold:*

1. EXPTIME*-complete if $\mathcal{T}$ and $\mathcal{A}$ are in any DL containing $\mathcal{EL}$ and contained in $\mathcal{SHOQ}$ or $\mathcal{SHOI}$.*
2. NEXPTIME*-complete if $\mathcal{T}$ and $\mathcal{A}$ are in any DL containing $\mathcal{ELIF}$ and contained in $\mathcal{SHOIQ}$.*
3. N2EXPTIME*-complete if $\mathcal{T}$ and $\mathcal{A}$ are in $\mathcal{SRIQ}$ or $\mathcal{SROIQ}$, and in* 2EXPTIME *if $\mathcal{T}$ and $\mathcal{A}$ are in $\mathcal{SROQ}$ or $\mathcal{SROI}$.*

Proof. The lower bound of item (1) follows from Theorem 1 and the well known EXPTIME-hardness of KB satisfiability in $\mathcal{ALC}$ [29]. Similarly, the hardness of items (2) and (3) follows from Theorem 1 together with the NEXPTIME-hardness of $\mathcal{ALCOIF}$ [31], and the N2EXPTIME-hardness of $\mathcal{SROIQ}$ [13]. For the upper bounds, we use Theorem 2 and the fact that KB satisfiability is decidable in the mentioned bounds for the listed logics: $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ in EXPTIME, $\mathcal{SROQ}$ and $\mathcal{SROI}$ in 2EXPTIME, $\mathcal{SHOIQ}$ in NEXPTIME, $\mathcal{SROIQ}$ in N2EXPTIME (see [4,13] and references therein).

4 Query Answering

In this section we consider the query answering problem over DL KBs. We cannot easily transfer complexity upper bounds from known KB satisfiability since, in general, queries are not naturally expressible in the syntax of DLs, and encoding them as part of a KB usually requires exponential space. We focus in *conjunctive queries (CQs)*, whose syntax and semantics are defined in the usual way. In a nutshell, a CQ is a conjunction of atoms of the forms $A(x)$ or $r(x, y)$, for a concept name A or a role name r, and variables x, y. In what follows, all queries are *Boolean queries* with all variables existentially quantified. The decision problem we consider is *query entailment*: deciding whether a given query q is true in all the models of a given KB $(\mathcal{T}, \Sigma, \mathcal{A})$.

It is well known that, under the classical open-world semantics, CQ entailment is hard for 2EXPTIME in most expressive DLs, but the complexity drops to EXPTIME in Horn fragments that disallow disjunction. Unfortunately, since the presence of closed predicates causes disjunction to be expressible, 2EXPTIME-hardness extends to many extensions of $\mathcal{EL}$. For CQs, 2EXPTIME-hardness can be shown whenever the DL supports inverse roles [17], or a single left-identity axiom $r \circ t \sqsubseteq t$, or a transitive super role of some role [6]. If we consider query languages that extend CQs, like positive queries or (fragments of) conjunctive (2-way) regular path queries (C(2)RPQs), the same hardness holds already for plain $\mathcal{ALC}$ [25], and hence for $\mathcal{EL}$ with closed predicates.

Below we denote by $\mathcal{EL}^{\mathsf{LI}}$ the extension of $\mathcal{EL}$ with a single left-identity axiom $r \circ t \sqsubseteq t$, and by $\mathcal{ELH}^{\mathsf{trans}}$ the extension of $\mathcal{EL}$ with role inclusions and a transitive role. Note that both logics are sublogics of $\mathcal{EL}^{++}$.

Theorem 3. *Deciding $(\mathcal{T}, \Sigma, \mathcal{A}) \models q$ is hard for 2EXPTIME in all the following cases:*

1. *$\mathcal{T}$ and $\mathcal{A}$ are in $\mathcal{ELI}$ and q is a CQ.*
2. *$\mathcal{T}$ and $\mathcal{A}$ are in $\mathcal{EL}^{\mathsf{LI}}$ or in $\mathcal{ELH}^{\mathsf{trans}}$, and q is a CQ.*
3. *$\mathcal{T}$ and $\mathcal{A}$ are in $\mathcal{EL}$ and q is either a positive query, a $*$-free CRPQ, or a $*$-free C2RPQ with only two variables.*

Proof. We have shown in Theorem 1 that for every $\mathcal{ALC}$ KB $\mathcal{K}$ there is an $\mathcal{EL}$ KB $\mathcal{K}'$ that has essentially the same models, and may only differ in the interpretation of symbols not occurring in $\mathcal{K}$. Hence, for every query q, we have $\mathcal{K} \models q$ iff $\mathcal{K}' \models q$. This translation can be applied to extensions of $\mathcal{ALC}$, and results in a KB with the same properties in the analogous extension of $\mathcal{EL}$. In particular, an $\mathcal{ALCI}$ KB is rewritten into a $\mathcal{ELI}$ one, and an $\mathcal{SH}$ KB into an $\mathcal{ELH}^{\mathsf{trans}}$ one. From this an existing results for $\mathcal{ALC}$ and its extensions, we obtain the desired lower bounds: item 1 follows from [18], item 2 follows from [6], and item 3 follows from [25].

Matching upper bounds are known, even for significantly more expressive queries and logics: in the standard setting, with no closed predicates, entailment of *positive two-way regular path queries (P2RPQs)* is in 2EXPTIME for any DL contained in $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, $\mathcal{ZOI}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$ [4]. From this and Theorem 2, we get the same upper bound for $\mathcal{ZOQ}$, $\mathcal{ZOI}$, $\mathcal{SHOQ}$, $\mathcal{SHOI}$ and their sublogics.

Corollary 2. *Let q be a P2RPQ. Then deciding $(\mathcal{T}, \Sigma, \mathcal{A}) \models q$ is* 2EXPTIME *complete for $\mathcal{T}$ and $\mathcal{A}$ in any DL containing $\mathcal{EL}$ and contained in $\mathcal{ZOQ}$, $\mathcal{ZOI}$, $\mathcal{SHOQ}$, $\mathcal{SHOI}$. The same holds for q a CQ if $\mathcal{T}$ and $\mathcal{A}$ are in a DL containing $\mathcal{ELI}$ or $\mathcal{EL}^{\mathsf{LI}}$.*

Corollary 2 implies that query entailment in the presence of closed predicates is almost always 2EXPTIME-complete in combined complexity. But there are some exceptions. On the one hand, the interaction of nominals, inverses, and counting makes query entailment very challenging. In the plain open-world setting, entailment of conjunctive queries is coN2EXPTIME-hard for $\mathcal{ALCOIF}$ [10], and it has been shown to be decidable [28], but no elementary upper bounds on its complexity are known. For its extension with transitive roles, and for the well known $\mathcal{SHOIQ}$, decidability remains open. Hence, in the presence of closed predicates, we do not get any interesting upper bounds for DLs that simultaneously support inverses and counting, like $\mathcal{ALCIF}$ and $\mathcal{SHIQ}$. Moreover, obtaining such bounds seems very hard. We remark that the authors of [7] proved that query entailment in $\mathcal{ALCIF}$ with closed predicates and query entailment under the standard open-world semantics in $\mathcal{ALCOIF}$ are mutually reducible.

On the other hand, the mentioned 2EXPTIME lower bounds for CQs require the presence of either inverse roles, left identity axioms, or transitivity and role hierarchies. For $\mathcal{EL}$ (with closed predicates), $\mathcal{ALC}$, and their extensions that have neither inverses nor left identities, we only have the EXPTIME lower bound from KB satisfiability, and the 2EXPTIME upper bound of Corollary 2. Without closed predicates, CQ entailment for plain $\mathcal{ALC}$, and even for $\mathcal{ALCHQ}$, is feasible in single exponential time [18,24]. A natural question is whether nominals, or equivalently, closed predicates, can be added to $\mathcal{ALCHQ}$ without increasing the worst-case complexity of CQ entailment. Unfortunately, the answer is negative (unless coNEXPTIME = EXPTIME), as we show next.

A coNEXPTIME lower bound for CQ entailment in $\mathcal{ALCO}$

In this section, we show that deciding whether $(\mathcal{T}, \mathcal{A}) \not\models q$ for a given CQ q and a given $\mathcal{ALCO}$ KB $(\mathcal{T}, \mathcal{A})$ (with the standard open-world semantics), is hard for non-deterministic single exponential time. By Theorem 1, the same applies to $\mathcal{EL}$ in the presence of closed predicates.

Before we start with the proof, we recall a useful property of $\mathcal{ALCO}$: for query answering, it is enough to focus on *forest-shaped models*. A *forest* is a set F of non-empty words such that $w \cdot c \in F$ with w non-empty implies $w \in F$. An interpretation $\mathcal{I}$ is *forest-shaped* if there is a bijection f from its domain to a forest, such that

- $f(a^{\mathcal{I}})$ has length one for every individual a, and
- $(e, e') \in r^{\mathcal{I}}$ implies that either $e' = a$ for some individual a, or $f(e')$ is of the form $f(e) \cdot c$ for some symbol c.

For many DLs and query languages, it has been shown that query entailment can be decided over forest shaped interpretations. This applies also to CQs over $\mathcal{ALCO}$ KBs:

Lemma 1 ([9,4]). *Let $\mathcal{K}$ be a given $\mathcal{ALCO}$ KB and let q be a CQ. Then $\mathcal{K} \not\models q$ iff there is a forest shaped interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \not\models q$.*

Now we show our lower bound. The proof is by reduction from the following coNEXPTIME-complete variant of the tiling problem:

Definition 1 (Domino System). *A* domino system $\mathfrak{D}$ *is a triple* (T, H, V)*, where* $T = \{0, \dots, k-1\}$*,* $k \geq 0$*, is a finite set of* tile types *and* $H, V \subseteq T \times T$ *represent the* horizontal and vertical matching conditions*. Let* $\mathfrak{D}$ *be a domino system and* $c = c_0, \dots, c_{n-1}$ *an* initial condition*, i.e. an* n*-tuple of tile types. A mapping* $\tau : \{0, \dots, 2^{n+1}-1\} \times \{0, \dots, 2^{n+1}-1\} \to T$ *is a* solution *for* $\mathfrak{D}$ *and* c *iff for all* $x, y < 2^{n+1}$*, the following holds (where* $\oplus_i$ *denotes addition modulo* i*):*

- *if* $\tau(x, y) = t$ *and* $\tau(x \oplus_{2^{n+1}} 1, y) = t'$*, then* $(t, t') \in H$
- *if* $\tau(x, y) = t$ *and* $\tau(x, y \oplus_{2^{n+1}} 1) = t'$*, then* $(t, t') \in V$
- $\tau(i, 0) = c_i$ *for* $i < n$*.*

For the reduction, we do not need a full $\mathcal{ALCO}$ knowledge base, but a simple ABox with single concept assertion $C_{\mathfrak{D},c}(a)$ for a complex $\mathcal{ALCO}$ concept $C_{\mathfrak{D},c}$. We show how to translate a given domino system $\mathfrak{D}$ and initial condition $c = c_0 \cdots c_{n-1}$ into an assertion $C_{\mathfrak{D},c}(a)$ and query $q_{\mathfrak{D},c}$ such that each forest-shaped model $\mathcal{I}$ of $C_{\mathfrak{D},c}(a)$ that satisfies $\mathcal{I} \not\models q_{\mathfrak{D},c}$ encodes a solution to $\mathfrak{D}$ and c, and conversely each solution to $\mathfrak{D}$ and c gives rise to a model of $C_{\mathfrak{D},c}(a)$ with $\mathcal{I} \not\models q_{\mathfrak{D},c}$.

Our reduction is based on the proof of coNEXPTIME-hardness of *rooted* query entailment in $\mathcal{ALCI}$ [17], and also resembles the similar proof for $\mathcal{S}$ [6]. In fact, the first part of the concept $C_{\mathfrak{D},c}$, which generates forest models that encode a potential solutions, is essentially as in [17]. The second part and the query are quite different, since they exploit nominals to detect errors in potential solutions.

Constructing the ABox. We now define the complex concept $C_{\mathfrak{D},c}$, and the desired ABox is $\{C_{\mathfrak{D},c}(a)\}$. We assume that $C_{\mathfrak{D},c}$ is a conjunction of the form $C^1_{\mathfrak{D},c} \sqcap C^2_{\mathfrak{D},c}$.

For convenience, let $m = 2n+2$. The purpose of the first conjunct $C^1_{\mathfrak{D},c}$ is to enforce a binary tree of depth m, whose edges are labeled with a single role r, and whose leaves are labeled with the numbers $0, \dots, 2^m - 1$ of a binary counter C, implemented using concept names $B_0, \dots, B_m$. Intuitively, each of these leaves ℓ stores a position in the $2^{n+1} \times 2^{n+1}$-grid to be tiled: the bits $B_0, \dots B_n$ encode the horizontal position x, and the bits $B_{n+1}, \dots B_m$ encode the vertical position y. We also use a concept name D_i for each tile type $i \in T$. Each leaf g storing a position (x, y) has as r-children three 'grid nodes' g_h, g_{right}, and g_{up} labeled G, which satisfy all the following conditions:

1. g_h represents the grid node with position (x, y), and stores the same bit address as g (that is, g_h and g coincide on the interpretation of all B_i).
2. g_{right} and g_{up} represent the right- and up-neighbor of g, and respectively store the addresses $(x \oplus_{2^{n+1}} 1, y)$ and $(x, y \oplus_{2^{n+1}} 1)$.
3. g_h is labeled G_h, while g_{right} and g_{up} are labeled G_s.
4. g_h (resp., g_{right}, g_{up}) satisfies exactly one concept D_i, representing the assigned tile type $\tau(x, y)$ (resp., $\tau(x \oplus_{2^{n+1}} 1, y)$, $\tau(x, y \oplus_{2^{n+1}} 1)$).
5. The tiling of the g_h nodes respects the initial condition, that is, if ℓ_h stores the position $(i, 0)$, then it satisfies D_{c_i}.
6. The tiling of g_{right} and g_{up} respect the matching conditions, that is, if g_h satisfies D_i, g_{right} satisfies D_j, and g_{up} satisfies $D_{j'}$, then $(D_i, D_j) \in H$ and $(D_i, D_{j'}) \in V$.

The tree we have described almost describes a solution for $\mathfrak{D}$, except for the crucial fact that different copies of the same node in the grid may have different types assigned.

That is, for an address (x, y), the g_{right} and g_{up} nodes with address (x, y) need not satisfy the same D_i as the g_h with address (x, y). We call a model $\mathcal{I}$ of $C^1_{\mathfrak{D},c}(a)$ *proper* if it satisfies the following condition:

($\star$) For every pair $g \in G^{\mathcal{I}}_h$, $g' \in G^{\mathcal{I}}_s$ such that $g \in B_i$ iff $g' \in B_i$ for all $0 \leq i \leq m$, there exists some $i < k$ such that $\{g, g'\} \subseteq D^{\mathcal{I}}_i$.

We can use an $\mathcal{ALC}$ concept $C^1_{\mathfrak{D},c}$ to enforce a tree as above, such that deciding the existence of a solution for $\mathfrak{D}$ and c reduces to finding a proper model of $C^1_{\mathfrak{D},c}$. Such constructions exist in the literature, and in fact, the concept we described is just a minor modification of the conjunction $C^1_{\mathfrak{D},c} \sqcap \cdots \sqcap C^6_{\mathfrak{D},c}$ given in [17], hence we omit its rather technical definition. Instead, we rely on the following claim:

Lemma 2 (implicit in [17]). *Let $\mathfrak{D}$ be a domino system and c an initial condition. Then we can build an $\mathcal{ALC}$ concept $C^1_{\mathfrak{D},c}$ such that there exists a solution for $\mathfrak{D}$ and c iff there exists a proper model of $C^1_{\mathfrak{D},c}(a)$. Moreover, the size of $C^1_{\mathfrak{D},c}$ and the time needed to construct it are polynomially bounded by the size of $\mathfrak{D}$.*

We construct below a query $q_{\mathfrak{D},c}$ that does *not* match a forest model of $C^1_{\mathfrak{D},c}(a)$ iff ($\star$) is satisfied. By Lemmas 2 and 1, this suffices to decide whether there exists a solution for $\mathfrak{D}$ and c. But before defining $q_{\mathfrak{D},c}$, we define the second conjunct $C^2_{\mathfrak{D},c}$ of $C_{\mathfrak{D},c}$. Its purpose is to add nodes and labels to the forest models of $C^1_{\mathfrak{D},c}(a)$ that allow us to test for ($\star$) using a CQ.

For defining $C^2_{\mathfrak{D},c}$, we use the following additional alphabet symbols:

- two individual names a_i and $\bar{a}_i$ and one concept name A_i for each bit B_i, $0 \leq i \leq m$,
- an individual name t_j for each tile type $j < k$,
- a concept T stating that some individual stands for a tile type.

Each G node g is linked via r-arcs to the individuals $a_i, \bar{a}_i$ that encode its bit address. We also link g nodes to the individuals that stand for the tile types, but we do it differently for the G_h nodes and the G_s nodes, as follows:

- If g is a G_h-node with tile type D_i, then g has an r-arc to t_i.
- If g is a G_s-node with tile type D_i, then g has an r-arc to each t_j with $j \neq i$.

Finally, we make both a_i and $\bar{a}_i$ instances of A_i, for each bit i, and we make all tile types t_j instances of T. Formally, this is all ensured using the conjunction $C^2_1 \sqcap C^2_2$ of the following two concepts:

$$C^2_1 = \forall r^{m+1}.\Big(\bigsqcap_{0 \leq i \leq m} \big(B_i \to \exists r.\{a_i\} \sqcap \neg B_i \to \exists r.\{\bar{a}_i\} \sqcap \bigsqcap_{0 \leq i < k} D_i \to \big((G_h \to \exists r.\{t_i\}) \sqcap (G_s \to (\bigsqcap_{0 \leq j < k, j \neq i} \exists r.t_j)) \big)$$

$$C^2_2 = \forall r^{m+2}.\big((\bigsqcap_{0 \leq i \leq m} \{a_i, \bar{a}_i\} \to A_i) \sqcap \big(\{t_0, \ldots t_{k-1}\} \to T \big) \big)$$

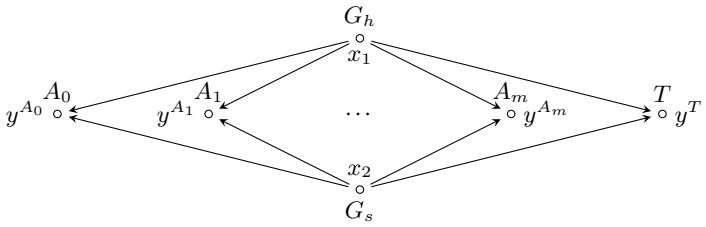


Fig. 1: The query $q_{\mathfrak{D},c}$

Now we are ready to define our ABox $\{C_{\mathfrak{D},c}(a)\}$, by taking $C^2_{\mathfrak{D},c} = C^2_1 \sqcap C^2_2$ as defined above, $C^1_{\mathfrak{D},c}$ as in Lemma 2, and $C_{\mathfrak{D},c} = C^1_{\mathfrak{D},c} \sqcap C^2_{\mathfrak{D},c}$. Every model of $C_{\mathfrak{D},c}(a)$ is a model of $C^1_{\mathfrak{D},c}(a)$, and every forest model of $C^1_{\mathfrak{D},c}(a)$ can be extended to a model of $C_{\mathfrak{D},c}(a)$ while preserving properness, hence from Lemma 2 we get:

Lemma 3. *$\mathfrak{D}$ and c have a solution iff there exists a proper forest model of $C_{\mathfrak{D},c}(a)$.*

It is only left to define a query $q_{\mathfrak{D},c}$ that matches a forest model of $C_{\mathfrak{D},c}(a)$ if and only if it is not proper. We will rely on the following properties ensured by $C^1_{\mathfrak{D},c}$. First, the connections to the individuals representing the bit address ensure the following:

(†) Let g, g' be two G-nodes. Then g and g' share an r-arc to a common individual from $a_i, \bar{a}_i$ for each $0 \leq i \leq m$ iff they have the same bit address.

Since the links to the tile types for the G_h-nodes and for the G_s-nodes are inverted, we also have:

(‡) Let g_h be a G_h-node and g_s be a G_s-node. Then there exists some t_j such that both g_h and g_s have an r-arc to t_j iff g_h and g_s have different tile types.

Hence, to establish non-properness it suffices to find a G_h-node and a G_s-node that share an r-arc to a common individual from $a_i, \bar{a}_i$ for each $0 \leq i \leq m$ (and hence share the same address), but also share a link to a t_j node (and hence have different tile type). This is done with the following query:

$$\begin{aligned} q_{\mathfrak{D},c} = &\exists x_1, x_2, y^{A_0}, , \dots, y^{A_m}, y^T . G_h(x_1), G_s(x_2), \\ &r(x_1, y^{A_0}), A_0(y^{A_0}), \dots, r(x_1, y^{A_m}), A_m(y^{A_m}), r(x_1, y^T), T(y^T), \\ &r(x_2, y^{A_0}), A_0(y^{A_0}), \dots, r(x_2, y^{A_m}), A_m(y^{A_m}), r(x_2, y^T), T(y^T). \end{aligned}$$

The query $q_{\mathfrak{D},c}$ is illustrated in Figure 1. To see that $q_{\mathfrak{D},c}$ has a match iff ($\star$) fails, we first note that x_1 can only be matched to a G_h node and x_2 to a G_s node. Each y^{A_i} must be matched to an instance of A_i, which is one of a_i and $\bar{a}_i$. Since x_1 and x_2 are connected to all the y^{A_i}, then they need to have either a_i or $\bar{a}_i$ as common successor, for each i, in every model where the query has a match. By (†), this is the case exactly when they have the same bit address. The variable y^T can match instances of T, which are only the tile type individuals t_j, and since x_1 and x_2 are both connected to y^T, we have

that x_1 and x_2 can only be matched to nodes sharing a link to a common t_j. By ($\ddagger$), and since the matches of x_1 and x_2 are a G_h node and a G_s node, they must have a different tile type. Hence the query has a match iff there are a G_h node g_h and a G_s node g_s that have the same bit address and different tile types, that is, there is a pair violating the condition of ($\star$) and the model is not proper. Hence we get that there is a model $\mathcal{I}$ of $C_{\mathfrak{D},c}(a)$ where there is no match for $q_{\mathfrak{D},c}$, and iff there exist a proper model of $C_{\mathfrak{D},c}(a)$ iff there is a solution for $\mathfrak{D}$ and c.

Lemma 4. *$C_{\mathfrak{D},c}(a) \not\models q_{\mathfrak{D},c}$ iff there is a solution for $\mathfrak{D}$ and c.*

From this, the hardness of the given tiling problem, and Theorem 1, we get:

Theorem 4. *The following problems are hard for* coNEXPTIME*:*

- *deciding $(\emptyset, \{C(a)\}) \models q$ for q a CQ and C an $\mathcal{ALCO}$ concept, and*
- *deciding $(\mathcal{T}, \Sigma, \mathcal{A}) \models q$ for q a CQ and $\mathcal{T}, \mathcal{A}$ in $\mathcal{EL}$.*

Unfortunately, we do not have matching upper bounds. We believe that both problems are likely to be solvable in coNEXPTIME, but we are still working on a suitable algorithm.

5 Discussion and Outlook

In this paper we have given several bounds on the combined complexity of reasoning in various DLs in the presence of closed predicates, for standard reasoning problems like KB satisfiability, as well as for answering queries ranging from CQs to P2RPQs. Unlike the data-complexity, that is coNP-complete in practically all cases (from DL-Lite$_{core}$[7,19] to expressive DLs like $\mathcal{ALCHOQ}$ and $\mathcal{ALCHOI}$ [21]), the combined complexity offers a complex landscape. We summarize some results in Table 1, emphasizing the cases in which closed predicates have an interesting effect on the complexity.

Apart from establishing the precise complexity of query answering in DLs between $\mathcal{ALCO}$ and $\mathcal{ALCHOQ}$ (without closed predicates), and in all DLs between $\mathcal{EL}$ and $\mathcal{ALCHOQ}$ with closed predicates, other problems remain open. Notably, in this paper we have not considered the DL-Lite family. An algorithm for CQ entailment in DL-Lite$_{\mathcal{F}}$ was developed in [7] to obtain a coNP upper bound in data complexity, but it only yields very high bounds on the combined complexity that are likely not to be optimal. That algorithm deals with the intricate interactions of inverse roles, functionality, and closed predicates, that behave as nominals, and it may be possible to use the characterization of countermodels given there as a starting point for a better combined complexity upper bound. In the DL-Lite variants that do not support functionality, countermodels are likely to have a simpler structure. However, even in these simpler languages, it is not apparent whether interesting upper bounds can be obtained by simple adaptations of existing techniques. In particular, it has been shown that singleton nominals do not increase neither the data nor the combined complexity of DL-Lite [11], but the effect of the combination of nominals and (restricted) disjunction that results from closed predicates remains to be studied for the DL-Lite family.

Finally, we have seen that in general, the disjunctive information encoded by closed predicates has a negative effect on the complexity of reasoning. In the light of this, it

	Without closed predicates		With closed predicates	
	KB consistency	CQ entailment	KB consistency	CQ entailment
$\mathcal{EL}$	P [1]	NP [14,27]	ExpTime (Cor. 1.1)	$\geq$ coNExpTime (Th. 4) $\leq$ 2ExpTime (Cor. 2)
$\mathcal{ELH}^{\text{trans}}$	P [1]	$\geq$ NP,$\leq$PSpace [15]	ExpTime (Cor. 1.1)	2ExpTime (Cor. 2)
$\mathcal{ELI}$	ExpTime [2]	ExpTime [5]	ExpTime (Cor. 1.1)	2ExpTime (Cor. 2)
Horn-$\mathcal{SHOI}$ Horn-$\mathcal{SHOQ}$	ExpTime [16,22]	ExpTime [23]	ExpTime (Cor. 1.1)	2ExpTime (Cor. 2)
$\mathcal{ELIF}$ Horn-$\mathcal{SHIQ}$	ExpTime [16]	ExpTime [5]	NExpTime (Cor. 1.2)	$\geq$ N2ExpTime [10] decidable [28] / $\leq$ open*
$\mathcal{ELOIF}$, Horn-$\mathcal{SHOIQ}$	ExpTime [22]	ExpTime [23]	NExpTime (Cor. 1.2)	$\geq$ N2ExpTime [10] decidable [28] / $\leq$ open*
$\mathcal{ALCO}$	ExpTime [32,8]	$\geq$ coNExpTime (Th. 4)	ExpTime (Cor. 1.1)	$\geq$ coNExpTime (Th. 4) $\leq$ 2ExpTime (Cor. 2)
$\mathcal{SHOQ}, \mathcal{SHOI}$	ExpTime [32,12,8]	2ExpTime [4,9]	ExpTime (Cor. 1.1)	2ExpTime (Cor. 2)
$\mathcal{SHOIQ}$	NExpTime [31]	$\geq$ N2ExpTime [10] $\leq$ open*	NExpTime (Cor. 1.2)	$\geq$ N2ExpTime [10] $\leq$ open*

Table 1: Combined complexity of reasoning in description logics with/without closed predicates. By $\geq$ we indicate lower bounds, by $\leq$ upper bounds, and the rest are all completeness results. For the cells marked with *, decidability if only simple roles occur in the query follows from [28], but no complexity upper bounds are known.

seems particularly interesting to study criteria that allow to identify instances of TBoxes and queries for which the complexity does not increase. Major contributions in this direction can be found in [19]. In particular, the authors propose *safety* criteria that ensure specific TBoxes have the convexity property, guaranteeing data tractability of queries. It seems that this criteria may also be useful for establishing the existence of a *universal model* for query answering, and when a suitable representation of such a model can be built in single exponential time, query answering is likely to be feasible in single exponential time. This seems a promising direction for further investigation.

References

1. F. Baader, S. Brand, and C. Lutz. Pushing the el envelope. In *Proc. of IJCAI 2005*, pages 364–369. Morgan-Kaufmann Publishers, 2005.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope further. In *Proc. of OWLED 2008*, 2008.
3. M. Cadoli, F. M. Donini, and M. Schaerf. Closed world reasoning in hybrid systems. In *Proc. of ISMIS 1990*, pages 474–481, 1990.
4. D. Calvanese, T. Eiter, and M. Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. of IJCAI 2009*, pages 714–720, 2009.
5. T. Eiter, G. Gottlob, M. Ortiz, and M. Šimkus. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proc. of JELIA 2008*, pages 166–179, Berlin, Heidelberg, 2008. Springer-Verlag.
6. T. Eiter, C. Lutz, M. Ortiz, and M. Šimkus. Query answering in description logics with transitive roles. In *Proc. of IJCAI 2009*, volume 9, pages 759–764, 2009.
7. E. Franconi, Y. A. Ibáñez-García, and I. Seylan. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.*, 278:71–84, 2011.
8. B. Glimm, I. Horrocks, and U. Sattler. Deciding $\mathcal{SHOQ}^{\cap}$ knowledge base consistency using alternating automata. In *Proc. of DL 2008*, volume 353 of *CEUR Workshop Proceedings*, 2008.
9. B. Glimm, I. Horrocks, and U. Sattler. Unions of conjunctive queries in SHOQ. In *Proc. of KR 2008*, pages 252–262. AAAI Press/The MIT Press, 2008.
10. B. Glimm, Y. Kazakov, and C. Lutz. Status QIO: An update. In *Proc. of DL 2009*, volume 745 of *CEUR Workshop Proceedings*, 2011.
11. M. Haddad and D. Calvanese. Extending DL-Lite_A with (singleton) nominals. In *Proc. DL 2013*, volume 1014 of *CEUR Workshop Proceedings*, 2013.
12. J. Hladik. A tableau system for the description logic SHIO. In U. Sattler, editor, *IJCAR Doctoral Programme*, volume 106 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
13. Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *Proc. of KR 2008*, pages 274–284. AAAI Press, 2008.
14. M. Krötzsch and S. Rudolph. Conjunctive queries for $\mathcal{EL}$ with composition of roles. In *Proc. of DL 2007*, volume 250 of *CEUR Electronic Workshop Proceedings*, 2007.
15. M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proc. of ISWC 2007 + ASWC 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 310–323. Springer, 2007.
16. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexities of Horn description logics. *ACM Trans. Comp. Log.*, 14(1):2:1–2:36, 2013.
17. C. Lutz. Inverse roles make conjunctive queries hard. In *Proc. of DL 2007*, pages 100–111, 2007.
18. C. Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. of IJCAR 2008*, pages 179–193. Springer, 2008.
19. C. Lutz, I. Seylan, and F. Wolter. Ontology-based data access with closed predicates is inherently intractable(sometimes). In *Proc. of IJCAI 2013*. IJCAI/AAAI, 2013.
20. C. Lutz, I. Seylan, and F. Wolter. Ontology-mediated queries with closed predicates. In *Proc. of IJCAI 2015*. IJCAI/AAAI,, 2015.
21. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. 41(1):61–98, 2008.
22. M. Ortiz, S. Rudolph, and M. Šimkus. Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In *Proc. of KR 2010*. AAAI Press, 2010.

23. M. Ortiz, S. Rudolph, and M. Šimkus. Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In *Proc. of IJCAI 2011*, pages 1039–1044. IJCAI/AAAI, 2011.
24. M. Ortiz, M. Šimkus, and T. Eiter. Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In *Proc. of AAAI 2008*, pages 504–510. AAAI Press, 2008.
25. M. Ortiz and M. Šimkus. Revisiting the hardness of query answering in expressive description logics. In *Proc. of RR 2014*, pages 216–223. Springer International Publishing, 2014.
26. R. Reiter. Equality and domain closure in first-order databases. *J. ACM*, 27(2):235–249, Apr. 1980.
27. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of DL 2007*, volume 250 of *CEUR Electronic Workshop Proceedings,* 2007.
28. S. Rudolph and B. Glimm. Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! *J. of Artificial Intelligence Research*, 39:429–481, 2010.
29. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI 1991*, pages 466–471. Morgan Kaufmann, 1991.
30. I. Seylan, E. Franconi, and J. de Bruijn. Effective query rewriting with ontologies over DBoxes. In *Proc. of IJCAI 2009*, pages 923–925, 2009.
31. S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.
32. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

Completion Graph Caching for Expressive Description Logics

Andreas Steigmiller*[1], Birte Glimm[1], and Thorsten Liebig[2]

[1] University of Ulm, Ulm, Germany, <first name>.<last name>@uni-ulm.de
[2] derivo GmbH, Ulm, Germany, liebig@derivo.de

1 Introduction

Reasoning in very expressive Description Logics (DLs) such as $\mathcal{SROIQ}$ is often hard since non-determinism, e.g., from disjunctions or cardinality restrictions, requires a case-by-case analysis and since a strict separation between intensional (TBox) and extensional (ABox) knowledge is not possible due to nominals. Current state-of-the-art reasoners for $\mathcal{SROIQ}$ are typically based on (hyper-)tableau algorithms, where higher level reasoning tasks such as classification are reduced to possibly many consistency tests. For large ABoxes and more expressive DLs, reasoning then easily becomes infeasible since the ABox has to be considered in each test.

For less expressive DLs, optimisations have been proposed that allow for considering only parts of the ABox for checking whether an individual is an instance of a specific concept [22,23]. It is, however, not clear how these optimisations can be extended to $\mathcal{SROIQ}$. Furthermore, approaches based on partitioning and modularisation require a syntactic pre-analysis of the concepts, roles, and individuals in a KB, which can be quite costly for more expressive DLs and, due to the static analysis, only queries with specific concepts are supported. Rewriting axioms such that ABox reasoning is improved also carries the risk that this negatively influences other reasoning tasks for which ABox reasoning is potentially not or less relevant.

Another possibility is the caching of the completion graph that is built by the tableau algorithm during the initial consistency check. The re-use of (parts of) the cached completion graph in subsequent tests reduces the number of consequences that have to be re-derived for the individuals of the ABox. Existing approaches based on this idea (e.g., [17]) are, however, not very suitable for handling non-determinism. Since caching and re-using non-deterministically derived facts can easily cause unsound consequences, the non-deterministically derived facts are usually simply discarded and re-derived if necessary. We address this and present a technique that allows for identifying non-deterministic facts that can safely be re-used. The approach is based on a set of conditions that can be checked locally and, thus, allows for efficiently identifying individuals step-by-step for which non-deterministic consequences have to be re-considered in subsequent tests. The presented technique can directly be integrated into existing tableau-based reasoning systems without significant adaptations and reduces reasoning effort for all tasks for which consequences from the ABox are potentially relevant. Moreover,

* The author acknowledges the support of the doctoral scholarship under the Postgraduate Scholarships Act of the Land of Baden-Wuerttemberg (LGFG).

it can directly be used for the DL $\mathcal{SROIQ}$, does not produce a significant overhead, and can easily be extended, e.g., to improve incremental ABox reasoning. Note that completion graph caching is complementary to many other optimisations that try to reduce the number of subsequent tests for higher level reasoning tasks, e.g., summarisation [2], abstraction and refinement [5], bulk processing and binary retrieval [6], (pseudo) model merging [7], extraction of known/possible instances from model abstractions [15].

The paper is organised as follows: We next introduce some preliminaries and then present the basic completion graph caching technique in Section 3. In Section 4, we present several extensions and applications to support nominals in ordinary satisfiability caching, to perform incremental reasoning for changing ABoxes, and to handle very large ABoxes by storing data in a representative way. Finally, we present an evaluation of the presented techniques in Section 5 and conclude in Section 6. Further details and an extended evaluation are available in a technical report [19].

2 Preliminaries

For brevity, we do not introduce DLs (see, e.g., [1,10]). We use (possibly with subscripts) C, D for (possibly complex) concepts, A, B for atomic concepts, and r, s for roles. We assume that a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ is expressed as the union of a TBox $\mathcal{T}$ consisting of GCIs of the form $C \sqsubseteq D$, a role hierarchy $\mathcal{R}$, and an ABox $\mathcal{A}$, such that the effects of complex roles are implicitly encoded (e.g., based on automata [12] or regular expressions [16]). We write $\mathsf{nnf}(C)$ to denote the negation normal form of C. For r a role, we set $\mathsf{inv}(r) := r^-$ and $\mathsf{inv}(r^-) := r$.

Model construction calculi such as tableau [10,13] decide the consistency of a KB $\mathcal{K}$ by trying to construct an abstraction of a model for $\mathcal{K}$, a so-called *completion graph*. For the purpose of this paper, we use a tuple of the form $(V, E, \mathcal{L}, \dot{\neq}, \mathcal{M})$ for a completion graph G, where each node $x \in V$ (edge $\langle x, y\rangle \in E$) represents one or more (pairs of) individuals. Each node x (edge $\langle x, y\rangle$) is labelled with a set of concepts (roles), $\mathcal{L}(x)$ ($\mathcal{L}(\langle x, y\rangle)$), which the (pairs of) individuals represented by x ($\langle x, y\rangle$) are instances of. The relation $\dot{\neq}$ records inequalities, which must hold between nodes, e.g., due to at-least cardinality restrictions, and the mapping $\mathcal{M}$ tracks merging activities, e.g., due to at-most cardinality restrictions or nominals. The algorithm works by decomposing concepts in the completion graph with a set of expansion rules. For example, if $C_1 \sqcup C_2 \in \mathcal{L}(v)$ for some node v, but neither $C_1 \in \mathcal{L}(v)$ nor $C_2 \in \mathcal{L}(v)$, then the rule for handling disjunctions non-deterministically adds one of the disjuncts to $\mathcal{L}(v)$. Similarly, if $\exists r.C \in \mathcal{L}(v)$, but v does not have an r-successor with C in its label, then the algorithm expands the completion graph by adding the required successor node. If a node v is merged into a node v', $\mathcal{M}$ is extended by $v \mapsto v'$. We use $\mathsf{mergedTo}^{\mathcal{M}}(v)$ to return the node into which a node v has been merged, i.e., $\mathsf{mergedTo}^{\mathcal{M}}(v) = \mathsf{mergedTo}^{\mathcal{M}}(w)$ for $v \mapsto w \in \mathcal{M}$ and $\mathsf{mergedTo}^{\mathcal{M}}(v) = v$ otherwise.

Unrestricted application of rules for existential or at-least cardinality restrictions can lead to the introduction of infinitely many new nodes. To guarantee termination, a cycle detection technique called *(pairwise) blocking* [11] restricts the application of these rules. The rules are repeatedly applied until either the completion graph is fully expanded (no more rules are applicable), in which case the graph can be used to con-

struct a model that *witnesses* the consistency of $\mathcal{K}$, or an obvious contradiction (called a *clash*) is discovered (e.g., both C and $\neg C$ in a node label), proving that the completion graph does not correspond to a model. A knowledge base $\mathcal{K}$ is *consistent* if the rules can be applied such that they build a fully expanded and clash-free completion graph.

3 Completion Graph Caching and Reusing

In the following, we use superscripts to distinguish different versions of completion graphs. We denote with $G^d = (V^d, E^d, \mathcal{L}^d, \doteq^d, \mathcal{M}^d)$ the last completion graph of the initial consistency test that is obtained with only deterministic rule applications and with $G^n = (V^n, E^n, \mathcal{L}^n, \doteq^n, \mathcal{M}^n)$ the fully expanded (and clash-free) completion graph that possibly also contains non-deterministic choices. Obviously, instead of starting with the initial completion graph G^0 to which no rule has (yet) been applied, we can use G^d to initialise a completion graph G for subsequent consistency tests which are, for example, required to prove or refute assumptions of higher level reasoning tasks. To be more precise, we extend G^d to G by the new individuals or by additional assertions to original individuals as required for a subsequent test and then we can apply the tableau expansion rules to G. Note, in order to be able to distinguish the nodes/nominals in the different completion graphs, we assume that all nodes/nominals that are newly created for G do not occur in existing completion graphs, such as G^d or G^n.

This re-use of G^d is an obvious and straightforward optimisation and it is already used by many state-of-the-art reasoning systems to successfully reduce the work that has to be repeated in subsequent consistency tests [17]. Especially if the knowledge base is deterministic, the tableau expansion rules only have to be applied for the newly added assertions in G. In principle, also G^n can be re-used instead of G^d [17], but this causes problems if non-deterministically derived facts of G^n are involved in new clashes. In particular, it is required to do backtracking in such cases, i.e., we have to jump back to the last version of the initial completion graph that does not contain the consequences of the last non-deterministic decision that is involved in the clash. Then, we have to continue the processing by choosing another alternative. Obviously, if we have to jump back to a very early version of the completion graph, then potentially many non-deterministic decisions must be re-processed. Moreover, after jumping back, we also have to add and re-process the newly added individuals and/or assertions.

To improve the handling of non-deterministic knowledge bases, our approach uses criteria to check whether nodes in G (or the nodes in a completion graph G' obtained from G by further rule applications) are "cached", i.e., there exist corresponding nodes in the cached completion graph of the initial consistency test and, therefore, it is not required to process them again. These caching criteria check whether the expansion of nodes is possible as in the cached completion graph G^n without influencing modified nodes in G', thus, only the processing of new and modified nodes is required.

Definition 1 (Caching Criteria). *Let $G^d = (V^d, E^d, \mathcal{L}^d, \doteq^d, \mathcal{M}^d)$ be a completion graph with only deterministically derived consequences and $G^n = (V^n, E^n, \mathcal{L}^n, \doteq^n, \mathcal{M}^n)$ a fully expanded and clash-free expansion of G^d. Moreover, let G be an extension of G^d and $G' = (V', E', \mathcal{L}', \doteq', \mathcal{M}')$ a completion graph that is obtained from G by applying tableau expansion rules.*

A node $v' \in V'$ *is* cached in G' *if caching of the node is not invalid, where the* caching is invalid *(we then also refer to the node as non-cached) if*

- C1 $v' \notin V^d$ *or* $\mathsf{mergedTo}^{\mathcal{M}^n}(v') \notin V^n$*;*
- C2 $\mathcal{L}'(v') \not\subseteq \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v'))$*;*
- C3 $\forall r.C \in \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v'))$ *and there is an r-neighbour node* w' *of* v' *such that* w' *is not cached and* $C \notin \mathcal{L}(w')$*;*
- C4 $\leqslant m\, r.C \in \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v'))$ *and the number of the non-cached r-neighbour nodes of* v' *without* $\mathsf{nnf}(\neg C)$ *in their labels together with the r-neighbours in* G^n *of* $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *with* C *in their labels is greater than* m*;*
- C5 $\exists r.C \in \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v'))$ *and every r-neighbour node* w' *of* v' *with* $C \notin \mathcal{L}(w')$ *and* $C \in \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(w'))$ *is not cached;*
- C6 $\geqslant m\, r.C \in \mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v'))$ *and the number of r-neighbour nodes* $w_1^n, \ldots, w_k^n$ *of* $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *with* $C \in \mathcal{L}^n(w_1^n), \ldots, C \in \mathcal{L}^n(w_k^n)$*, for which there is either no node* $w_i' \in V'$ *with* $\mathsf{mergedTo}^{\mathcal{M}^n}(w_i') = w_i^n$ *or* w_i' *with* $\mathsf{mergedTo}^{\mathcal{M}^n}(w_i') = w_i^n$ *and* $C \notin \mathcal{L}(w_i')$ *is not cached for* $1 \leq i \leq k$*, is less than* m*;*
- C7 $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *is a nominal node with* $\leqslant m\, r.C$ *in its label and there exists a blockable and non-cached* $\mathsf{inv}(r)$*-predecessor node* w' *of* v' *with* $\mathsf{nnf}(\neg C) \notin \mathcal{L}(w')$*;*
- C8 $\mathsf{mergedTo}^{\mathcal{M}^n}(w')$ *is an r-neighbour node of* $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *such that* w' *is not cached and* w' *is not an r-neighbour node of* v'*;*
- C9 $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *has a successor node* u^n *such that* u^n *or a descendant of* u^n *has a successor node* $\mathsf{mergedTo}^{\mathcal{M}^n}(w')$ *for which* $w' \in V'$*,* w' *is not cached, and there exists no node* $u' \in V'$ *with* $\mathsf{mergedTo}^{\mathcal{M}^n}(u') = u^n$*;*
- C10 $\mathsf{mergedTo}^{\mathcal{M}^n}(v')$ *is blocked by* $\mathsf{mergedTo}^{\mathcal{M}^n}(w')$ *and* $w' \in V'$ *is non-cached;*
- C11 *there is a non-cached node* $w' \in V'$ *and* $\mathsf{mergedTo}^{\mathcal{M}^n}(v') = \mathsf{mergedTo}^{\mathcal{M}^n}(w')$*; or*
- C12 *there is a node* $w' \in V'$ *such that* $v' \dot{\neq}' w'$ *and* $\mathsf{mergedTo}^{\mathcal{M}^n}(v') = \mathsf{mergedTo}^{\mathcal{M}^n}(w')$.

Conditions C1 and C2 ensure that a node also exists in the cached completion graph G^n and that its label is a subset of the corresponding label in G^n such that the same expansion is possible. C3 checks whether the expansion of a node would add a concept of the form $\forall r.C$ such that it could propagate C to a non-cached neighbour node. Analogously, C4 checks for potentially violated at-most cardinality restrictions by counting the new or modified neighbours in G' and the neighbours in G^n. C5 and C6 verify that existential and at-least cardinality restrictions are still satisfied if the cached nodes are expanded identically. C7 checks whether the NN-rule of the tableau algorithm would be applicable after the expansion, i.e., we check whether all potentially relevant neighbours in G' are nominal nodes. C8 checks whether the expansion would add additional roles to edge labels between cached and non-cached nodes, which could be problematic for disjoint roles. For C9: If a node w', for which caching is invalid, is connected to nodes in G^n that are only available in G^n (e.g., non-deterministically created ones), then we have to identify caching as invalid for those ancestors of these nodes that are also in G' such that these connections to w' can be re-built. Otherwise, we would potentially miss new consequences that could be propagated from w'. C10 is used to reactivate the processing of nodes for which the caching of the blocker node is invalid. C11 and C12 ensure that merging is possible as in G^n: C11 checks whether the node into which the node is merged is also cached and C12 ensures that there is no additional entry for $\dot{\neq}'$ that would cause a clash if the nodes were merged as in G^n.

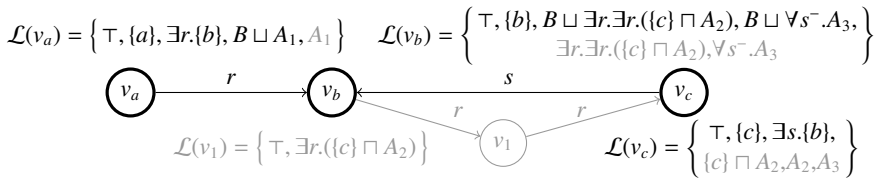


Fig. 1. Constructed and cached completion graph for Example 1 with deterministically (coloured black) and non-deterministically (coloured grey) derived facts

Since only those nodes can be cached, which are available in the "deterministic" completion graph G^d, it is important to maximise the deterministic processing of the completion graph. This can, for example, be achieved by processing the completion graph with deterministic rules only until the generation of new nodes is subset blocked. Subset blocking is not sufficient for more expressive DLs, but it prevents the expansion of too many successor nodes in case non-deterministic rule applications merge and prune some parts of the completion graph. Absorption techniques (e.g., [14,18,21]) are also essential since they reduce the overall non-determinism in an ontology.

Example 1. Assume that the tableau algorithm builds a completion graph as depicted in Figure 1 for testing the consistency of a knowledge base $\mathcal{K}$ containing the axioms

$$\{a\} \sqsubseteq B \sqcup A_1 \qquad \{b\} \sqsubseteq B \sqcup \exists r.\exists r.(\{c\} \sqcap A_2) \qquad \{c\} \sqsubseteq \exists s.\{b\}$$
$$\{a\} \sqsubseteq \exists r.\{b\} \qquad \{b\} \sqsubseteq B \sqcup \forall s^-.A_3.$$

The deterministic version of the completion graph, G^d, contains the elements coloured in black in Figure 1 and the non-deterministic version, G^n, additionally contains the elements coloured in grey. If we want to determine which individuals are instances of the concept $\exists r.\top$, then we have to check, for each individual i in $\mathcal{K}$, the consistency of $\mathcal{K}$ extended by $\mathsf{nnf}(\neg\exists r.\top)(i)$, which is equivalent to adding the axiom $\{i\} \sqsubseteq \forall r.\bot$. The individual a is obviously an instance of this concept, which can also be observed by the fact that expanding the extended completion graph adds $\forall r.\bot$ to $\mathcal{L}^d(v_a)$, which immediately results in a clash. In contrast, if we extend $\mathcal{L}^d(v_b)$ by $\forall r.\bot$, we have to process the disjunctions $B \sqcup \exists r.\exists r.(\{c\} \sqcap A_2)$ and $B \sqcup \forall s^-.A_3$. The disjunct $\exists r.\exists r.(\{c\} \sqcap A_2)$ would result in a clash and, therefore, we choose B, which also satisfies the second disjunction. Note that v_a and v_c do not have to be processed since v_a is cached, i.e., its expansion is possible in the same way as in G^n, and v_c does not have any concepts for which processing is required. Last but not least, we have to test whether $\mathcal{L}^d(v_c)$ extended by $\forall r.\bot$ is satisfiable. Now, the caching of v_c is obviously invalid (due to C2) and, therefore, also the caching of v_b is invalid: C3 can be applied for $\forall s^-.A_3$ and C9 for the successor node v_1 of v_b in G^n which also has the non-cached successor node v_c. Since v_a is still cached, we only have to process v_b again, which does, however, not result in a clash. Hence, only a is an instance of $\exists r.\top$.

Most conditions can be checked locally: For a non-cached node, we simply follow its edges w.r.t. G' and G^n to (potentially indirect) neighbour nodes that are also available in G^d. Exceptions are Conditions C10, C11, and C12, for which we can, however, simply trace back established blocking relations or $\mathcal{M}^n$ to find nodes for which the caching

criteria have to be checked. The implementation can also be simplified by keeping the caching of a node invalid once it has been identified as invalid (even if the caching becomes possible again after the node has been expanded identically). Instead of directly checking the caching criteria, the relevant nodes can also be stored in a set/queue to check the conditions when it is (more) convenient. Clearly, it depends on the ontology, whether it is more beneficial to test the caching criteria earlier or later. If the criteria are tested early, then we could unnecessarily re-process some parts of the cached completion graph since the application of (non-deterministic) expansion rules can satisfy some conditions. A later testing could cause a repeated re-processing of the same parts if a lot of backtracking is required and consequences of re-activated nodes are involved in clashes. Alternatively, one could learn for the entire ontology or for single nodes, whether the criteria should be checked earlier or later. Unfortunately, this cannot easily be realised for all reasoning systems since it requires that dependencies between derived facts are precisely tracked in order to identify nodes that are often involved in the creation of clashes and for which the criteria should be checked early.

It is also possible to non-deterministically re-use derived consequences from G^n, i.e., if the caching is invalid for a node v and $\mathsf{mergedTo}^{\mathcal{M}^n}(v)$ is in G^n, then we can non-deterministically add the missing concepts from $\mathcal{L}^n(\mathsf{mergedTo}^{\mathcal{M}^n}(v))$ to $\mathcal{L}(v)$. Since the resulting completion graph is very similar to the cached one, caching can often be established quickly for many nodes. Of course, if some of the non-deterministically added concepts are involved in clashes, then we potentially have to backtrack and process the alternative where this node is ordinarily processed. Another nice side effect of storing G^n is that we can use the non-deterministic decisions from G^n as an orientation in subsequent consistency tests. By prioritising the processing of the same non-deterministic alternatives as for G^n, we can potentially find a solution that is very similar to G^n without exploring much of the search space.

4 Caching Extensions and Applications

In this section, we sketch additional applications of the caching technique, which allow for supporting nominals for the satisfiability caching of node labels and for reducing the incremental reasoning effort for changing ABoxes. Furthermore, we describe an extension that allows for caching (parts of) the completion graph in a representative way, whereby an explicit representation of many nodes in the completion graph can be avoided and, therefore, the memory consumption can be reduced.

Satisfiability Caching with Nominals: Caching the satisfiability status of labels of (blockable) nodes in completion graphs is an important optimisation technique for tableau-based reasoning systems [3,4]. If one obtains a fully expanded and clash-free completion graph, then the contained node labels can be cached and, if identical labels occur in other completion graphs, then their expansion (i.e., the creation of required successor nodes) is not necessary since their satisfiability has already been proven. Of course, for more expressive DLs that include, for example, inverse roles and cardinality restrictions, we have to consider pairs of node labels as well as the edge labels between these nodes. Unfortunately, this kind of satisfiability caching does not work for DLs with nominals. In particular, connections to nominal nodes can be used to propagate

new concepts from one blockable node in the completion graph to any other blockable node, whereas for DLs without nominals, the consequences can only be propagated from or to successor nodes. Hence, the caching of node labels is not easily possible and many reasoners deactivate this kind of caching for knowledge bases with nominals.

However, in combination with completion graph caching, we re-gain the possibility to cache some labels of (blockable) nodes for knowledge bases with nominals. Roughly speaking, we first identify which nodes "depend on" which nominals, i.e., which nominal nodes are existentially quantified as a successor/descendant for a node. The labels of such nodes can then be cached together with the dependent nominals, i.e., with those nominals on which the nodes depend, if all nodes for the dependent nominals are still cached w.r.t. the initial completion graph. These cache entries can be re-used for nodes with identical labels in subsequent completion graphs as long as the completion graph caching of the nodes for the dependent nominals is not invalid. Of course, the blocked processing of nodes due to matching cache entries has to be reactivated if the completion graph caching of a node for a dependent nominal becomes invalid. Moreover, we cannot cache the labels of blockable nodes that depend on newly generated nominals since possible interactions over these nodes are not detected.

Incremental Reasoning for Changing ABoxes: Many reasoning systems re-start reasoning from scratch if a few axioms in the knowledge base have been changed. However, it is not very likely that a few changes in the ABox of a knowledge base have a huge impact on reasoning. In particular, many ABox assertions only propagate consequences to the local neighbourhood of the modified individuals and, therefore, the results of reasoning tasks such as classification are often not affected, even if nominals are used in the knowledge base. With the presented completion graph caching, we can easily track which nodes from the cached completion graph are modified in subsequent tests and for which caching is invalidated to perform higher level reasoning tasks. Hence, if the changed ABox assertions have only a known, locally limited influence, then the reasoning effort for many (higher level) reasoning tasks can be reduced by checking whether a tracked node is affected by the changes. To detect the influences of the changes, an incremental consistency check can be performed where all modified individuals and their neighbours are deterministically re-constructed step-by-step until "compatibility" with the previous deterministic completion graph is achieved, i.e., the same deterministic consequences are derived for the nodes as in the previous completion graph. The non-deterministic version of the new completion graph can then be obtained by continuing the (non-deterministic) processing of the re-constructed nodes and by using the presented completion graph caching for all remaining nodes. Hence, we can identify the influenced nodes by comparing the newly obtained completion graph with the previous one. Compared to other incremental consistency checking approaches (e.g., [9]), the re-construction of changed parts supported by the completion graph caching enables a better handling of non-determinism and does not require a memory intensive tracking of which consequences are caused by which ABox assertions. The idea of tracking those parts of a completion graph that are potentially relevant/used for the calculation of higher level reasoning tasks and comparing them with the changed parts in new completion graphs has already been proposed for answering conjunctive queries under incremental ABox updates [8], but the completion graph caching simplifies the

Table 1. Ontology metrics for selected benchmark ontologies (A stands for Axioms, C for Classes, P for Properties, I for Individuals, CA for Class Assertions, OPA for Object Property Assertions, and DPA for Data Property Assertions)

Ontology	Expressivity	#A	#C	#P	#I	#CA	#OPA	#DPA
OGSF	$\mathcal{SROIQ}(\mathcal{D})$	1,235	386	179	57	45	58	20
Wine	$\mathcal{SHOIN}(\mathcal{D})$	1,546	214	31	367	409	492	2
DOLCE	$\mathcal{SHOIN}$	1,667	209	317	42	101	36	0
OBI	$\mathcal{SROIQ}(\mathcal{D})$	28,770	3,549	152	161	273	19	1
USDA-5	$\mathcal{ALCIF}(\mathcal{D})$	1,401	30	147	1,214	1,214	12	0
COSMO	$\mathcal{SHOIN}(\mathcal{D})$	29,655	7,790	941	7,817	8,675	3,240	665
DPC1	$\mathcal{ALCIF}(\mathcal{D})$	55,020	1,920	94	28,023	15,445	39,453	0
UOBM-1	$\mathcal{SHOIN}(\mathcal{D})$	260,728	69	44	25,453	46,403	143,549	70,628

realisation of this technique and significantly reduces the overhead for identifying those parts of higher level reasoning tasks that have to be re-computed. Moreover, with the completion graph caching, also very expressive DLs such as $\mathcal{SROIQ}$ can be supported.

Representative Caching: In order to reduce the memory consumption for caching the completion graph, the technique can be adapted such that all relevant data is stored in a representative way, which allows for building "local" completion graphs for small subsets of the entire ABox until the existence of a complete completion graph considering all individuals can be guaranteed. To be more precise, if a fully expanded and clash-free completion graph is constructed for a subset of the ABox (e.g., a subset of all individuals and their assertions), then we extract and generalise information from the processed individuals and store them in a representative cache. If we then try to build a completion graph for another subset of the ABox that has some overlapping with a previously handled subset (e.g., role assertions for which edges to previous handled individuals have to be created), then we load the available data from the cache and continue the processing of the overlapping part until it is "compatible", i.e., the expansion of the remaining individuals in the cache can be guaranteed as in the previously constructed completion graphs. Of course, this only works well for knowledge bases for which there is not too much non-deterministic interaction between the separately handled ABox parts. Moreover, compared to the ordinary completion graph caching, we are clearly trading a lower memory consumption against an increased runtime since more work potentially has to be repeated to establish compatibility.

5 Implementation and Evaluation

The completion graph caching introduced in Section 3 is integrated in our reasoning system Konclude [20] and we selected several well-known benchmark ontologies (cf. Table 1) for the evaluation of the presented techniques. The evaluation was carried out on a Dell PowerEdge R420 server running with two Intel Xeon E5-2440 hexa core processors at 2.4 GHz with Hyper-Threading and 144 GB RAM under a 64bit Ubuntu 12.04.2 LTS. In order to make the evaluation independent of the number of CPU cores, we used only one worker thread for Konclude. We ignored the time spent for parsing ontologies and writing results and we used a time limit of 5 minutes, i.e., 300 seconds.

Table 2. Reasoning times for different completion graph caching techniques (in seconds)

Ontology	Prep.+ Cons.	Classification				Realisation			
		No-C	Det-C	ET-C	LT-C	No-C	Det-C	ET-C	LT-C
OGSF	0.0	3.8	1.0	0.2	0.2	0.1	0.0	0.0	0.0
Wine	0.0	49.5	29.6	0.8	0.8	49.1	25.8	0.2	0.1
OBI	0.2	65.9	19.2	1.5	1.5	2.2	2.0	0.0	0.1
DOLCE	0.0	6.7	1.1	0.2	0.2	≥ 300.0	5.2	0.1	0.1
USDA-5	4.1	0.8	1.0	1.0	0.8	≥ 300.0	38.7	20.5	20.2
COSMO	0.6	≥ 300.0	≥ 300.0	42.2	11.2	n/a	n/a	11.2	19.9
DPC1	6.5	0.1	0.2	0.1	0.1	≥ 300.0	53.3	19.4	20.5
UOBM-1	6.7	240.6	4.8	1.3	1.1	≥ 300.0	≥ 300.0	≥ 300.0	≥ 300.0

Table 2 shows the reasoning times for consistency checking (including preprocessing), classification, and realisation (in seconds) with different completion graph caching techniques integrated in Konclude. Please note that the class hierarchy is required to realise an ontology, i.e., classification is a prerequisite of realisation, and, analogously, consistency checking as well as preprocessing are prerequisites of classification. Thus, realisation cannot be performed if the time limit is already reached for classification.

If no completion graph caching is activated (No-C), then the realisation and classification can often require a large amount of time since Konclude has to re-process the entire ABox for all instance and subsumption tests (if the ontology uses nominals). For several ontologies, such as Wine and DOLCE, the caching and re-use of the deterministic completion graph from the consistency check (Det-C) already leads to significant improvements. Nevertheless, with the two variants ET-C and LT-C of the presented completion graph caching technique, where ET-C uses an "early testing" and LT-C a "late testing" of the defined caching criteria, Konclude can further reduce the reasoning times. In particular, with both completion graph caching techniques, all evaluated ontologies can easily be classified and also the realisation can be realised efficiently for all but UOBM-1. Table 2 also reveals that only for COMSO there is a remarkable difference between ET-C and LT-C, where this difference can be explained by the fact that there is often more interaction with the individuals from the ABox for instance tests than for satisfiability and subsumption tests and, therefore, ET-C can be better for realisation due to the potentially lower effort in combination with backtracking.

The effects of the different completion graph caching techniques can also be observed for the OWL DL Realisation dataset of the ORE 2014 competition,[3] which contains several ontologies with non-deterministic language features and non-trivial ABoxes. By excluding 1,331 s spent for preprocessing and consistency checking, the accumulated classification times over the contained 200 ontologies are 3,996 s for the version No-C, 2,503 s for Det-C, 1,801 s for ET-C, and 1,606 s for LT-C. Clearly, the dataset also contains many ontologies for which completion graph caching does not seem to have a significant impact, for example, if the ontologies can be processed deterministically or the ontologies are too difficult to even perform consistency checking (which is the case for 3 ontologies). Nevertheless, our completion graph caching improves the classification time with similar performances for LT-C and ET-C. By using

[3] http://www.easychair.org/smart-program/VSL2014/ORE-index.html

Table 3. Incremental reasoning effort for different reasoning tasks on changed ABoxes

Ontology	$\frac{\lvert\Delta\rvert \cdot 100}{\lvert\mathcal{K}\rvert}$	Consistency			Classification			Realisation		
		Time [s]		changed	Time [s]		reclassified	Time [s]		recomp. indi-
		$\mathcal{K}$	$\mathcal{K}^{\mp\Delta}$	nodes [%]	$\mathcal{K}$	$\mathcal{K}^{\mp\Delta}$	classes [%]	$\mathcal{K}$	$\mathcal{K}^{\mp\Delta}$	viduals [%]
USDA-5	1	3.1	0.0	0.0	0.9	–	–	18.2	0.0	1.0
USDA-5	2	3.2	0.0	0.0	0.8	–	–	14.9	0.0	2.0
USDA-5	4	3.2	0.1	0.0	0.8	–	–	17.0	0.0	3.9
COSMO	1	0.4	0.0	3.1	24.4	17.4	73.0	0.4	0.2	3.1
COSMO	2	0.4	0.1	5.5	25.9	18.0	73.0	0.5	0.3	5.6
COSMO	4	0.4	0.1	9.9	25.2	18.6	72.8	0.6	0.4	10.0
DPC1	1	5.0	0.6	1.5	0.1	–	–	29.1	14.0	10.0
DPC1	2	4.9	1.1	2.6	0.1	–	–	27.7	19.9	16.9
DPC1	4	5.0	2.0	4.4	0.1	–	–	29.3	26.7	26.5
UOBM-1	1	3.6	2.3	10.0	1.3	0.8	5.9	$\geq$ 300.0		n/a
UOBM-1	2	3.7	2.9	14.0	1.4	1.1	7.9	$\geq$ 300.0		n/a
UOBM-1	4	3.7	3.5	18.2	1.4	1.7	11.3	$\geq$ 300.0		n/a

the satisfiability caching extension for nominals, as presented in Section 4, the accumulated classification time can be further improved to 725 s. Similar results are also achieved for the realisation of these ontologies. By excluding the times for all prerequisites, the accumulated realisation times over all 200 ontologies are 1,740 s for No-C, 1,498 s for Det-C, 1,061 s for ET-C, 1,256 s for LT-C, and 923 s for the version where the satisfiability caching extension for nominals is additionally activated.

Incremental Reasoning Experiments: To test the incremental reasoning based on the presented completion graph caching, we used those ontologies of Table 1 that have a large amount of ABox assertions and for which Konclude still has a clearly measurable reasoning time, i.e., USDA-5, COSMO, DPC1, and UOBM-1. We simulated a changed ABox for these ontologies by randomly removing a certain amount of assertions from the ontology (denoted by $\mathcal{K}$) and by re-adding the removed assertions and removing new assertions (denoted by $\mathcal{K}^{\mp\Delta}$). For each ontology, we evaluated 10 random modifications that have 1, 2, and 4 % of the size of the ontology's ABox. For simplicity, all modifications have the same amount of removed and added assertions. The obtained results for the presented incremental reasoning approach are shown in Table 3.

For consistency, the first two columns show the (incremental) consistency checking time (in seconds) for $\mathcal{K}$ and $\mathcal{K}^{\mp\Delta}$, respectively, and the third column shows the percentage of the nodes in the completion graph for $\mathcal{K}$ that has been changed for the application of the modification. It can be observed that, especially for smaller modifications, the incremental consistency check often requires much less time than the initial consistency check. In particular, the individuals of USDA-5 are sparsely connected via object properties and, therefore, often only the modified individuals have to be re-built. For larger modifications, the incremental consistency checking time increases significantly for some ontologies, e.g., UOBM-1, and does almost catch up to the consistency checking time for the initial ontology. On the one hand, our incremental consistency checking approach clearly has some additional overhead due to the fact that nodes are re-built step by step until an expansion as for the initial completion graph can be guaran-

teed, but, on the other hand, our prototypical implementation has still a lot of room for improvements. For example, we currently also re-build nodes for individuals for which only new assertions have been added although it would be sufficient to simply extend the nodes of the previous deterministic completion graph by the new consequences.

For classification, the first two columns show analogously the (incremental) classification time for $\mathcal{K}$ and $\mathcal{K}^{\mp\mathcal{A}}$, respectively, and the third column represents the percentage of the classes for which satisfiability and subsumption tests were re-calculated. At the moment, the used/modified nodes from the cached completion graph are tracked together for all satisfiability and subsumption tests and we mark those classes for which nodes have been tracked. It can be observed that for the ontologies with nominals (e.g., UOBM-1), only a few classes have to be re-classified and, in several cases, re-classification is not required at all.

Also for realisation, the first two columns show the (incremental) realisation time for $\mathcal{K}$ and $\mathcal{K}^{\mp\mathcal{A}}$, respectively, and the last column shows the percentage of the number of individuals that are potentially affected by the changes and for which the (possible) types have to be re-computed. For this, we separately track, for each individual, the nodes of the cached completion graph that are used/modified by the instance tests.

Representative Caching Experiments: We integrated a first prototypical version of the presented representative caching in our reasoning system Konclude, which is, however, not yet compatible with all other integrated features and optimisations. As of now, the integrated representative caching is primarily used for "simple individuals" that do not have too much interaction with other individuals in the ABox. In cases where representative caching could easily cause performance deficits (e.g., through the intensive use of nominals), Konclude caches the relevant parts of such completion graphs by using the ordinary technique. Moreover, data property assertions are, at the moment, internally transformed into class assertions and, as a consequence, nodes for individuals with data property assertions can currently not be representatively cached. However, first experiments are very encouraging. For example, Homo_sapiens is a very large $\mathcal{SROIQ}$ ontology from the Oxford ontology library with $244,232$ classes, 255 object properties, and $289,236$ individuals for which Konclude requires $10,211$ MB in total to check the consistency by using representative caching, whereas $19,721$ MB are required by Konclude for consistency checking with a fully expanded completion graph. Note that a large amount ($9,875$ MB) of the required memory is used for the (unoptimised) internal representation, the data from the preprocessing, and the parsed OWL objects which are kept in memory to facilitate the handling of added/removed axioms.

6 Conclusions

We have presented a refinement of the completion graph caching technique that improves ABox reasoning also for very expressive Description Logics through a more sophisticated handling of non-deterministic consequences. In addition, we sketched extensions and applications of the caching technique, which allow for supporting nominals for the satisfiability caching of node labels, for reducing the incremental reasoning effort for changing ABoxes, and for handling very large ABoxes by storing partially processed parts of the completion graph in a representative way.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
2. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Scalable highly expressive reasoner (SHER). J. of of Web Semantics 7(4), 357–361 (2009)
3. Donini, F.M., Massacci, F.: EXPTIME tableaux for $\mathcal{ALC}$. J. of Artificial Intelligence 124(1), 87–138 (2000)
4. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. of Automated Reasoning 53(3), 1–25 (2014)
5. Glimm, B., Kazakov, Y., Liebig, T., Tran, T.K., Vialard, V.: Abstraction refinement for ontology materialization. In: Proc. 13th Int. Semantic Web Conf. (ISWC'14). LNCS, vol. 8797, pp. 180–195. Springer (2014)
6. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. J. of Automated Reasoning 41(2), 99–142 (2008)
7. Haarslev, V., Möller, R., Turhan, A.Y.: Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In: Proc. 1st Int. Joint Conf. on Automated Reasoning (IJCAR'01). LNCS, vol. 2083, pp. 61–75. Springer (2001)
8. Halaschek-Wiener, C., Hendler, J.: Toward expressive syndication on the web. In: Proc. 16th Int. Conf. on World Wide Web (WWW'07). ACM (2007)
9. Halaschek-Wiener, C., Parsia, B., Sirin, E.: Description logic reasoning with syntactic updates. In: Proc. 4th Confederated Int. Conf. On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE(OTM'06), LNCS, vol. 4275, pp. 722–737. Springer (2006)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
11. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. of Logic and Computation 9(3), 385–410 (1999)
12. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. Artificial Intelligence 160(1), 79–104 (2004)
13. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. of Automated Reasoning 39(3), 249–276 (2007)
14. Hudek, A.K., Weddell, G.E.: Binary absorption in tableaux-based reasoning for description logics. In: Proc. 19th Int. Workshop on Description Logics (DL'06). vol. 189. CEUR (2006)
15. Kollia, I., Glimm, B.: Optimizing SPARQL query answering over OWL ontologies. J. of Artificial Intelligence Research 48, 253–303 (2013)
16. Simančík, F.: Elimination of complex RIAs without automata. In: Proc. 25th Int. Workshop on Description Logics (DL'12). CEUR Workshop Proceedings, vol. 846. CEUR-WS.org (2012)
17. Sirin, E., Cuenca Grau, B., Parsia, B.: From wine to water: Optimizing description logic reasoning for nominals. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 90–99. AAAI Press (2006)
18. Steigmiller, A., Glimm, B., Liebig, T.: Optimised absorption for expressive description logics. In: Proc. 27th Int. Workshop on Description Logics (DL'14). vol. 1193. CEUR (2014)
19. Steigmiller, A., Glimm, B., Liebig, T.: Completion graph caching extensions and applications for expressive description logics. Tech. rep., Ulm University, Ulm, Germany (2015), available online at `https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Publikationen/2015/DL2015CGCTR.pdf`

20. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. J. of Web Semantics 27(1) (2014)
21. Tsarkov, D., Horrocks, I.: Efficient reasoning with range and domain constraints. In: Proc. 17th Int. Workshop on Description Logics (DL'04). vol. 104. CEUR (2004)
22. Wandelt, S., Möller, R.: Towards ABox modularization of semi-expressive description logics. Applied Ontology 7(2), 133–167 (2012)
23. Wu, J., Hudek, A.K., Toman, D., Weddell, G.E.: Absorption for ABoxes. J. of Automated Reasoning 53(3), 215–243 (2014)

On the Utility of $\mathcal{CFDI}_{nc}^{\forall-}$

David Toman and Grant Weddell

Cheriton School of Computer Science
University of Waterloo, Canada
{david,gweddell}@cs.uwaterloo.ca

Abstract. We consider the description logic $\mathcal{CFDI}_{nc}^{\forall-}$, a feature-based dialect that allows capturing value restrictions, a variety of identification constraints, and unqualified feature inverses. We introduce PTIME algorithms for various reasoning tasks in this logic, such as knowledge base consistency and logical implication and discuss the necessity of restrictions over $\mathcal{CFDI}_{nc}^{\forall}$ to maintain tractability. We then show how $\mathcal{CFDI}_{nc}^{\forall-}$'s modeling capabilities make it suitable for capturing relational and object-relational data sources (including of n-ary relations) in a natural way. In addition, we show that $\mathcal{CFDI}_{nc}^{\forall-}$ can simulate reasoning in DL-Lite$_{\text{core}}^{\mathcal{F}}$. We also discuss an approach to capturing a limited variant of role hierarchies within $\mathcal{CFDI}_{nc}^{\forall-}$.

1 Introduction

We have been developing the $\mathcal{CFD}$ family of feature-based *description logic* (DL) dialects that are designed primarily to support efficient PTIME reasoning services about object relational data sources. The dialects are notable for their ability to support terminological cycles with universal restrictions over functional roles together with a rich variety of functional constraints such as keys and functional dependencies over functional role paths.

The dialect $\mathcal{CFD}$ was the first member of this family, initially proposed in [8]. In [16], the authors show that reasoning about logical consequence remains in PTIME when concept conjunction is allowed on left-hand-sides of inclusion dependencies, but that this is no longer the case should a variety of other concept constructors also be allowed. In particular, it was shown that adding inverse features in posed questions alone made reasoning about logical consequence in $\mathcal{CFD}$ intractable.

The dialect $\mathcal{CFD}_{nc}$ was subsequently introduced in which negation on right-hand-sides of inclusion dependencies replaced the ability to have conjunction on left-hand-sides [17]. This allowed the capture of so-called disjointness constraints, and also made it possible to support additional reasoning services in PTIME, notably conjunctive query answering. These results generalize to $\mathcal{CFD}_{nc}^{\forall}$ knowledge bases in which universal restrictions are also permitted on left-hand-sides of inclusion dependencies [18].

An earlier version of this paper has appeared as [19].

In this paper, we consider the dialect $\mathcal{CFDI}^{\forall}_{nc}$ which extends $\mathcal{CFD}^{\forall}_{nc}$ with an ability to have unqualified inverse features in inclusion dependencies, and also introduce a less general dialect $\mathcal{CFDI}^{\forall-}_{nc}$ in which a given $\mathcal{CFDI}^{\forall}_{nc}$ TBox is presumed to satisfy additional syntactic restrictions. The restrictions relate to combinations of value restrictions and inverses and to combinations of value restrictions and path functional dependencies. A Summary of our main results concerning reasoning in $\mathcal{CFDI}^{\forall-}_{nc}$, in particular PTIME algorithms for both logical consequence and for knowledge base consistency for $\mathcal{CFDI}^{\forall-}_{nc}$ knowledge bases.

For the remainder the paper, we give an overview of a number of applications of $\mathcal{CFDI}^{\forall-}_{nc}$, starting with how it can be used to address issues relating to relational data sources over database schema that can include arbitrary combinations of functional dependencies and unary inclusion dependencies. We also show how the task of evaluating instance queries over RDF data sources based on a DL-Lite$^{\mathcal{F}}_{\text{core}}$ entailment regime can be reduced to reasoning about $\mathcal{CFDI}^{\forall-}_{nc}$ knowledge base consistency. Note that the DL dialect DL-Lite$^{\mathcal{F}}_{\text{core}}$ is of particular relevance to the W3C OWL 2 QL profile. A discussion of related work and future directions then follow in Section 6.

2 The Description Logics $\mathcal{CFDI}^{\forall}_{nc}$ and $\mathcal{CFDI}^{\forall-}_{nc}$

All members of the $\mathcal{CFD}$ family of DLs are fragments of FOL with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *features*. Note that incorporating features deviates from normal practice to use binary predicate symbols called *roles*. However, as we shall see, features make it easier to incorporate concept constructors suited to the capture of relational data sources that include various dependencies by a straightforward reification of n-ary predicates. Thus, e.g., a role R can be reified as a primitive concept R_C and two features $domR$ and $ranR$ in $\mathcal{CFDI}^{\forall}_{nc}$ or $\mathcal{CFDI}^{\forall-}_{nc}$, and an inclusion dependency A $\sqsubseteq \forall R$.B can then be captured as an inclusion dependency $\forall domR$.A $\sqsubseteq \forall ranR$.B.

Definition 1 ($\mathcal{CFDI}^{\forall}_{nc}$ Knowledge Bases) Let F, PC and IN be disjoint sets of (names of) features, primitive concepts and individuals, respectively. A *path function* Pf is a word in F^* with the usual convention that the empty word is denoted by *id* and concatenation by ".". *Concept descriptions* C and D are defined by the grammars on the left-hand-side of Figure 1 in which occurrences of "A" denote primitive concepts. A concept "C : $\mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$" produced by the last production of the grammar for D is called a *path functional dependency* (PFD).

Metadata and data in a $\mathcal{CFDI}^{\forall}_{nc}$ knowledge base $\mathcal{K}$ are respectively defined by a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. Assume A $\in \mathsf{PC}$, C and D are arbitrary concepts given by the grammars in Figure 1, $\{\mathsf{Pf}_1, \mathsf{Pf}_2\} \subseteq \mathsf{F}^*$ and that $\{a, b\} \subseteq \mathsf{IN}$. Then $\mathcal{T}$ consists of a finite set of *inclusion dependencies* of the form C $\sqsubseteq$ D, and $\mathcal{A}$

Syntax	Semantics: "$(\cdot)^{\mathcal{I}}$"
$\mathrm{C} ::= \mathrm{A}$	$\mathrm{A}^{\mathcal{I}} \subseteq \triangle$
$\mid\ \forall \mathsf{Pf}.\mathrm{C}$	$\{x \mid \mathsf{Pf}^{\mathcal{I}}(x) \in \mathrm{C}^{\mathcal{I}}\}$
$\mid\ \exists f^{-1}$	$\{x \mid \exists y \in \triangle : f^{\mathcal{I}}(y) = x\}$
$\mathrm{D} ::= \mathrm{C}$	$\mathrm{C}^{\mathcal{I}} \subseteq \triangle$
$\mid\ \neg \mathrm{C}$	$\triangle \setminus \mathrm{C}^{\mathcal{I}}$
$\mid\ \forall \mathsf{Pf}.\mathrm{D}$	$\{x \mid \mathsf{Pf}^{\mathcal{I}}(x) \in \mathrm{D}^{\mathcal{I}}\}$
$\mid\ \exists f^{-1}$	$\{x \mid \exists y \in \triangle : f^{\mathcal{I}}(y) = x\}$
$\mid\ \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$	$\{x \mid \forall y \in \mathrm{C}^{\mathcal{I}} : (\bigwedge_{i=1}^{k} \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y)) \Rightarrow \mathsf{Pf}^{\mathcal{I}}(x) = \mathsf{Pf}^{\mathcal{I}}(y)\}$

Fig. 1. $\mathcal{CFDI}^{\forall}_{nc}$ Concepts.

consists of a finite set of facts in the form of *concept assertions* $\mathrm{A}(a)$, and *path function assertions* $\mathsf{Pf}_1(a) = \mathsf{Pf}_2(b)$. Any PFD occurring in $\mathcal{T}$ must also satisfy a *regularity* condition by adhering to one of the following two forms:

$$\mathrm{C} : \mathsf{Pf}\,.\,\mathsf{Pf}_1, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \to \mathsf{Pf} \quad \text{or} \quad \mathrm{C} : \mathsf{Pf}\,.\,\mathsf{Pf}_1, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}\,.g. \tag{1}$$

A PFD is a *key* if it adheres to the first of these forms.

Semantics is defined in the standard way with respect to an interpretation $\mathcal{I} = (\triangle, (\cdot)^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretation of primitive concepts A to be subsets of $\triangle$, features f to be total functions on $\triangle$, and individuals a to be elements of $\triangle$. The interpretation function is extended to path expressions by interpreting id, the empty word, as the identity function $\lambda x.x$, concatenation as function composition, and to derived concept descriptions C or D as defined in Figure 1.

An interpretation $\mathcal{I}$ satisfies an inclusion dependency $\mathrm{C} \sqsubseteq \mathrm{D}$ if $\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$, a concept assertion $\mathrm{A}(a)$ if $a^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$, and a path function assertion $\mathsf{Pf}_1(a) = \mathsf{Pf}_2(b)$ if $\mathsf{Pf}_1^{\mathcal{I}}(a^{\mathcal{I}}) = \mathsf{Pf}_2^{\mathcal{I}}(b^{\mathcal{I}})$. $\mathcal{I}$ satisfies a knowledge base $\mathcal{K}$ if it satisfies each inclusion dependency and assertion in $\mathcal{K}$. □

Condition (1) on occurrences of the PFD concept constructor distinguish, e.g., PFDs of the form $\mathrm{C} : f \to id$ and $\mathrm{C} : f \to g$ from PFDs of the form $\mathrm{C} : f \to g.h$, and are necessary on $\mathcal{CFD}$ alone to avoid both intractability of reasoning about logical consequence [9] and undecidability of reasoning about KB consistency [15]. Conversely, and as usual, allowing conjunction (resp. disjunction) on the right-hand-sides (resp. left-hand-sides) of inclusion dependencies is a simple syntactic sugar.

Finally, note that $\mathcal{CFDI}^{\forall}_{nc}$ does not assume the unique name assumption, but that its ability to express disjointness enables mutual inequality between all pairs of n individuals to be captured by introducing $O(n)$ new atomic concepts, concepts assertions and inclusion dependencies in a straightforward way.

Lemma 2 ($\mathcal{CFDI}^{\forall}_{nc}$ KB Normal Form) For every KB $(\mathcal{T}, \mathcal{A})$, there is an equi-satisfiable KB $(\mathcal{T}', \mathcal{A}')$ in which subsumptions in $\mathcal{T}'$ adhere to the following

forms:

$$\mathrm{A} \sqsubseteq \mathrm{B}, \;\; \mathrm{A} \sqsubseteq \forall f.\mathrm{B}, \;\; \forall f.\mathrm{A} \sqsubseteq \mathrm{B}, \;\; \mathrm{A} \sqsubseteq \exists f^{-1}, \text{ or } \mathrm{A} \sqsubseteq \mathrm{A}' : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf},$$

where A and A′ are primitive concepts and B is a primitive concept or a negation of a primitive concept, and where ABox $\mathcal{A}'$ contains only assertions of the form $\mathrm{A}(a)$, $f(a) = b$, and $a = b$. □

Obtaining $\mathcal{T}'$ and $\mathcal{A}'$ from an arbitrary knowledge base $\mathcal{K}$ that are linear in the size of $\mathcal{K}$ is easily achieved by a straightforward conservative extension using auxiliary names for intermediate concept descriptions and individuals. For further details, see the definition of *simple concepts* in [13, 15].

Hereon, we identify $\neg\forall\,\mathsf{Pf}\,.A$ with $\forall\,\mathsf{Pf}\,.\neg A$, and say that $\forall\,\mathsf{Pf}\,.\mathrm{A}$ and $\forall\,\mathsf{Pf}\,.\neg\mathrm{A}$ are *complementary* for $\mathsf{Pf} \in \mathsf{F}^*$. Also, when a particular KB $(\mathcal{T}, \mathcal{A})$ is considered, we assume the sets PC and F contain symbols that appear in $\mathcal{T}$ and $\mathcal{A}$ only.

Unfortunately, use unqualified inverse features make reasoning about logical consequence over an arbitrary $\mathcal{CFDI}^{\forall}_{nc}$ KB $\mathcal{K}$ intractable [19]. To recover PTIME reasoning for both logical implication and KB consistency, $\mathcal{K}$ will need to satisfy additional syntactic restrictions.

Definition 3 ($\mathcal{CFDI}^{\forall-}_{nc}$ Knowledge Bases) A $\mathcal{CFDI}^{\forall-}_{nc}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a $\mathcal{CFDI}^{\forall}_{nc}$ KB in normal form that satisfies the following two conditions.

1. (*inverse feature and value restriction interaction*) If $\{\mathrm{A} \sqsubseteq \exists f^{-1}, \forall f.\mathrm{A}' \sqsubseteq \mathrm{B}\} \subseteq \mathcal{T}$ then (a) $\mathrm{A} \sqsubseteq \mathrm{A}' \in \mathcal{T}$, (b) $\mathrm{A}' \sqsubseteq \mathrm{A} \in \mathcal{T}$ or (c) $\mathrm{A} \sqsubseteq \neg\mathrm{A}' \in \mathcal{T}$.

2. (*inverse feature and PFD interaction*) Any PFD occurring in $\mathcal{T}$ must also satisfy a *stronger* regularity condition by adhering to one of the following two forms:

$$\mathrm{C} : \mathsf{Pf}\,.\,\mathsf{Pf}_1, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \quad \text{or} \quad \mathrm{C} : \mathsf{Pf}\,.f, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.g. \qquad (2)$$

Relaxing either of the conditions leads to EXPTIME and PSPACE completeness, respectively [19]. Note also, that the additional condition (2) imposed on PFDs applies only to non-key PFDs. Overall, however, such restrictions do not seem to impact the modeling utility of $\mathcal{CFDI}^{\forall}_{nc}$ in relation to keys and functional constraints. Indeed, we show that arbitrary functional dependencies in relational schema are easily captured.

3 $\mathcal{CFDI}^{\forall-}_{nc}$ TBoxes and Concept Satisfiability

It is easy to see that every $\mathcal{CFDI}^{\forall-}_{nc}$ TBox $\mathcal{T}$ is consistent (by setting all primitive concepts to be interpreted as the empty set). To test for *(primitive) concept satisfiability* we use the following construction:

Definition 4 (TBox Closure) Let $\mathcal{T}$ be a $\mathcal{CFDI}^{\forall-}_{nc}$ TBox in normal form. We define $\mathsf{Clos}(\mathcal{T})$ to be the least set of subsumptions that contains $\mathcal{T}$ and is closed under the following five inference rules:

1. $C_1 \sqsubseteq C_1 \in \mathsf{Clos}(\mathcal{T})$;
2. If $\{C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_3\} \subseteq \mathsf{Clos}(\mathcal{T})$, then $C_1 \sqsubseteq C_3 \in \mathsf{Clos}(\mathcal{T})$;
3. If $\{C_1 \sqsubseteq D_1, C_2 \sqsubseteq D_2\} \subseteq \mathsf{Clos}(\mathcal{T})$ and D_1 and D_2 are complementary, then $C_1 \sqsubseteq \neg C_2 \in \mathsf{Clos}(\mathcal{T})$;
4. If $\mathrm{A} \sqsubseteq \mathrm{B} \in \mathsf{Clos}(\mathcal{T})$, then $\forall f.\mathrm{A} \sqsubseteq \forall f.\mathrm{B} \in \mathsf{Clos}(\mathcal{T})$; and
5. If $\{\mathrm{A} \sqsubseteq \exists f^{-1}, \forall f.\mathrm{A}' \sqsubseteq \forall f.\mathrm{B}, \mathrm{A} \sqsubseteq \mathrm{A}'\} \subseteq \mathsf{Clos}(\mathcal{T})$, then $\mathrm{A} \sqsubseteq \mathrm{B} \in \mathsf{Clos}(\mathcal{T})$,

where A is a primitive concept, B is a primitive concept or its negation, and where C_1, C_2, D_1, and D_2 are subconcepts of concepts in $\mathcal{T}$ or their negations. □

Note that $\mathsf{Clos}(\mathcal{T})$ is at most quadratic in $|\mathcal{T}|$. It is also easy to verify that each inclusion added to $\mathsf{Clos}(\mathcal{T})$ by the inferences (1-4) in Definition 4 is logically implied by $\mathcal{T}$. Also, a variant of the closure rule (5) is not needed for the case $\mathrm{A}' \sqsubseteq \mathrm{A}$ since we also have $\mathrm{A}' \sqsubseteq \exists f^{-1}$, nor it is needed in the case $\mathrm{A}' \sqsubseteq \neg\mathrm{A}$ since, in this case, the value restriction in the rule is satisfied vacuously.

Given $\mathsf{Clos}(\mathcal{T})$, an object o, and a primitive concept A, we define the following family of subsets of PC indexed by paths of features (and their inverses), starting from o, as follows:

1. $S_o = \{\mathrm{B} \mid \mathrm{A} \sqsubseteq \mathrm{B} \in \mathsf{Clos}(\mathcal{T})\}$;
2. $S_{f(x)} = \{\mathrm{B} \mid \mathrm{A} \sqsubseteq \forall f.\mathrm{B} \in \mathsf{Clos}(\mathcal{T})$ and $\mathrm{A} \in S_x\}$, when $f \in \mathsf{F}$ and x not of the form "$f^-(y)$"; and
3. $S_{f^-(x)} = \{\mathrm{B} \mid \forall f.\mathrm{A} \sqsubseteq \mathrm{B}$ and $\mathrm{A} \in S_x\}$, when $\mathrm{A}' \sqsubseteq \exists f^{-1} \in \mathsf{Clos}(\mathcal{T})$, $\mathrm{A}' \in S_x$, and x not of the form "$f(y)$".

We say that S_x is *defined* if it conforms to one of the three above cases, and that it is *consistent* if, whenever $\{\mathrm{A}, \mathrm{A}'\} \subseteq S_x$, $\mathrm{A} \sqsubseteq \neg\mathrm{A}' \notin \mathsf{Clos}(\mathcal{T})$.

Theorem 5 (Primitive Concept Satisfiability) Let $\mathcal{T}$ be a $\mathcal{CFDI}_{nc}^{\forall -}$ TBox in normal form and A a primitive concept description. Then A is satisfiable with respect to T if and only if $\mathrm{A} \sqsubseteq \neg\mathrm{A} \notin \mathsf{Clos}(\mathcal{T})$.

Proof (sketch): We build a model of $\mathcal{T}$ in which $o \in \mathrm{A}^{\mathcal{I}}$ for some $o \in \triangle$ as follows:

- $\triangle = \{x \mid S_x$ is defined$\}$;
- $f^{\mathcal{I}} = \{(x, f(x)) \mid S_{f(x)}$ is defined$\} \cup \{(f^-(x), x) \mid S_{f^-(x)}$ is defined$\}$; and
- $\mathrm{A}^{\mathcal{I}} = \{x \mid S_x$ is defined, $\mathrm{A} \in S_x\}$.

It is easy to see that, due to closure rules in Definition 4, all the defined sets S_x must be consistent. Otherwise, A ($\in S_0$) must be inconsistent, implying in turn that $\mathrm{A} \sqsubseteq \neg\mathrm{A} \in \mathsf{Clos}(\mathcal{T})$, a contradiction. Hence, $\mathcal{I} = (\triangle, .^{\mathcal{I}})$ is a model of $\mathcal{T}$ (it satisfies all dependencies in $\mathsf{Clos}(\mathcal{T})$) such that $o \in \mathrm{A}^{\mathcal{I}}$. □

Note that the model witnessing satisfiability of A does not contain any identical path agreements (beyond the trivial $id = id$) and hence vacuously satisfies all PFDs in $\mathcal{T}$.

The above theorem can be used to check satisfiability of complex (non-PFD) concepts; e.g., satisfiability of $\forall \mathsf{Pf}.B$ w.r.t. $\mathcal{T}$ can be tested by checking satisfiability of a new primitive concept A w.r.t. $\mathcal{T} \cup \{\mathrm{A} \sqsubseteq \forall \mathsf{Pf}.B\}$. It also provides a technique for checking satisfiability of finite conjunctions of primitive concepts with respect to $\mathcal{T}$:

Corollary 6 Let $\mathcal{T}$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ TBox in normal form and $\mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{k}$ primitive concepts. Then $\mathrm{A}_1 \sqcap \ldots \sqcap \mathrm{A}_\mathrm{k}$ is satisfiable with respect to $\mathcal{T}$ if and only if A is satisfiable with respect to $\mathcal{T} \cup \{\mathrm{A} \sqsubseteq \mathrm{A}_1, \ldots, \mathrm{A} \sqsubseteq \mathrm{A}_\mathrm{k}\}$, for A a fresh primitive concept. □

4 Knowledge Base Consistency and Logical Implication

We start with the problem of determining if a given $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base is consistent. This is resolved in a straightforward way with the following notion of an *interesting* path function and the subsequent definition of an ABox completion procedure.

Definition 7 Let $\mathcal{T}$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ TBox. We say that a path function Pf is *interesting in* $\mathcal{T}$ if it is a common prefix of all Pf_i in a PFD $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \in \mathcal{T}$. □

Definition 8 Let $(\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base. We define an ABox $\mathsf{completion}_{\mathcal{T}}(\mathcal{A})$ to be the least ABox $\mathcal{A}'$ such that $\mathcal{A} \subseteq \mathcal{A}'$ and $\mathcal{A}'$ is closed under the rules in Figure 2. □

Note that since $\mathcal{A}$ is in normal form, individuals can only be declared to be members of primitive concepts. Thus, a $\mathcal{CFDI}_{nc}^{\forall-}$ ABox alone cannot lead to inconsistency. Only when combined with a TBox does it become possible that certain conjunctions of primitive concepts must interpret as empty in every model, thus leading to KB inconsistency. This observation combined with Corollary 6 yields the following theorem:

Theorem 9 ($\mathcal{CFDI}_{nc}^{\forall-}$ KB consistency) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ KB (in normal form). Then $\mathcal{K}$ is consistent if and only if $\{\mathrm{A} \mid \mathrm{A}(a) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})\}$ is satisfiable with respect to $\mathsf{Clos}(\mathcal{T})$ for all objects a in $\mathcal{A}$. □

It is easy to verify that constructing $\mathsf{Clos}(\mathcal{T})$ and $\mathsf{completion}_{\mathcal{T}}(\mathcal{A})$ is polynomial in $|\mathcal{K}|$, and that testing for consistency implicitly contains Horn-SAT due to the presence of PFDs. Thus, we have the following:

Corollary 10 Consistency of $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases is complete for PTIME. □

ABox Equality Rules:

1. If $\{a = b, b = c\} \subseteq \mathcal{A}$, then $a = c \in \mathcal{A}$.
2. If $\{f(a) = b, b = c\} \subseteq \mathcal{A}$, then $f(a) = c \in \mathcal{A}$.
3. If $\{a = b, f(b) = c\} \subseteq \mathcal{A}$, then $f(a) = c \in \mathcal{A}$.
4. If $\{f(a) = b, f(a) = c\} \subseteq \mathcal{A}$, then $b = c \in \mathcal{A}$.
5. If $\{a = b, \mathrm{A}(a)\} \subseteq \mathcal{A}$, then $\mathrm{A}(b) \in \mathcal{A}$.

ABox–TBox Interactions:

6. If $\mathrm{A}(a) \in \mathcal{A}$ and $\mathrm{A} \sqsubseteq \mathrm{B} \in \mathsf{Clos}(\mathcal{T})$, then $\mathrm{B}(a) \in \mathcal{A}$.
7. If $\{\mathrm{A}(a), f(a) = b\} \subseteq \mathcal{A}$ and $\mathrm{A} \sqsubseteq \forall f.\mathrm{B} \in \mathsf{Clos}(\mathcal{T})$, then $\mathrm{B}(b) \in \mathcal{A}$.
8. If $\{\mathrm{A}(a), f(b) = a\} \subseteq \mathcal{A}$ and $\forall f.\mathrm{A} \sqsubseteq \mathrm{B} \in \mathsf{Clos}(\mathcal{T})$, then $\mathrm{B}(b) \in \mathcal{A}$.

ABox–Inverse Interactions:

9. If $\mathsf{Pf} = f_1 f_2 \cdots f_k$ is interesting in $\mathcal{T}$, $\mathrm{A}_0(a_0) \in \mathcal{A}$, a_0 is an object in the original ABox $\mathcal{A}$, and $\{\mathrm{A}_{i-1} \sqsubseteq \exists {f_i}^{-1}, \forall f_i.\mathrm{A}_{i-1} \sqsubseteq \mathrm{A}_i\} \subseteq \mathsf{Clos}(\mathcal{T})$ for $0 < i \leq k$, then $\{\mathrm{A}_i(a_i), f_i(a_{i-1}) = a_i\} \subseteq \mathcal{A}$.

ABox–PFD Interactions:

10. If $\{A(a), B(b)\} \subseteq \mathcal{A}$, $\{\mathsf{Pf}'_i(a) = c_i, \mathsf{Pf}'_i(b) = c_i\} \subseteq \mathcal{A}$ for $0 < i \leq k$, and $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \in \mathcal{T}$ such that Pf'_i is a prefix of Pf_i, then
    (a) $\{\mathsf{Pf}'(a) = c, \mathsf{Pf}'(b) = c\} \subseteq \mathcal{A}$ for Pf' a prefix of Pf,
    (b) If $\{\mathsf{Pf}(a) = c, \mathsf{Pf}(b) = d\} \subseteq \mathcal{A}$, then $c = d \in \mathcal{A}$, or
    (c) If Pf is of the form $\mathsf{Pf}''.f$ and $\{\mathsf{Pf}''(a) = c, \mathsf{Pf}''(b) = d\} \subseteq \mathcal{A}$, then $f(c) = e$ and $f(d) = e$ to $\mathcal{A}$ for a new individual e.

Fig. 2. ABox Completion Rules.

It is also straightforward to reduce logical implication for a $\mathcal{CFDI}^{\forall-}_{nc}$ TBox $\mathcal{T}$ to knowledge base consistency. Indeed, subsumptions between literals are directly present in $\mathsf{Clos}(\mathcal{T})$. Logical implication involving more general concept descriptions, such as PFDs, is reduced to knowledge base (in)consistency by encoding a counterexample as an ABox.

Theorem 11 Logical consequence for $\mathcal{CFDI}^{\forall-}_{nc}$ terminologies is complete for PTIME. □

5 Applications

We now introduce two major applications for $\mathcal{CFDI}^{\forall-}_{nc}$: capture of relational (and object-relational) database schemas and its ability to fully simulate DL-Lite$^{\mathcal{F}}_{\text{core}}$. We also show how *role hierarchies* can be partially accommodated by $\mathcal{CFDI}^{\forall-}_{nc}$.

5.1 Relational Data Sources and BCNF

There are a number of applications of $\mathcal{CFDI}_{nc}^{\forall-}$ in addressing issues that surface with relational data sources. To illustrate, we show how a relational schema (S, Σ) with relation symbols S and with functional dependencies and unary foreign keys Σ can be easily mapped to a $\mathcal{CFDI}_{nc}^{\forall-}$ terminology $\mathcal{T}_{(S,\Sigma)}$, and then exhibit a straightforward reduction of so-called *Boyce-Codd normal form* (BCNF) diagnosis to logical consequence over $\mathcal{T}_{(S,\Sigma)}$.

First the mapping: each $R(A_1 : D_1, \ldots, A_k : D_k)$ in S (i.e., a relation of arity k) is reified by mapping to the following inclusion dependencies in $\mathcal{T}_{(S,\Sigma)}$:

$$\begin{aligned} &\mathrm{C}_R \sqsubseteq \mathrm{C}_R : a_{R.A_1}, \ldots, a_{R.A_k} \rightarrow id \text{ and} \\ &\mathrm{C}_R \sqsubseteq \forall a_{R.A_i}.D_i, \text{ for each } 0 < i \leq k, \end{aligned}$$

where (a) C_R is a primitive concept for which an interpretation will be the *tuple objects* that correspond to tuples in R, (b) $a_{R.A_i}$ are features that yield values of fields in such tuples, and (c) D_i are primitive concepts standing for the *domains* of the features. In addition, for each pair of $R, R' \in S$, add to $\mathcal{T}_{(S,\Sigma)}$

$$\mathrm{C}_R \sqsubseteq \neg \mathrm{C}_{R'}.$$

Each functional dependency $R : A_{i_1}, \ldots, A_{i_k} \rightarrow A_{i_0}$ in Σ is then mapped to an inclusion dependency:

$$\mathrm{C}_R \sqsubseteq \mathrm{C}_R : a_{R.A_{i_1}}, \ldots, a_{R.A_{i_k}} \rightarrow a_{R.A_{i_0}},$$

and each unary inclusion dependency $R[A] \subseteq R'[A']$ in Σ to three inclusion dependencies, where A is a fresh primitive concept unique to $R[A] \subseteq R'[A']$:

$$\begin{aligned} \mathrm{C}_R &\sqsubseteq \forall a_{R.A}.\mathrm{A}, \\ \mathrm{A} &\sqsubseteq \exists {a_{R'.A'}}^{-1}, \text{ and} \\ \forall a_{R'.A'}.\mathrm{A} &\sqsubseteq \mathrm{C}_{R'}. \end{aligned}$$

BCNF diagnosis then translates to logical consequence in $\mathcal{CFDI}_{nc}^{\forall-}$ in a straightforward fashion:

Theorem 12 (Diagnosing BCNF for Relational Data Sources) Each relation R in (S, Σ) is in BCNF iff there is no set of features $\{a_{R.A_{i_0}}, \ldots, a_{R.A_{i_k}}\}$ in $\mathcal{T}_{(S,\Sigma)}$ such that

$$\mathcal{T}_{(S,\Sigma)} \models \mathrm{C}_R \sqsubseteq \mathrm{C}_R : a_{R.A_{i_1}}, \ldots, a_{R.A_{i_k}} \rightarrow a_{R.A_0}$$

but where

$$\mathcal{T}_{(S,\Sigma)} \not\models \mathrm{C}_R \sqsubseteq \mathrm{C}_R : a_{R.A_{i_1}}, \ldots, a_{R.A_{i_k}} \rightarrow id\,.$$

□

Note that this easily generalizes to the object-relational setting where the domain D_i of an attribute may now refer directly to R_i-objects, and where a generalization of binary decompositions called *pivoting* is the means of repairing violations of BCNF [3, 4].

An application in query optimization over relational data sources relates to SQL `distinct`-keyword elimination, that is, detecting where operations in query plans to remove duplicates can be safely eliminated [8]. Such rewrites can be reduced to knowledge based consistency problems in $\mathcal{CFDI}_{nc}^{\forall -}$ by using an ABox to encode simple selection conditions in SQL queries [7].

5.2 Encoding of DL-Lite

Another application of $\mathcal{CFDI}_{nc}^{\forall -}$ in a different setting relates to the problem of evaluating *basic graph patterns* in SPARQL with a presumed entailment regime defined by DL-Lite$_{\text{core}}^{\mathcal{F}}$, a DL dialect that is related to the W3C OWL 2 QL profile. Such tasks reduce fundamentally to instance checking problems which reduce, in turn, to knowledge base consistency problems in the standard way.

Our reduction is based on mapping a given DL-Lite$_{\text{core}}^{\mathcal{F}}$ knowledge base $\mathcal{K}$ to a $\mathcal{CFDI}_{nc}^{\forall -}$ knowledge base $M_{\mathcal{K}}$ as follows: for each role P in $\mathcal{K}$, we reify P by introducing a new primitive concept C_P and by adding the following key PFD to $M_{\mathcal{K}}$:

$$\mathrm{C}_P \sqsubseteq \mathrm{C}_P : \mathit{domP}, \mathit{ranP} \rightarrow \mathit{id}\,.$$

The following rules define the mapping of each inclusion dependency in $\mathcal{K}$ (in normal form [2]) and each ABox assertion in $\mathcal{K}$ to corresponding dependencies and assertions in $M_{\mathcal{K}}$:

$$\begin{array}{lcl}
\mathrm{A}_1 \sqsubseteq \mathrm{A}_2 & \mapsto & \{\mathrm{A}_1 \sqsubseteq \mathrm{A}_2\},\\
\mathrm{A}_1 \sqsubseteq \neg\mathrm{A}_2 & \mapsto & \{\mathrm{A}_1 \sqsubseteq \neg\mathrm{A}_2\},\\
\mathrm{A}_1 \sqsubseteq \exists P & \mapsto & \{\mathrm{A}_1 \sqsubseteq \exists \mathit{domP}^{-1}, \forall \mathit{domP}.\mathrm{A}_1 \sqsubseteq \mathrm{C}_P\},\\
\mathrm{A}_1 \sqsubseteq \exists P^- & \mapsto & \{\mathrm{A}_1 \sqsubseteq \exists \mathit{ranP}^{-1}, \forall \mathit{ranP}.\mathrm{A}_1 \sqsubseteq \mathrm{C}_P\},\\
\exists P \sqsubseteq \mathrm{A}_1 & \mapsto & \{\mathrm{C}_P \sqsubseteq \forall \mathit{domP}.\mathrm{A}_1\},\\
\exists P^- \sqsubseteq \mathrm{A}_1 & \mapsto & \{\mathrm{C}_P \sqsubseteq \forall \mathit{ranP}.\mathrm{A}_1\},\\
(\mathsf{func}\ P) & \mapsto & \{\mathrm{C}_P \sqsubseteq \mathrm{C}_P : \mathit{domP} \rightarrow \mathit{id}\},\\
(\mathsf{func}\ P^-) & \mapsto & \{\mathrm{C}_P \sqsubseteq \mathrm{C}_P : \mathit{ranP} \rightarrow \mathit{id}\},\\
a : \mathrm{A} & \mapsto & \{a : \mathrm{A}\} \text{ and}\\
P(a,b) & \mapsto & \{c_{a,b}^P : \mathrm{C}_P, \mathit{domP}(c_{a,b}^P) = a, \mathit{ranP}(c_{a,b}^P) = b\}.
\end{array}$$

This mapping yields the following as a straightforward consequence:

Theorem 13 (DL-Lite$_{\text{core}}^{\mathcal{F}}$ Reasoning) Let $\mathcal{K}$ be a DL-Lite$_{\text{core}}^{\mathcal{F}}$ KB. Then knowledge base consistency, logical implication, and instance checking with respect to $\mathcal{K}$ can be reduced to reasoning about KB consistency with respect to $M_{\mathcal{K}}$. □

On Role Hierarchies The reduction above is only defined for DL-Lite$_{\text{core}}^{\mathcal{F}}$. Indeed, it is well known that an (unrestricted) combination *functionality* with *role hierarchies*, e.g., DL-Lite$_{\text{core}}^{\mathcal{HF}}$, leads to intractability [2]. On the other hand,

the ability to reify roles seems to allow to capture a limited version of *role hierarchies*[1].

Example 14 Consider roles R and S and the corresponding primitive concepts C_R and C_S, respectively. In contrast to the development in the previous section, we assume that the domains and ranges of the reified roles are captured by the feature *dom* and *ran* (common to both the reified roles). Then we can capture subsumption and disjointness of these roles as follows:

$$\begin{array}{lcl} R \sqsubseteq S & \mapsto & \mathrm{C}_R \sqsubseteq \mathrm{C}_S, \mathrm{C}_R \sqsubseteq \mathrm{C}_S : \mathit{dom}, \mathit{ran} \to \mathit{id}, \\ R \sqcap S \sqsubseteq \bot & \mapsto & \mathrm{C}_R \sqsubseteq \neg \mathrm{C}_S, \mathrm{C}_R \sqsubseteq \mathrm{C}_S : \mathit{dom}, \mathit{ran} \to \mathit{id}, \end{array}$$

assuming that the reified role R (and analogously S) also satisfies the key constraint $\mathrm{C}_R \sqsubseteq \mathrm{C}_R : \mathit{dom}, \mathit{ran} \to \mathit{id}$. Role typing is achieved in a way analogous to DL-Lite$^{\mathcal{F}}_{\mathrm{core}}$.

Note, however, that such a reduction does *not* lend itself to capturing role hierarchies between roles and *inverses* of roles: this is due to fixing the (names of the) features *dom* and *ran*. Moreover, the condition introduced in Definition 3(1), that governs the interactions between inverse features and value restrictions, introduces additional interactions that interfere with (simulating) role hierarchies, in particular in cases when *mandatory participation* constraints are present.

Example 15 Consider roles R_1 and R_2 and the corresponding primitive concepts C_{R_1} and C_{R_2}, respectively, and associated constraints that declare typing for the roles,

$$\begin{array}{l} \mathrm{C}_{R_1} \sqsubseteq \forall \mathit{dom}.\mathrm{A}_1, \mathrm{C}_{R_1} \sqsubseteq \forall \mathit{ran}.\mathrm{B}_1, \mathrm{C}_{R_1} \sqsubseteq \mathrm{C}_{R_1} : \mathit{dom}, \mathit{ran} \to \mathit{id} \\ \mathrm{C}_{R_2} \sqsubseteq \forall \mathit{dom}.\mathrm{A}_2, \mathrm{C}_{R_2} \sqsubseteq \forall \mathit{ran}.\mathrm{B}_2, \mathrm{C}_{R_2} \sqsubseteq \mathrm{C}_{R_2} : \mathit{dom}, \mathit{ran} \to \mathit{id} \end{array}$$

originating, e.g., from an ER diagram postulating that entity sets A_i and B_i participate in a relationship R_i (for $i = 1, 2$). Now consider a situation where the participation of A_i in R_i is *mandatory* (expressed, e.g., as $\mathrm{A}_i \sqsubseteq \exists R_i$ in DL-Lite). This leads to the following constraints:

$$\mathrm{A}_1 \sqsubseteq \exists \mathit{dom}^{-1}, \forall \mathit{dom}.\mathrm{A}_1 \sqsubseteq \mathrm{C}_{R_1} \text{ and } \mathrm{A}_2 \sqsubseteq \exists \mathit{dom}^{-1}, \forall \mathit{dom}.\mathrm{A}_2 \sqsubseteq \mathrm{C}_{R_2}.$$

Condition (1) in Definition 3 then requires that one of

$$\mathrm{A}_1 \sqsubseteq \mathrm{A}_2, \mathrm{A}_2 \sqsubseteq \mathrm{A}_1, \text{ or } \mathrm{A}_1 \sqsubseteq \neg \mathrm{A}_2$$

are present in the TBox. The first (and second) conditions imply that $\mathrm{C}_{R_1} \sqsubseteq \mathrm{C}_{R_2}$ ($\mathrm{C}_{R_2} \sqsubseteq \mathrm{C}_{R_1}$, respectively). The third condition states that the domains of (the reified versions of) R_1 and R_2 are disjoint, hence the roles themselves must also be disjoint. Hence, in the presence of $\mathrm{C}_{R_1} \sqsubseteq \mathrm{C}_{R_2} : \mathit{dom}, \mathit{ran} \to \mathit{id}$, the concepts C_{R_1} and C_{R_2} must also be disjoint.

In this setting role hierarchies can be mapped to $\mathcal{CFDI}^{\forall}_{nc}$ are as follows:

[1] Unlike DL-Lite$^{(\mathcal{HF})}_{\mathrm{core}}$, that restricts the applicability of functional constraints in the presence of role hierarchies, we study what forms role hierarchies can be captured while retaining the ability to specify arbitrary keys and functional dependencies.

1. only primitive roles are supported,
2. for each pair of roles participating in the same role hierarchy, either one of the roles is a super-role of the other, or the roles' domains/ranges are disjoint.

The first restriction originates in the way (binary) roles are reified—by assigning canonically-named features. This prevents modeling constraints such as $R \sqsubseteq R^-$ (which would seem to require simple equational constraints for feature renaming). The second condition is essential to maintaining tractability of reasoning [19]. Note, however, that no such restriction is needed for roles that do *not* participate in the same role hierarchy; this is achieved by appropriate choice of names for the features *dom* and *ran* similarly to the development in Section 5.2.

One can, however, model object participation in sibling roles participating in a role hierarchy using *delegation* [1], leading to a more complex translation of role assertions to $\mathcal{CFDI}^{\forall-}_{nc}$:

Example 16 Consider roles R_1, R_2, and S involved in a role hierarchy $R_1 \sqsubseteq S$ and $R_2 \sqsubseteq S$. To assert that A objects must participate in both the roles R_i, for $i \in \{1, 2\}$, we first explicitly establish the domains of the roles (the same applies for ranges of roles),

$$\mathrm{DR}_i \sqsubseteq \exists dom^{-1}, \forall dom.\mathrm{DR}_i \sqsubseteq \mathrm{C}_{R_i}, \text{ and } \mathrm{C}_{R_i} \sqsubseteq \forall \mathrm{DR}_i..$$

Then, instead of asserting $\mathrm{A} \sqsubseteq \mathrm{DR}_i$ (which immediately leads to inconsistency due to our PTIME restrictions on roles) we assert $\mathrm{A} \sqsubseteq \forall f_{R_i}.\mathrm{DR}_i$ where the f_{R_i} images of an A object are the *delegates* used to participate in the roles R_i.

Last, the $\mathcal{CFDI}^{\forall}_{nc}$-based approach to role hierarchies can easily be extended to handling hierarchies of non-homogeneous relationships (again, via reification and appropriate naming of features) that originate, e.g., from relating the aggregation constructs via inheritance in the EER model [10, 11].

6 Related Work and Future Directions

Toman and Weddell have also proposed the $\mathcal{DLF}$ family of feature-based Boolean-complete DL dialects obtained by allowing arbitrary use of negation in concepts [12]. In particular, they have shown that allowing inverse features in such dialects makes reasoning about logical consequence undecidable [14]. They have also shown that allowing negation on left-hand-sides of inclusion dependencies in $\mathcal{CFDI}^{\forall}_{nc}$ leads to intractability, but that PTIME algorithms exist for reasoning about logical consequence and knowledge base consistency if a number of additional conditions are satisfied [20].

A variety of path based identification constraints have been proposed [5] together with analogous applications in relational schema diagnosis [6], although $\mathcal{CFDI}^{\forall-}_{nc}$ seems to provide a more natural and transparent approach to this problem.

References

1. M. Aksit, J.W. Dijkstra, and A. Tripathi. Atomic delegation: object-oriented transactions. *Software, IEEE*, 8(2):84–92, March 1991.
2. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. AI Research*, 36:1–69, 2009.
3. Joachim Biskup, Ralf Menzel, Torsten Polle, and Yehoshua Sagiv. Decomposition of Relationships through Pivoting. In *Conceptual Modeling*, pages 28–41, 1996.
4. Joachim Biskup and Torsten Polle. Decomposition of Database Classes under Path Functional Dependencies and Onto Constraints. In *Foundations of Information and Knowledge Systems*, pages 31–49, 2000.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-based identification constraints in description logics. In Gerhard Brewka and Jérôme Lang, editors, *KR*, pages 231–241, 2008.
6. Diego Calvanese, Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, and Mantas Simkus. Capturing Relational Schemas and Functional Dependencies in RDFS. In *to appear in Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, 2014.
7. Vitaliy Khizder, David Toman, and Grant E. Weddell. Adding ABoxes to a Description Logic with Uniqueness Constraints via Path Agreements. In *Proc. International Workshop on Description Logics DL2007*, pages 339–346, 2007.
8. Vitaliy L. Khizder, David Toman, and Grant Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, pages 1017–1032, 2000.
9. Vitaliy L. Khizder, David Toman, and Grant Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *Int. Conf. on Database Theory ICDT'01*, pages 54–67, 2001.
10. Il-Yeol Song and Peter P. Chen. Entity relationship model. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, volume 1, pages 1003–1009. Springer, 2009.
11. Bernhard Thalheim. Extended entity relationship model. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, volume 1, pages 1083–1091. Springer, 2009.
12. David Toman and Grant Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Description Logics 2001*, pages 76–85. CEUR-WS vol.49, 2001.
13. David Toman and Grant Weddell. On Keys and Functional Dependencies as First-Class Citizens in Description Logics. In *Proc. of Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 647–661, 2006.
14. David Toman and Grant E. Weddell. On the interaction between inverse features and path-functional dependencies in description logics. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 603–608, 2005.
15. David Toman and Grant E. Weddell. On keys and functional dependencies as first-class citizens in description logics. *J. Aut. Reasoning*, 40(2-3):117–132, 2008.
16. David Toman and Grant E. Weddell. Applications and extensions of PTIME description logics with functional constraints. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 948–954, 2009.
17. David Toman and Grant E. Weddell. Conjunctive Query Answering in $\mathcal{CFD}_{nc}$: A PTIME Description Logic with Functional Constraints and Disjointness. In *Australasian Conference on Artificial Intelligence*, pages 350–361, 2013.

18. David Toman and Grant E. Weddell. Answering Queries over $\mathcal{CFD}^{\forall}_{nc}$ Knowledge Bases. Technical Report CS-2014-14, Cheriton School of Computer Science, University of Waterloo, 2014.
19. David Toman and Grant E. Weddell. On adding inverse features to the description logic $\mathcal{CFD}^{\forall}_{nc}$. In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014.*, pages 587–599, 2014.
20. David Toman and Grant E. Weddell. Pushing the $\mathcal{CFD}_{nc}$ Envelope. In *International Workshop on Description Logics DL2014*, 2014.

Query Rewriting in Horn-$\mathcal{SHIQ}$ (Extended Abstract)

Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou

School of Electrical and Computer Engineering,
National Technical University of Athens, Greece

1 Introduction

Resolution is an important and attractive tool for rewriting a Description Logic ontology into (disjunctive) datalog for the purposes of query answering [3, 5, 6]. This is because there are already many general purpose resolution calculi that can support deduction in much more expressive logics and which have well-defined powerful redundancy elimination criteria like clause subsumption. However, the generality of these procedures implies that the characteristics and structure of DL axioms are not fully exploited during saturation and as a consequence the designed rewriting algorithms often suffer from performance issues [7, 4]. More precisely, resolution algorithms like those designed in [3, 5, 6] usually produce far too many clauses that contain function terms, many of which are never used to derive other function-free clauses that are members of the ontology rewriting. For example, a typical algorithm will always resolve clauses $A(x) \leftarrow R(x, y) \wedge B(y)$ and $R(x, f(x)) \leftarrow C(x)$ to produce $A(x) \leftarrow C(x) \wedge B(f(x))$ even if the function term $f(x)$ cannot be subsequently eliminated.

In our previous work we have defined a novel resolution-based rewriting algorithm for DL-Lite and $\mathcal{ELHI}$ that largely avoids the generation of such redundant clauses [8]. The algorithm uses a macro-inference rule, called *n-shrinking*, which searches for sets of clauses such that when these are used as side-premises in consecutive resolutions, intermediate clauses with function terms can be produced but the final resolvent must be function-free. For example, in the previous scenario, the algorithm will consider resolving the two clauses to produce $A(x) \leftarrow C(x) \wedge B(f(x))$ only if a clause of the form $B(f(x)) \leftarrow C(x)$ also exists in the working set of clauses and hence the function-free clause $A(x) \leftarrow C(x)$ can finally be obtained. In order to reduce the size of the set where such pairs of side premises are looked up, in contrast to previous approaches, the algorithm does not "propagate" function symbols. For example, all algorithms in [3, 5, 6] will resolve clauses $S(x, f(x)) \leftarrow C(x)$ and $R(x, y) \leftarrow S(x, y)$ to produce $R(x, f(x)) \leftarrow C(x)$. In contrast, the algorithm in [8] will first try to *unfold* the clause $R(x, y) \leftarrow S(x, y)$ on some clause of the form $A(x) \leftarrow S(x, y) \wedge B(y)$ to produce $A(x) \leftarrow R(x, y) \wedge B(y)$ and then consider n-shrinking on that clause.

We have considerably extended our previous work and designed a datalog rewriting algorithm for Horn-$\mathcal{SHIQ}$ ontologies that follows the same principles.

Our first extension is to modify the unfolding and n-shrinking rules to be applicable on clauses with equality that are a distinctive feature of Horn-$\mathcal{SHIQ}$.

Example 1. Consider an ontology consisting of the following axioms given also in clausal form:

$$A \sqsubseteq \leq 1R.B \rightsquigarrow y \approx z \leftarrow A(x) \wedge R(x,y) \wedge R(x,z) \wedge B(y) \wedge B(z) \quad (1)$$

$$D \sqsubseteq \exists R.\top \rightsquigarrow R(x, g(x)) \leftarrow D(x) \quad (2)$$

$$C \sqsubseteq \exists S.B \rightsquigarrow S(x, f(x)) \leftarrow C(x), \quad B(f(x)) \leftarrow C(x) \quad (3)$$

$$S \sqsubseteq R \rightsquigarrow R(x,y) \leftarrow S(x,y) \quad (4)$$

Unfolding between (1) and (4) produces $y \approx z \leftarrow A(x) \wedge S(x,y) \wedge R(x,z) \wedge B(y) \wedge B(z)$ on which shrinking with premises clauses (3)(a) and (3)(b) produces $f(x) \approx z \leftarrow A(x) \wedge C(x) \wedge R(x,z) \wedge B(z)$. Notice how shrinking prevents resolving clause (1) with clause (2) which would construct a redundant resolvent. $\diamondsuit$

The biggest challenge in the new algorithm is to deal with equality reasoning. Like in [3] we employ superposition, however, following the ideas set in [8] we try to restrict its application in such a way that unnecessary intermediate clauses are constructed only when necessary. First, due to n-shrinking, only equality clauses with function-free bodies can appear in the working set (see also previous example). Hence, superposition inferences can be restricted to have only such clauses as side premises which significantly reduces the search space. However, superposition inferences can introduce new function terms in the body of a clause. Consider for example clauses $f(g(x')) \approx x' \leftarrow B(x')$ and $R(x, f(x)) \leftarrow A(x)$. The first clause can be superposed into the second with unifier $x \mapsto g(x')$ to obtain the resolvent $R(g(x'), x') \leftarrow A(g(x')) \wedge B(x')$. Now, first, note that by the *basic* strategy of superposition [1] (superposition remains complete if it is only applied into function terms not introduced by previous unification steps) no subsequent superpositions need to be applied on the body of clause $R(g(x'), x') \leftarrow A(g(x')) \wedge B(x')$. Second, according to the ideas in [8] and our previous discussion this intermediate clause is of importance only if $g(x')$ can be eliminated from the body. These two observations combined imply that we can devise the following macro-superposition inferences:

$$\frac{R(x, f(x)) \leftarrow A(x) \quad f(g(x)) \approx x \leftarrow B(x), A(g(x)) \leftarrow C(x)}{R(g(x), x) \leftarrow C(x) \wedge B(x)}$$

$$\frac{B(f(x)) \leftarrow A(x) \quad f(g(x)) \approx x \leftarrow B(x), A(g(x)) \leftarrow C(x)}{B(x) \leftarrow C(x) \wedge B(x)}$$

where $f(x)$ has not been introduced by a previous unification.

A detailed description and definition of the algorithm can be found at `http://www.image.ece.ntua.gr/~despoina/document.pdf`.

2 Evaluation

We have implemented our rewriting algorithm into our prototype system Rapid.[1] We conducted an experimental evaluation and compared it against Clipper [2], to the best of our knowledge, the only available conjunctive query rewriting system for Horn-$\mathcal{SHIQ}$ ontologies. Our test suite included Horn-$\mathcal{SHIQ}$ fragments of the ontologies NASA SWEET 2.3, Periodic, and DOLCE2.1Lite-Plus. We also used the UOBM ontology that is provided in Clipper's test suite. For the UOBM ontology we used the 10 queries that come together with Clipper, while for the rest we manually constructed 5 test queries. All tests were performed on a 2,26GHz Intel Core 2 Duo laptop running OS X 10.9.5 and JVM 1.7. We set a timeout to 2 hours. Table 1 indicates the results. As can be seen regarding UOBM (left sub-table) Rapid is consistently faster than Clipper. The sizes of the computed rewritings are roughly the same and differences are attributed to the different structures of the computed datalog rewritings. Regarding the much larger and relatively real-world ontologies (right sub-table) Clipper failed to terminate within the set time limit whereas Rapid requires at most a few seconds.

Table 1: Evaluation results

(a)

UOBM (207 axioms)			
t (ms)		Rew. size	
Rapid	Clipper	Rapid	Clipper
11	64	3	2
18	56	14	16
65	1415	109	920
25	59	22	45
18	67	16	33
20	117	13	16
21	65	14	15
14	61	10	25
28	55	31	33
23	61	23	23

(b)

$\mathcal{O}$	Axioms	t (ms)	Rew. size
NASA SWEET	11578	56	170
		142	399
		177	551
		414	979
		348	989
Periodic_full	43689	29	23
		1768	533
		1336	631
		129	195
		605	714
DOLCE2.1	1055	1314	1192
		1230	1194
		35	85
		163	205
		1379	1192

Acknowledgements Giorgos Stoilos was funded by a Marie Curie Career Reintegration Grant within EU's FP7 under grant agreement 303914.

[1] `http://www.image.ece.ntua.gr/~achort/rapid/`

References

1. Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation. *Information and computation*, 121(2):172–192, 1995.
2. Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *Proc. of AAAI 2012*, 2012.
3. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. of Autom. Reas.*, 39(3):351–384, 2007.
4. Martha Imprialou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking ontology-based query rewriting systems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 2012.
5. Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. A Comparison of Query Rewriting Techniques for DL-lite. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proc. of the 22nd Int. Workshop on Description Logics (DL 2009)*, 2009.
6. Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *Journal of Applied Logic*, 8(2):186–209, 2010.
7. Frantisek Simancik, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond horn ontologies. In *Proc. of IJCAI 2011*, pages 1093–1098, 2011.
8. Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos Stamou. Optimising resolution-based rewriting algorithms for owl ontologies. *Journal of Web Semantics*, Accepted, In press, 2015.

Decidable Verification of Knowledge-Based Programs over Description Logic Actions with Sensing ⋆

(Extended Abstract ⋆⋆)

Benjamin Zarrieß[1] and Jens Claßen[2]

[1] Theoretical Computer Science, TU Dresden, Germany,
[2] Knowledge-Based Systems Group, RWTH Aachen University, Germany

1 Introduction

Since the GOLOG [5, 10] family of action programming languages has become a popular means for control of high-level agents, the verification of temporal properties of GOLOG programs has received increasing attention [4, 7]. Both the GOLOG language itself and the underlying Situation Calculus [11, 13] are of high (first-order) expressivity, which renders the general problem undecidable. Identifying non-trivial fragments where decidability is given is therefore a worthwhile endeavour [6, 15].

In this extended abstract we consider the class of so-called *knowledge-based programs*, which are suited for more realistic scenarios where the agent possesses only incomplete information about its surroundings and has to use sensing in order to acquire additional knowledge at run-time. As opposed to classical GOLOG, knowledge-based programs contain explicit references to the agent's knowledge, thus enabling it to choose its course of action based on what it knows and does not know. Formalizations of knowledge-based programs in the epistemic Situation Calculus were proposed by Reiter [14] and later by Claßen and Lakemeyer [3].

Here we review our work on a new epistemic action formalism based on the basic Description Logic (DL) $\mathcal{ALC}$ obtained by combining and extending earlier proposals for DL action formalisms [1] and epistemic DLs [8]. From the latter we use a concept constructor for knowledge to formulate test conditions within programs and desired properties thereof, while we extend the former by not only including physical, but also sensing actions. More precisely, in our setting a knowledge-based programs for the control of a single agent consists of the following ingredients: 1. an (objective) $\mathcal{ALC}$-TBox and ABox representing the *initial static knowledge* of the agent about the world.; 2. a set of *primitive actions* describing the basic abilities of an agent to change the world and to gain new information from the environment and 3. a *program expression* defining the possible courses of action by combining primitive actions and subjective conditions formulated in the epistemic DL $\mathcal{ALCOK}$ (an extension of $\mathcal{ALC}$ with nominals ($\mathcal{O}$) and an epistemic constructor ($\mathcal{K}$)) using programming constructs

⋆ Supported by DFG Research Unit FOR 1513, (`http://www.hybrid-reasoning.org`)
⋆⋆ See [2] for the long versions of the paper and [16] for the technical report.

for sequencing, iteration and nondeterministic choice. Desired properties of such a program can be expressed in LTL over $\mathcal{ALC}$-concept inclusions and $\mathcal{ALCOK}$-ABox assertions - a logic we call $\mathcal{ALCOK}$-LTL. The *verification problem* asks whether or not all runs of a given knowledge-based program satisfy a given $\mathcal{ALCOK}$-LTL formula.

Verifying knowledge-based programs with this language yields multiple advantages. First, under reasonable restrictions we obtain decidability of verification for a formalism whose expressiveness goes far beyond propositional logic. Moreover, it enables us to resort to powerful DL reasoning systems. Finally, the new formalism also inherits many useful properties of the epistemic Situation Calculus and $\mathcal{ES}$ such as Reiter's [12] solution to the frame problem and a reasoning mechanism resembling Levesque and Lakemeyer's [9] Representation Theorem where reasoning about knowledge is reduced to reasoning in the standard DL $\mathcal{ALCO}$.

2 Example

As an example consider a mobile robot in a factory whose task it is to detect faulty gears and do the necessary repairs before turning them on. The agent is equipped with the following KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ representing its initial knowledge about the world:

$$\mathcal{T} = \{\mathit{Fault} \sqsubseteq \mathit{CritFault} \sqcup \mathit{UncritFault}, \exists \mathit{has\text{-}f}.\top \sqsubseteq \mathit{System}, \mathit{System} \sqsubseteq \forall \mathit{has\text{-}f}.\mathit{Fault}\};$$
$$\mathcal{A} = \{\mathit{System}(\mathit{gear}), \neg \mathit{On}(\mathit{gear}), \mathit{Fault}(\mathit{blocked})\}.$$

The first concept inclusion (CI) in $\mathcal{T}$ states that faults are critical faults or uncritical ones, the last two CIs define the domain *System* and range *Fault* for the role *has-f*. $\mathcal{A}$ describes a simple initial situation.

To represent conditional effects of primitive actions and axioms whose truth can be sensed we use boolean combinations of *atoms*, i.e. ABox assertions where in place of individuals also variables are allowed. An *effect* is of the form φ/γ, where φ is a boolean combination of atoms and γ is a literal of the form $A(z), \neg A(z), r(z, z')$ and $\neg r(z, z')$. A *primitive action* is a pair of the finite sets eff and sense, where eff is a set of effects and sense a finite set of boolean combinations of atoms. For example consider the following actions:

$$\texttt{turn-on}(x) : (\mathsf{eff} = \{(\neg\exists \mathit{has\text{-}f}.\mathit{CritFault}(x))/\mathit{On}(x)\}, \mathsf{sense} = \emptyset),$$
$$\texttt{sense-on}(x) : (\mathsf{eff} = \emptyset, \mathsf{sense} = \{\mathit{On}(x)\}).$$

$\texttt{turn-on}(x)$ with variable x has a single conditional effect that causes x to be *On* after the action is executed only if x previously has no critical fault. No sensing result is provided. $\texttt{sense-on}(x)$ is a pure sensing action that represents the agent's ability to perceive whether $\mathit{On}(x)$ is true in the real world.

Semantically, a primitive ground action induces a binary relation on *epistemic interpretations* $(\mathcal{I}, \mathcal{W})$ which allow us to explicitly distinguish changes affecting the real world, represented by the interpretation $\mathcal{I}$, and changes to the knowledge state $\mathcal{W}$, which is a set of interpretations (i.e., possible worlds) over a common countably infinite domain. In our semantics we also assume that the agent knows

the physical effects of its primitive actions. For instance, assume $\mathcal{K}$ as given above is all the agent knows initially about the world. Thus, it is initially known that *gear* is *not on*, but the effect condition $\neg\exists \mathit{has\text{-}f}.\mathit{CritFault}(\mathit{gear})$ of $\texttt{turn-on}(\mathit{gear})$ is unknown, i.e. there is a least one possible world satisfying $\mathcal{K}$ where *gear* is an instance of $\exists \mathit{has\text{-}f}.\mathit{CritFault}$ and one where this is not the case. Consequently, the actual outcome of executing $\texttt{turn-on}(\mathit{gear})$ in $\mathcal{K}$ is also unknown. This can be expressed by the epistemic ABox assertion $\neg\mathbf{K}\,\mathit{On} \sqcap \neg\mathbf{K}\neg\mathit{On}(\mathit{gear})$, where the $\mathbf{K}$ is used here as a concept constructor, intuitively denoting the *known* instances. If the agent now in turn executes $\texttt{sense-on}(\mathit{gear})$, it will also come to know whether *gear* has a critical fault or not, i.e. both epistemic ABox assertions $\mathbf{K}\exists \mathit{has\text{-}f}.\mathit{CritFault} \sqcup \mathbf{K}\neg\exists \mathit{has\text{-}f}.\mathit{CritFault}(\mathit{gear})$ and $\mathbf{K}\,\mathit{On} \sqcup \mathbf{K}\neg\mathit{On}(\mathit{gear})$ come to hold. A knowledge-based program describing the behaviour of an agent is then given as follows, where $\texttt{sense-f}(\mathit{gear}, x)$ is an additional sensing action for checking if *gear* has fault x and $\texttt{repair}(\mathit{gear}, x)$ an action for removing fault x of *gear*.

while $\neg\mathbf{K}(\forall \mathit{has\text{-}f}.\neg\mathbf{K}\,\mathit{Fault})(\mathit{gear})$
  $\mathsf{pick}(x) : \mathbf{K}\mathit{Fault}(x) \wedge \neg\mathbf{K}\mathit{has\text{-}f}(\mathit{gear}, x)? \wedge \neg\mathbf{K}\neg\mathit{has\text{-}f}(\mathit{gear}, x).\texttt{sense-f}(\mathit{gear}, x)$;
  if $\mathbf{K}\mathit{has\text{-}f}(\mathit{gear}, x)$ **then** $\texttt{repair}(\mathit{gear}, x)$ **else continue**;
end; $\texttt{turn-on}(\mathit{gear})$; $\texttt{sense-on}(\mathit{gear})$

As long as the agent does not know that *gear* has no known fault, a known fault x is chosen non-deterministically for which it is unknown whether *gear* has it or not. The agent then senses whether *gear* has this fault and repairs it if necessary. After completing the loop the agent turns on the gear system and checks if this was successful. An example for a property of this program to be verified is if a gear initially has an unknown critical fault, then the agent will eventually come to know it. This can be expressed by the following $\mathcal{ALCOK}$-LTL formula:

$$\exists \mathit{has\text{-}f}.(\mathit{CritFault} \sqcap \neg\mathbf{K}\mathit{Fault})(\mathit{gear}) \rightarrow \Diamond\mathbf{K}\exists \mathit{has\text{-}f}.(\mathit{CritFault} \sqcap \neg\mathbf{K}\mathit{Fault})(\mathit{gear}).$$

In our semantics of actions it is not guaranteed that the TBox given in the initial KB always holds. However persistence of a TBox $\mathcal{T}$ in a program can be verified by checking validity of the $\mathcal{ALCOK}$-LTL formula $\Box\big(\bigwedge_{\varrho\in\mathcal{T}} \varrho\big)$.

3 Results

Unfortunately, it turns out that the verification problem is undecidable for an already quite small subset of our formalism. In our setting a state of the program consists of an epistemic interpretation and a program expression representing the program that remains to be executed. Thus, we end up with an infinite state transition system. As the source of undecidability we have identified the pick-operator for non-deterministic choice of argument, which may range over the whole countably infinite domain. However, we also have the positive result that decidability of the verification problem can be retained for a syntactically restricted fragment of the formalism where pick operators are extended with epistemic guards such that the agent is only allowed to choose an argument among the *known* individuals. We have devised an algorithm with a 2ExpSpace upper bound.

References

1. Baader, F., Lutz, C., Miličić, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proc. of AAAI 2005
2. Zarrieß, B., Claßen, J.: Verification of knowledge-based programs over description logic programs. In: Proc. of IJCAI-15 (2015)
3. Claßen, J., Lakemeyer, G.: Foundations for knowledge-based programs using ES. In: Proc. of KR 2006
4. Claßen, J., Lakemeyer, G.: A logic for non-terminating Golog programs. In: Proc. of KR 2008
5. De Giacomo, G., Lespérance, Y., Levesque, H.J.: ConGolog, a concurrent programming language based on the situation calculus. AIJ 121(1–2), 109–169 (2000)
6. De Giacomo, G., Lespérance, Y., Patrizi, F.: Bounded situation calculus action theories and decidable verification. In: Proc. of KR 2012
7. De Giacomo, G., Lespérance, Y., Pearce, A.R.: Situation calculus based programs for representing and reasoning about game structures. In: Proc. of KR 2010
8. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A.: An epistemic operator for description logics. AIJ 100(1-2), 225–274 (1998)
9. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. MIT Press (2001)
10. Levesque, H.J., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming 31(1–3), 59–83 (1997)
11. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 4, pp. 463–502. (1969)
12. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy pp. 359–380 (1991)
13. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
14. Reiter, R.: On knowledge-based programming with sensing in the situation calculus. ACM Trans. Comput. Log. 2(4), 433–457 (2001)
15. Zarrieß, B., Claßen, J.: Verifying CTL* properties of Golog programs over local-effect actions. In: Proc. of ECAI 2014
16. Zarrieß, B., Claßen, J.: Verification of knowledge-based programs over description logic actions. LTCS-Report 15-10,
    See `http://lat.inf.tu-dresden.de/research/reports.html`.

Concept Forgetting in $\mathcal{ALCOI}$-Ontologies using an Ackermann Approach

Yizheng Zhao and Renate A. Schmidt

The University of Manchester, UK

Abstract. We present a method for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$. The method is an adaptation and improvement of a second-order quantifier elimination method developed for modal logics and used for computing correspondence properties for modal axioms. It follows an approach exploiting a result of Ackermann adapted to description logics. Important features inherited from the modal approach are that the inference rules are guided by an ordering compatible with the elimination order of the concept symbols. This provides more control over the inference process and reduces non-determinism, and the size of the search space. The method is extended with a new case splitting inference rule, and several simplification rules. Compared to related forgetting and uniform interpolation methods for description logics, the method can handle inverse roles, nominals and ABoxes. Compared to the modal approach on which it is based, it is more efficient in time and has higher success rates. The method has been implemented in Java using the OWL API. Preliminary experimental results show that the order in which the concept symbols are eliminated significantly affects the success rate and efficiency.

1 Introduction

Ontology-based technologies provide novel ways of building knowledge processing systems and play an important role in many different areas, both in research projects but also in industry applications. Big ontologies contain however large numbers of symbols and knowledge modelled in them is rich and inevitably heterogeneous. Thus there are situations, where it is useful to be able to restrict the ontology to a subset of the signature and *forget* those symbols which do not belong to the subset, for example, when an ontology needs to be analysed by an ontology engineer to gain an understanding of the information represented in it. Another example is a scenario where ontologies are distributed at separate remote sites and information is exchanged via agents. Since the vocabularies known to the agents at the different sites will vary, communication between the agents needs to be limited to using the common language to avoid ambiguity and confusion caused by the inconsistency of the vocabularies being used. At this point, it would be beneficial if the signature symbols in one ontology that are not known to the other agents can be eliminated without losing information required for the communication. In other words, signature symbols belonging

to only one of the ontologies are forgotten, and communication is restricted to information expressed in the shared language of the agents' ontologies. Another use of forgetting is restricting the vocabulary of an ontology to more general concept symbols, and forgetting those that are more specific, to create a summary of the ontology [29]. Situations where ontologies are published, shared, or disseminated, but some sensitive parts described in terms of particular signature symbols needs to be kept confidential or unseen to the receiver, is a potential application of forgetting [3]. This is relevant for medical and military uses, and uses in industry to ensure proprietary information can be kept hidden.

The contribution of this paper is the presentation of a method for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$. $\mathcal{ALCOI}$ extends the description logic $\mathcal{ALC}$ with nominals and inverse roles. Forgetting concept symbols for $\mathcal{ALCOI}$ is a topic where no method is available yet, but a number of related methods exist. Forgetting can be viewed as the problem dual to uniform interpolation. A lot of recent work has been focussed on uniform interpolation of mainly TBoxes represented in several description logics, ranging from ones with more limited expressivity, such as DL-Lite [31] and $\mathcal{EL}$ [20, 22] and $\mathcal{EL}$-extensions [11], to more expressive ones, such as $\mathcal{ALC}$ [21, 30, 19, 14, 13], $\mathcal{ALCH}$ [12], $\mathcal{SIF}$ [17] and $\mathcal{SHQ}$ [15].

Forgetting can however also be viewed as a second-order quantification problem, which is the view we take in this paper. In second-order quantifier elimination, the aim is to eliminate existentially quantified predicate symbols in order to translate second-order formulae into equivalent formulae in first-order logic [5, 6, 8, 4, 7, 26, 23, 24, 28]. In uniform interpolation the aim is to eliminate symbols too, though it is not required that the result is logically equivalent to the corresponding formula in second-order logic, only that all important consequences are preserved.

Our method is adapted from a method, called MSQEL, designed for modal logic to compute first-order frame correspondence properties for modal axioms and rules [26]. The adaptation exploits the close relationship between description logics and modal logics [25]. Our method contributes three novel aspects. It is the first method for forgetting concept symbols from ontologies specified in the description logic $\mathcal{ALCOI}$. It inherits from MSQEL the consideration of elimination orders, which has been shown to improve the success rate and make it succeed on a wider range for problems in the modal logic corresponding to $\mathcal{ALCOI}$ [26]. The success rate and its scope is further improved by the incorporation of a new case splitting rule and generalised simplification rules. Results of an empirical evaluation show improved success rates and performance for these techniques.

2 Definition of $\mathcal{ALCOI}$ and Other Basic Definitions

Let N_C and N_R be the set of atomic concepts and the set of atomic roles, respectively, and let N_O be the set of nominals. $\mathcal{ALCOI}$-concepts have one of these forms:

$$a \mid \bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists R.C \mid \exists R^-.C \mid \forall R.C \mid \forall R^-.C,$$

where $a \in N_O$, $A \in N_C$, $R \in N_R$, and C and D are arbitrary $\mathcal{ALCOI}$-concepts. R^- denotes the inverse of the role R. By definition, R^{--} is expressively the same as R.

An ontology usually consists of two parts, namely a TBox and an ABox. A TBox contains a set of axioms of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are concepts. A concept definition $C \equiv D$ can be expressed by two general inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. In $\mathcal{ALCOI}$, ABox axioms can be expressed as inclusions in the TBox: a concept assertion $C(a)$ can be expressed as $a \sqsubseteq C$, and a role assertion $R(a, b)$ as $a \sqsubseteq \exists R.b$. In this paper, we treat the ABox axioms as inclusions, consequently in our considerations $\mathcal{ALCOI}$-ontologies are assumed to contain TBox axioms only.

We define an interpretation $\mathcal{I}$ for $\mathcal{ALCOI}$ over the signature (N_C, N_R, N_O) as the pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set that represents the interpretation domain, and $\cdot^{\mathcal{I}}$ is the interpretation function that assigns to every nominal $a \in N_O$ a singleton set $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$; to every concept symbol $A \in N_C$ a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$; and to every role symbol $R \in N_R$ a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We specify the semantics of $\mathcal{ALCOI}$-concepts by extending the interpretation function to the following:

$$\bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$(R^-)^{\mathcal{I}} = \{(y, x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\}$$

The semantics of the TBox-axioms is defined as follows: an interpretation $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ *satisfies* $C \equiv D$ iff $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$. If $\mathcal{O}$ is a set of TBox axioms, $\mathcal{I}$ is a *model* of $\mathcal{O}$ iff it satisfies every axiom in $\mathcal{O}$, denoted by $\mathcal{I} \models \mathcal{O}$.

In the rest of the paper, we also need the following definitions. A *clause* is a disjunction of $\mathcal{ALCOI}$-concepts. Let A be a concept symbol and let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are A-*equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the valuation assigned to A. This means their domains coincide, i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, and for each symbol s in the signature except for A, $s^{\mathcal{I}} = s^{\mathcal{I}'}$. More generally, suppose $\Sigma = \{A_1, \ldots, A_m\} \subseteq N_C$, $\mathcal{I}$ and $\mathcal{I}'$ are Σ-*equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ are the same but differ possibly in the valuations assigned to the concept symbols in Σ.

3 Forgetting as Second-Order Quantifier Elimination

We are interested in forgetting concept symbols in axioms of an ontology $\mathcal{O}$ of TBox axioms. Let $\mathrm{sig}(\mathcal{O})$ denote the signature of $\mathcal{O}$.

Definition 1. *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{ALCOI}$-ontologies and let $\Sigma = \{A_1, \ldots, A_m\}$ be a set of concept symbols. $\mathcal{O}'$ is the* result of forgetting the symbols in Σ from $\mathcal{O}$,

if $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O}) \backslash \Sigma$ *and for any interpretation* $\mathcal{I}$*,*

$$\mathcal{I} \models \mathcal{O} \quad \textit{iff} \quad \mathcal{I}' \models \mathcal{O}' \quad \textit{for some interpretation } \mathcal{I}' \ \Sigma\textit{-equivalent to } \mathcal{I}.$$

The symbols in Σ *are the* symbols to be forgotten*. We refer to them as the* non-base symbols *and the symbols in* $\mathrm{sig}(\mathcal{O}) \backslash \Sigma$ *as the* base symbols*. The result of forgetting a concept symbol* A *from* $\mathcal{O}$ *is the result of forgetting* $\{A\}$ *from* $\mathcal{O}$*.*

The result of forgetting a symbol A from an ontology $\mathcal{O}$ can be represented as $\exists X\, \mathcal{O}^A_X$ in the extension of the language with existentially quantified concept variables. $\mathcal{O}^A_X$ is our notation for substituting every occurrence of A is $\mathcal{O}$ by X. In general, for the target language which extends the (source) language of the logic under consideration with existential quantification of predicate symbols, the result of forgetting always exists. The challenge of forgetting, as a computational problem, is to find an ontology $\mathcal{O}'$ (without any occurrences of X) in the source language (without second-order quantification) that is equivalent to $\exists X\, \mathcal{O}^A_X$, where $\mathcal{O}$ is expressed in the source language. Finding such an ontology $\mathcal{O}'$ that is equivalent to $\exists X\, \mathcal{O}^A_X$ is an instance of the *second-order quantifier elimination problem*. Forgetting a concept symbol A is thus the problem of eliminating the existential quantifier $\exists X$ from $\exists X\, \mathcal{O}^A_X$. In the following, we slightly informally say the aim is to eliminate the symbol A from $\mathcal{O}$. For this we apply second-order quantifier elimination techniques to the axioms of $\mathcal{O}$ in order to forget A (the non-base symbol). In particular, we are going to exploit an adaptation of a result of Ackermann [1], which is known as Ackermann's Lemma in the literature on second-order quantifier elimination [8].

Theorem 1 (Ackermann's Lemma for $\mathcal{ALCOI}$). *Let* $\mathcal{O}$ *be an* $\mathcal{ALCOI}$*-ontology, let* C *be a concept expression and suppose the concept symbol* A *does not occur in* C*. Let* $\mathcal{I}$ *be an arbitrary* $\mathcal{ALCOI}$*-interpretation. (i) If* A *occurs only positively in* $\mathcal{O}$*, then* $\mathcal{I} \models \mathcal{O}^A_C$ *iff for some interpretation* $\mathcal{I}'$ A*-equivalent to* $\mathcal{I}$*,* $\mathcal{I}' \models A \sqsubseteq C, \mathcal{O}$*. (ii) If* A *occurs only negatively in* $\mathcal{O}$ *then* $\mathcal{I} \models \mathcal{O}^A_C$ *iff for some interpretation* $\mathcal{I}'$ A*-equivalent to* $\mathcal{I}$*,* $\mathcal{I}' \models C \sqsubseteq A, \mathcal{O}$*.*

4 The Forgetting Method DSQEL

Our forgetting method is called DSQEL, which is short for Description logics Second-order Quantifier ELimination.

Figure 1 outlines the basic routine of the DSQEL method to forget concept symbols in $\mathcal{ALCOI}$-ontologies $\mathcal{O}$. Once receiving the input ontology and a set Σ of concept symbols to forget, the method proceeds as follows. In *Phase 1*, a preprocessing step is performed to transform the axioms into a set of clauses. This is done by replacing all inclusions $C \sqsubseteq D$ by $\neg C \sqcup D$, and all equivalences $C \equiv D$ by $\neg C \sqcup D$ and $\neg D \sqcup C$. Inexpensive equivalence-preserving syntactic simplification rules are also applied in this phase to simplify clauses. For example, $C \sqcup (C \sqcap D)$ is simplified to C. *Phase 2* counts the number of positive (and negative) occurrences of each concept symbol in Σ. Using these counts an

1. Transform ontology $\mathcal{O}$ to clausal representation $N := \text{clause}(\mathcal{O})$.
2. Process every concept symbol A in Σ and check the frequency of A in polarity terms to generate the ordering $\succ$.
3. Guided by $\succ$, apply the DSQEL calculus to each of the concept symbols in Σ to produce the ontology $\mathcal{O}'$ (with clauses interpreted in the obvious way as inclusions).
4. Apply the simplification rules provided to $\mathcal{O}'$ if needed and return the resultant ontology containing the symbols only in $\text{sig}(\mathcal{O}')\backslash\Sigma$.

Fig. 1. The phases in the basic DSQEL routine

ordering $\succ$ is defined on the symbols in Σ. This ordering determines the order in which the symbols in Σ are eliminated in the next phase. *Phase 3* applies the DSQEL calculus described in the next section to the non-base symbols in Σ one by one, starting with the symbol A largest in the ordering $\succ$. To forget A the inference rules of the DSQEL calculus are applied to the axioms containing A. Then the next largest non-base symbol is eliminated, and so on.

Forgetting a concept symbol may lead to a change of the polarities of the occurrences of the remaining Σ-symbols, and a new elimination order may have to be computed based on the refreshed polarity counts, before the forgetting method to continues. This means Phase 2 and Phase 3 will be alternately and repeatedly executed with recomputed elimination orders. If the largest current concept symbols to be eliminated could not be completely eliminated by DSQEL, then a different ordering not attempted before will be used. In the case that all possible orderings have been tried and every attempt to eliminate all non-base symbols using DSQEL is not successful, the method returns failure, because it was unable to solve the problem. On the other hand, when after a call of DSQEL the set returned does not contain any non-base symbols, then this is the result of forgetting Σ from $\mathcal{O}$.

Phase 4 subsequently applies further simplification rules and transforms the resulting axioms to simpler representations.

Different elimination orders of concept symbols applied may lead to different but equivalent results. The results can be viewed (when the remaining non-base symbols are existentially quantified) as equivalent representations of $\exists\Sigma\,\mathcal{O}$.

What is returned by the algorithm, if it terminates successfully, is a (possibly empty) ontology with all occurrences of the non-base symbols eliminated. I.e., what is returned is an ontology specified in terms of only the symbols in $\text{sig}(\mathcal{O})\backslash\Sigma$.

There are situations where our method does not succeed, for instance, when no forgetting result finitely expressible in $\mathcal{ALCOI}$ exists. This means the method is not complete, but since no complete method can exist for forgetting, as considered in this paper, this is to be expected. Concept forgetting is already not always computable for the description logic $\mathcal{EL}$ [10]. We also note that concept symbols cannot be eliminated by our method does not necessarily mean that

Ackermann:	$\dfrac{N, C_1 \sqcup A, \ldots, C_n \sqcup A}{(N^A_{\sim C_1 \sqcup \ldots \sqcup \sim C_n})^{\neg\neg C_1, \ldots, \neg\neg C_n}_{C_1, \ldots, C_n}}$
provided:	(i) A is a non-base symbol, (ii) A does not occur in any of the C_i, (iii) A is strictly maximal wrt. each C_i, and (iv) N is negative with respect to A.
Purify:	$\dfrac{N}{(N^A_{\neg\top})^{\neg\neg\top}_{\top}}$
provided:	(i) A is a non-base symbol in N, and (ii) N is negative with respect to A.

Fig. 2. The elimination rules

they are ineliminable. It might be the case that they are eliminable, but simply our method is unable to find a solution.

We can show DSQEL algorithm is correct and is guaranteed to terminate. This follows as an adaptation of the correctness and termination of the MSQEL procedure proved in [26], since all adaptations of MSQEL to DSQEL preserve logical equivalence and the calculus given in the next section is correct and terminates.

5 The DSQEL Forgetting Calculus

The order in which the non-base symbols are eliminated is determined by the ordering $\succ$ computed in Phase 2 of the DSQEL algorithm. We say a concept symbol A is *strictly maximal* with respect to a concept C if for any concept symbol B ($\neq A$) in C, $A \succ B$.

A concept C is *positive* (*negative*) with respect to a concept symbol A iff all occurrences of A in C are *positive* (*negative*). A set of concepts N is *positive* (*negative*) with respect to a concept symbol A iff all occurrences of A in N are *positive* (*negative*).

The Ackermann rule and the Purify rule, given in Figure 2, are the forgetting rules in the DSQEL calculus, which will lead to the elimination of a non-base concept symbol. Both of them have to meet particular requirements on the form of the concepts to which they apply. N is a set of $\mathcal{ALCOI}$-clauses, and by N^D_C, we mean the set obtained from N by substituting the expression C for all occurrences of D in N, where C and D are both $\mathcal{ALCOI}$-concepts. The inference rules in the DSQEL calculus are restricted by a set of side-conditions; for example, the side-conditions of the Ackermann rule require that A must be a non-base symbol and does not occur in $C_1, \ldots, C_n$, no non-base symbol occurring in C_i ($1 \leq i \leq n$) is larger than A under the ordering $\succ$, and every occurrence

of A in N must be negative. The Purify rule can be seen as a special case of the Ackermann rule, since it eliminates the non-base symbols that occur only negatively, that is, when there are no positive occurrences of A.

Surfacing: $$\frac{N, C \sqcup \forall R^{\sigma}.D}{N, \forall R^{\sigma,-}.C \sqcup D}$$

provided: (i) A is the largest non-base symbol in $C \sqcup \forall R^{\sigma}.D$,
(ii) A does not occur in C, and
(iii) A occurs positively in $\forall R^{\sigma}.D$.

Skolemization: $$\frac{N, \neg a \sqcup \neg\forall R^{\sigma}.C}{N, \neg a \sqcup \neg\forall R^{\sigma}.\neg b, \neg b \sqcup \sim C}$$

provided: (i) A is the largest non-base symbol in $\neg a \sqcup \neg\forall R^{\sigma}.C$,
(ii) A occurs positively in $\neg\forall R^{\sigma}.C$, and
(iii) b is a new nominal.

Clausify: $$\frac{N, C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)}{N, C \sqcup \sim D_1, ..., C \sqcup \sim D_n}$$

provided: (i) A is the largest non-base symbol in $C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)$,
(ii) A occurs positively in $D_1 \sqcup ... \sqcup D_n$.

Sign Switching: $$\frac{N}{(N^{A}_{\neg A})^{\neg\neg A}_{A}}$$

provided: (i) N is closed with respect to the other rules,
(ii) A is the largest non-base symbol in N, and
(iii) Sign switching wrt. A has not been performed before.

Fig. 3. The rewriting rules

The rules in Figure 3, are used to rewrite the formulae so they can be transformed into the form where either the Ackermann rule or the Purify rule is applicable. To apply these rules, the positive occurrences of the non-base symbol first have to be made 'individually isolated'. In addition, every positive occurrence of the non-base symbol must occur at the top level as a literal in a clause [26], that is, they must not occur under a quantifier restriction operator or any other logical operator except for disjunction. This is done by repeatedly using the surfacing rule. By R^{σ}, we mean the composition of a sequence of roles and by $R^{\sigma,-}$ we mean the composition of the sequence of inverses of the roles in R^{σ} with the order in the sequence reversed.

The Skolemization rule rewrites the existential expression in a clause of the form $\neg a \sqcup \neg\forall R^{\sigma}.C$, where a is a nominal. The implicit existential quantifier in $\neg\forall R^{\sigma}.C$ is Skolemized by introducing a new Skolem constant (nominal) b.

The clausify rule transforms a concept of the form $C \sqcup \neg(D_1 \sqcup ... \sqcup D_n)$ into a set of clauses.

The Sign switching rule is used to switch the polarity of a non-base symbol. It is applicable only when no other rules in the calculus are applicable with respect to this non-base symbol and the Sign switching rule has not been performed on this non-base symbol before.

Case Splitting: $$\frac{N, \neg a \sqcup C_1 \sqcup ... \sqcup C_n}{N, \neg a \sqcup C_1 \mid ... \mid N, \neg a \sqcup C_n}$$

provided: (i) A is the largest non-base symbol in $\neg a \sqcup C_1 \sqcup ... \sqcup C_n$,
(ii) A occurs positively in $C_1 \sqcup ... \sqcup C_n$.

Fig. 4. The case splitting rule

A novelty in the DSQEL calculus is the *case splitting* inference rule given in Figure 4. It splits a clause of the form $\neg a \sqcup C_1 \sqcup ... \sqcup C_n$ into smaller subclauses $\neg a \sqcup C_1$, $\neg a \sqcup C_2$, ..., $\neg a \sqcup C_n$. A single clause $\neg a \sqcup C$, together with N, forms a *case*. The original formula means that at least one of the disjuncts C_i ($1 \leq i \leq n$) is true for a. The benefits of the case splitting rule are twofold. On the one hand, the case splitting rule makes up for a limitation of the Skolemization rule, because it splits a disjunction with more than two disjuncts into several smaller cases, which the Skolemization rule is able to handle. On the other hand, our tests show it reduces the search space and increases the success rate.

Condensing I: $$\frac{N\big[C \sqcup \forall R^{\sigma_1}.\forall R^{\sigma_1,-}.... \forall R^{\sigma_n}.\forall R^{\sigma_n,-}.(C \sqcup D)\big]}{N\big[C \sqcup D \sqcup \forall R^{\sigma_1}\bot\big]}$$

provided: (i) C and D are arbitrary concepts, and
(ii) $\sigma^i \leq \sigma^1$ for $1 \leq i \leq n$

$$\frac{N\big[C \sqcup \forall R^{\sigma_1}.\forall R^{\sigma_1,-}.... \forall R^{\sigma_n}.\forall R^{\sigma_n,-}.(C \sqcup D)\big]}{N\big[C \sqcup D \sqcup \forall R^{\sigma_n}\bot\big]}$$

provided: (i) C and D are arbitrary concepts, and
(ii) $\sigma^i \leq \sigma^n$ for $1 \leq i \leq n$

Fig. 5. Sample simplification rule

We also introduced several simplification rules to transform more expressions so that inference rules become applicable, they keep expressions in simpler forms for efficiency, and most importantly, they lead to success of forgetting in more cases. Figure 5 displays two cases of the simplification rules, called Condensing I, with which one can handle clauses of a particular pattern where other existing methods fail.

6 Empirical Results

In order to evaluate how the DSQEL method behaves on real-life ontologies, we implemented it in Java using the OWL API and applied the implementation to a set of ontologies from the NCBO BioPortal[1], a large repository of biomedical ontologies. The experiments were run on a machine with an Intel® Core™ i7-4790 processor, and four cores running at up to 3.60 GHz and 8 GB of DDR3-1600 MHz RAM.

Since DSQEL handles expressivity as far as $\mathcal{ALCOI}$, the ontologies for our evaluation were restricted to their $\mathcal{ALCOI}$-fragments, and axioms outside of the scope of $\mathcal{ALCOI}$ were dropped from the ontologies. Consequently, we used 292 ontologies from the repository for our evaluation. We ran the experiments on each ontology 100 times and averaged the results to explore how forgetting was influenced by the number of the concept symbols in an ontology. A timeout of 1000 seconds was used.

To fit with possible needs of applications, we conducted experiments where we forget 10%, 30%, and 50% of the concept symbols in the ontologies. The DSQEL algorithm processes each non-base symbol and counts the number of their positive (and negative) occurrences. Based on these counts, an elimination order is generated by a heuristic algorithm. In order to see how the elimination order affects the performance of the calculus, we ran two sets of experiments, where we omitted the analysis of the elimination order for one set, and applied the analysis to the other set. The evaluation results with respect to forgetting 10%, 30%, 50% of the concept symbols in the ontologies, without and with the analysis of the elimination order, are shown in Table 1.

It can be seen that, the analysis of the elimination order leads to a decrease in the average duration of the runs of every experiment, which means that it takes shorter time to complete the same task than when analysis of the elimination order is not performed. It is evident that the analysis for the elimination order has brought a positive effect on the overall success rate (increase by 8.1%) and the timeout rate (decrease by 5.9%).

Evaluations of more aspects are being conducted at the moment. These evaluations are focussed on, for example, how the case splitting rule makes a difference to the behaviour of the DSQEL calculus, and how our method compares to the related methods of SCAN [7], DLS [5], DLS* [6], SQEMA [4], MSQEL [26], and LETHE [16] for computing uniform interpolants, in terms of the success rate and efficiency (duration and timeouts).

[1] http://bioportal.bioontology.org/

Input		Experiment				Output
Axioms Avg.	Symbols Avg.	Percentage	Order	Timeouts	Duration Avg.	Success Rate
1407	876	10%	✗	3.8%	4.509 sec.	90.1%
			✓	1.7%	2.404 sec.	97.6%
		30%	✗	7.5%	8.562 sec.	88.4%
			✓	2.2%	2.753 sec.	95.5%
		50%	✗	13.4%	15.068 sec.	85.3%
			✓	3.1%	3.004 sec.	94.9%
1407	876	Average	✗	8.2%	9.380 sec.	87.9%
			✓	2.3%	2.720 sec.	96.0%

Table 1. Forgetting 10%, 30%, and 50% of concept symbols in ontologies

7 Related Work

Probably the most important early work on the elimination of second-order quantifiers is that of Ackermann [1] in 1935. Only in 1992, Gabbay and Ohlbach [7] developed the first practical algorithm, called SCAN. SCAN is a resolution-based second-order quantifier elimination algorithm and can be used to forget predicate symbols from first-order logic formulae [24].

It has been shown that the SCAN algorithm is complete and terminates for modal axioms belonging to the famous Sahlqvist class [9]. In 1994, the hierarchical theorem proving method was developed by Bachmair et al. [2] and it has been shown that it can be used to solve second-order quantification problems. Around the same time, in 1995, Szałas [27] described a different algorithm for the second-order quantifier elimination problem, which exploits Ackermann's Lemma. The method was further extended to the DLS algorithm by Doherty et al. [5]. DLS uses a generalised version of Ackermann's Lemma and allows the elimination of existential second-order quantifiers from second-order formulae, and obtaining corresponding first-order equivalents. Nonnengart and Szałas [23] generalised the main result underlying the DLS algorithm to include fixpoints. Based on this work, Doherty et al. [6] proposed the DLS* algorithm, which attempts the derivation of either an equivalent first-order formula or a fixpoint formula from the original formula. DLS and DLS* are Ackermann-based second-order quantifier elimination methods. Ackermann-based second-order quantifier elimination was first applied to description logics in [28] by Szałas, where description logics were extended by a form of second-order quantification over concepts. More recently, Conradie et al. [4] introduced the SQEMA algorithm, which is also an Ackermann-based method but for modal logic formulae. It is specialised to find correspondences between modal formulae and hybrid modal logic formulae (and first-order formulae). Schmidt [26] has extended SQEMA and developed MSQEL as a refinement. A key novelty is the use of elimination orders, and the presentation of second-order quantifier elimination as an abstract calculus.

Investigation of forgetting as uniform interpolation in more expressive description logics was started in [29] and [21]. The first approach to compute uni-

form interpolations for $\mathcal{ALC}$-TBoxes was presented in [29]. It is a tableau-based approach, where a disjunctive normal form is required for the representation of the TBox-axioms and the uniform interpolants are incrementally approximated. It was shown in [21] that deciding the existence of uniform interpolants that can be finitely represented in $\mathcal{ALC}$ without fixpoints is 2-ExpTime-complete and in the worst case, the size of uniform interpolants is triple exponential with respect to the size of the original TBox. The first goal-oriented method based on clausal resolution was presented in [19] for computing uniform interpolants of $\mathcal{ALC}$-TBoxes, where experimental results show the practicality for real-life ontologies. Koopmann and Schmidt presented another resolution-based method exploiting structural transformation to compute uniform interpolants of $\mathcal{ALC}$-ontologies, which uses fixpoint operators to make uniform interpolants finitely representable [14]. The method has been further extended to handle $\mathcal{ALCH}$ [12], $\mathcal{SIF}$ [17], $\mathcal{SHQ}$ [15], and $\mathcal{ALC}$ with ABoxes [18].

8 Conclusion and Future Work

We have presented a second-order quantifier elimination method, called Dsqel, for forgetting concept symbols in ontologies specified in the description logic $\mathcal{ALCOI}$.

It is adapted from Msqel, an Ackermann-based second-order quantifier elimination method for a multi-modal tense logic with second-order quantification. The method is enhanced with new inference and simplification rules. The adaptation was motivated for the purpose of applying the second-order quantifier elimination techniques to the area of knowledge representation, where description logics provide important logical formalisms.

We have had a prototype implementation of our forgetting method, fully realising the Dsqel method. It is known that the success of a forgetting problem is highly dependent on, apart from the calculus itself, the non-base symbols Σ to be forgotten, and the elimination order which the method follows. The evaluation results of first experiments reported in this paper show promising and very good success rates for concept symbol forgetting.

Optimisation to both the calculus and the implementation is underway. One optimisation being investigated is the incorporation of more simplification rules in order to increase the efficiency and success rate further. We are also currently working on finding a heuristic algorithm using a dynamic way of ordering the non-base symbols, because the elimination of a particular concept symbol may affect the optimality of the elimination order, and thus computing a new order may be beneficial. Research shows that the order in which the inference rules are applied is significant because, as we observed, it is a main factor affecting the efficiency of the method.

Extending the method to handle ontologies going expressively further than $\mathcal{ALCOI}$ is a direction of the ongoing research. To explore how the forgetting of role symbols can be incorporated into our method is also of interest.

References

1. W. Ackermann. Untersuchungen *ü*ber das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.
2. L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3-4):193–212, 1994.
3. C. Bernardo and B. Motik. B.: Reasoning over ontologies with hidden content: The importby-query approach. *J. Artificial Intelligence Research*, 45:197–255, 2012.
4. W. Conradie, V. Goranko, and D. Vakarelov. Algorithmic correspondence and completeness in modal logic. I. the core algorithm SQEMA. *Logical Methods in Computer Science*, 2(1), 2006.
5. P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.
6. P. Doherty, W. Łukaszewicz, and A. Szałas. General domain circumscription and its effective reductions. *Fundam. Inf.*, 36(1):23–55, 1998.
7. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second–order predicate logic. In *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann, 1992.
8. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.
9. V. Goranko, U. Hustadt, R. A. Schmidt, and D. Vakarelov. SCAN is complete for all sahlqvist formulae. In *Relational and Kleene-Algebraic Methods in Computer Science*, volume 3051 of *Lecture Notes in Computer Science*, pages 149–162, 2004.
10. B. Konev, C. Lutz, D. Walther, and F. Wolter. Model-theoretic inseparability and modularity of description logic ontologies. *Artificial Intelligence*, 203(0):66–103, 2013.
11. B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic $\mathcal{EL}$. In *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK*, 2009.
12. P. Koopmann and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2013.
13. P. Koopmann and R. A. Schmidt. Implementation and evaluation of forgetting in ALC-ontologies. In *Proceedings of the 7th International Workshop on Modular Ontologies (WoMo 2013)*, volume CEUR-WS/Vol-1081 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org, 2013.
14. P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2013.
15. P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-Ontologies. In *Automated Reasoning - IJCAR 2014. Proceedings*, pages 434–448. Springer, 2014.
16. P. Koopmann and R. A. Schmidt. LETHE: A saturation-based tool for non-classical reasoning, 2015. Manuscript, submitted.
17. P. Koopmann and R. A. Schmidt. Saturation-based forgetting in the description logic $\mathcal{SIF}$, 2015. This volume.
18. P. Koopmann and R. A. Schmidt. Uniform interpolation and forgetting $\mathcal{ALC}$-ontologies with aboxes. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 175–181, 2015.

19. M. Ludwig and B. Konev. Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ tboxes. In *Proceedings of the 26th International Workshop on Description Logics (DL-2013)*, volume 1014 of *CEUR-WS*, pages 377–389. CEUR-WS.org, 2013.
20. C. Lutz, I. Seylan, and F. Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Principles of Knowledge Representation and Reasoning: KR 2012*. AAAI Press, 2012.
21. C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of IJCAI 2011*, pages 989–995. IJCAI/AAAI, 2011.
22. N. Nikitina. Forgetting in general $\mathcal{EL}$ terminologies. In *Description Logics*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
23. A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. 24:307–328, 1999.
24. H. J. Ohlbach. SCAN—elimination of predicate quantifiers. In M. A. McRobbie and J. K. Slaney, editors, *Automated Deduction: In proceedings of CADE-13*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 161–165. Springer, 1996.
25. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of IJCAI'91*, pages 466–471. Morgan Kaufmann, 1991.
26. R. A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.
27. A. Szałas. On the correspondence between modal and classical logic: an automated approach. *Journal of Logic and Computation*, 3:605–620, 1993.
28. A. Szałas. Second-order reasoning in description logics. *Journal of Applied Non-Classical Logics*, 16(3-4):517–530, 2006.
29. K. Wang, Z. Wang, R. Topor, J. Pan, and G. Antoniou. Concept and role forgetting in $\mathcal{ALC}$ ontologies. In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2009.
30. K. Wang, Z. Wang, R. Topor, J. Z. Pan, and G. Antoniou. Eliminating concepts and roles from ontologies in expressive description logics. *Computational Intelligence*, 30(2):205–232, 2014.
31. Z. Wang, K. Wang, R. Topor, and J. Pan. Forgetting for knowledge bases in DL-lite. In *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 245–257. Springer, 2008.

PAGOdA: Pay-as-you-go ABox Reasoning

Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

1 Introduction

Ontologies are increasingly used to provide structure and enhance access to large datasets. In such applications the ontology can be seen as a TBox, and the dataset as an ABox, with the key reasoning problem being conjunctive query (CQ) answering. Unfortunately, for OWL 2 this problem is known to be of high worst case complexity, even when complexity is measured with respect to the size of the data, and in realistic settings datasets may be very large.

One way to address this issue is to restrict the ontology to a fragment with better computational properties, and this is the motivation behind the OWL 2 profiles. Another approach is to optimise reasoning for arbitrary OWL 2 ontologies. This latter approach has proved very successful for TBox reasoning, with systems such as Konclude, HermiT, Pellet and Racer being widely used to reason over large-scale ontologies. Up to now, however, reasoning with large ABoxes has, in practice, largely been restricted to the OWL 2 profiles.

In this paper we describe PAGOdA: a highly optimised reasoning system that supports CQ answering with respect to an arbitrary OWL 2 ontology and an RDF dataset (roughly equivalent to a $\mathcal{SROIQ}$ TBox and ABox). It uses a novel approach to query answering that combines a datalog (or OWL 2 RL) reasoner (currently RDFox [11]) with a fully-fledged OWL 2 reasoner (currently HermiT [5]) to provide scalable performance while still guaranteeing sound and complete answers.[1] PAGOdA delegates the bulk of the computational workload to the datalog reasoner, with the extent to which the fully-fledged reasoner is needed depending on interactions between the ontology, the dataset and the query. This approach is 'pay-as-you-go' in the sense that query answering is fully delegated to the datalog reasoner whenever the input ontology is expressed in any of the OWL 2 profies; furthermore, even when using an out-of-profile ontology, queries can often be fully answered using only the datalog reasoner; and even when the fully-fledged reasoner is required, PAGOdA employs a range of optimisations, including relevant subset extraction, summarisation and dependency analysis, to reduce the number and size of the relevant reasoning problems.

This approach has proved to be very effective in practice: in our tests of more than 4,000 queries over 8 ontologies, none of which is contained within any of the OWL profiles, more than 99% of queries were fully answered without

[1] In practice we are limited by the capabilities of OWL 2 reasoners, which typically restrict the structure of the ontology and/or query in order to ensure decidability (which is open for CQ answering over unrestricted OWL 2 ontologies).

resorting to the fully-fledged reasoner. Moreover, even when the fully-fledged reasoner was used, the above mentioned optimisations were highly effective: the size of the dataset was typically reduced by an order magnitude, and often by several orders of magnitude, and it seldom required more than a single test to resolve the status of all potential answer tuples. Taken together, our experiments demonstrate that PAGOdA can provide an efficient conjunctive query answering service in real-world scenarios requiring both expressive ontologies and datasets containing hundreds of millions of facts.

The basic approach implemented in PAGOdA has been described in [13–15], and full details about the algorithms currently implemented can be found in a an accompanying technical report.[2] Here, we provide an overview of the system and summarise the results of an extensive evaluation.

2 The PAGOdA System

PAGOdA is written in Java and it is available under an academic license.[3] As well as RDFox and HermiT, PAGOdA also exploits the combined approach for $\mathcal{ELHO}^r_\bot$ implemented in KARMA.[4]

PAGOdA accepts as input arbitrary OWL 2 DL ontologies, datasets in turtle format and CQs in SPARQL. Queries can be interpreted under ground or certain answer semantics. In the former case, PAGOdA is sound and complete. In the latter case, however, PAGOdA is limited by the capabilities of HermiT, which can only check entailment of ground or DL concept queries; hence, PAGOdA can guarantee completeness only if the lower and upper bounds match, or if the query can be transformed into a DL concept query via rolling-up.[5] Otherwise, PAGOdA returns a sound (but possibly incomplete) set of answers, along with a bound on the incompleteness of the computed answer set.

The architecture of PAGOdA is depicted in Figure 1. We could, in principle, use any materialisation-based datalog reasoner that supports CQ evaluation and the incremental addition of facts, and any fully-fledged OWL 2 DL reasoner that supports fact entailment.

PAGOdA uses four instances of RDFox (one in each of the lower and upper bound and subset extractor components) and two instances of HermiT (one in each of the summary filter and dependency graph components).

The process of fully answering a query can be divided into several steps. Here, we distinguish between query independent steps and query dependent ones. As we can see in Figure 1, the 'loading ontology' and 'materialisation' steps are query independent. Therefore, both of them are counted as *pre-processing* steps. 'Computing query bounds', 'extracting subset' and 'full reasoning' are query dependent, and are called *query processing* steps.

We next describe each component, following the process flow of PAGOdA.

[2] `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/pagoda-tr.pdf`
[3] `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/`
[4] `http://www.cs.ox.ac.uk/isg/tools/KARMA/`.
[5] PAGOdA implements an extension of the well-known rollig-up technique.

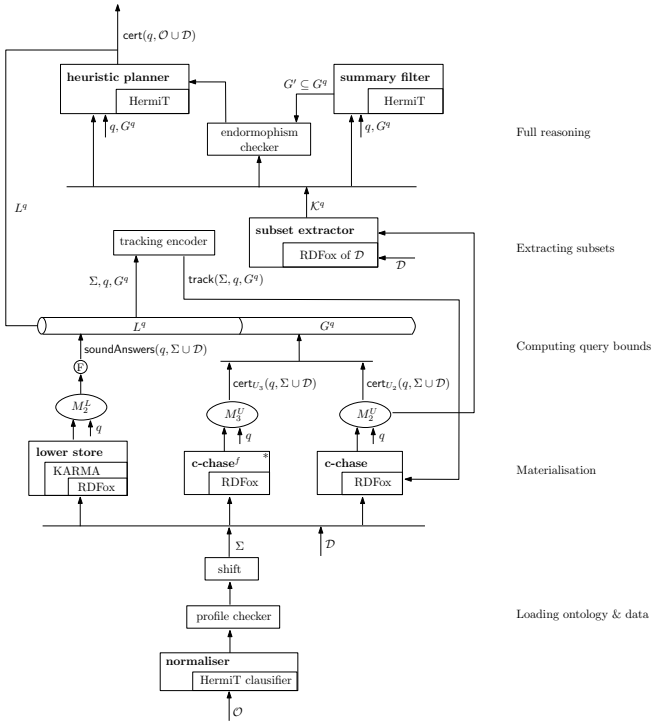


Fig. 1: The architecture of PAGOdA

Loading ontology and data. PAGOdA uses the OWL API to parse the input ontology $\mathcal{O}$. The dataset $\mathcal{D}$ is given separately in turtle format. The normaliser then transforms the ontology into a set of rules corresponding to the axioms in $\mathcal{O}$. PAGOdA's normaliser is an extension of HermiT's clausification component [5], which transforms axioms into so-called DL-clauses [12]. The dataset $\mathcal{D}$ is loaded directly into (the four instances of) RDFox.

After normalisation, the ontology is checked to determine if it is inside OWL 2 RL (resp. $\mathcal{ELHO}^r_\bot$); if so, then RDFox (resp. KARMA) is already sound and complete, and PAGOdA simply processes $\mathcal{O}$, $\mathcal{D}$ and subsequent queries using the relevant component. Otherwise, PAGOdA uses a variant of *shifting*—a polynomial program transformation commonly used in Answer Set Programming [4]—to enrich the deterministic part of the ontology with some additional information from disjunctive rules, resulting in a rule set Σ.

Materialisation. There are three components involved in this step, namely lower bound, c-chase and c-chase^f. Each of these takes as input Σ and $\mathcal{D}$, and each computes a materialisation (shown in Figure 1 as ellipses). The lower bound component first uses RDFox to compute a materialisation of $\mathcal{D}$ using the datalog subset of Σ; it then uses the materialised dataset as input to KARMA, which computes the materialisation M_2^L using the $\mathcal{ELHO}^r_\bot$ subset of Σ. The c-chase and c-chase^f components compute the M_2^U and M_3^U upper bound ma-

terialisations using chase-like procedures [1]. The former (M_2^U) is computed by over-approximating Σ into datalog; this involves, roughly speaking, transforming disjunctions into conjunctions, and replacing existentially quantified variables with fresh constants [15]. However, PAGOdA optimises the treatment of "Skolemised" existential rules by not applying them if the existential restriction is already satisfied in the data. In M_3^U, PAGOdA further optimises the treatment of disjunctions by selecting a single disjunct in the head of disjunctive rules, using a heuristic choice function that tries to select disjuncts that will not (eventually) lead to a contradiction.

If $\bot$ is derived while computing M_2^L, then the input ontology and dataset is unsatisfiable, and PAGOdA simply reports this and terminates. If $\bot$ is derived while computing M_3^U, then the computation is aborted and M_3^U is no longer used. If $\bot$ is derived while computing M_2^U, then PAGOdA checks the satisfiability of $\Sigma \cup \mathcal{D}$ (using the optimised query answering procedure described below). If $\Sigma \cup \mathcal{D}$ is unsatisfiable, then PAGOdA reports this and terminates; otherwise the input ontology and dataset is satisfiable, and PAGOdA is able to answer queries.

Computing query bounds. Given a query q, PAGOdA uses the M_2^L lower bound materialisation to compute the lower bound answer L^q, exploiting the filtration procedure in KARMA to eliminate spurious answer tuples (shown as a circle with an "F" in it in Figure 1). If $\bot$ was not derived when computing the M_3^U materialisation, then the upper bound answer U^q is the intersection of the query answers w.r.t. M_3^U and M_2^U; otherwise U^q is computed using only M_2^U.

Extracting subsets. If $L^q = U^q$, then PAGOdA simply returns L^q; otherwise it must determine the status of the tuples in the "gap" $G^q = U^q \setminus L^q$. To do this, PAGOdA extracts subsets of Σ and $\mathcal{D}$ that are sufficient to check the entailment of each such tuple. First, the tracking encoder component is used to compute a datalog program that tracks rule applications that led to the derivation of the tuples in G^q. This program is then added to the rules and data in the c-chase component, and RDFox is used to extend the c-chase materialisation accordingly. The freshly derived facts (over the tracking predicates introduced by the tracking encoder) are then passed to the subset extractor component, which identifies the relevant facts and rules in Σ and $\mathcal{D}$.

Full reasoning. PAGOdA uses HermiT to verify answers in G^q. As HermiT only accepts queries given either as facts or DL concepts, we have implemented the standard rolling-up technique to transform CQs into concepts [7]. In the summary filter component, PAGOdA uses summarisation techniques inspired by the SHER system to quickly identify spurious gap tuples [2, 3]. The remaining gap answers $G' \subseteq G^q$ are then passed to the endormorphism checker, which exploits a greedy algorithm to compute a (incomplete) dependency graph between answers in G'. An edge $\mathbf{a} \to \mathbf{b}$ in such graph between gap answers $\mathbf{a}$ and $\mathbf{b}$ encodes the following dependency: $\mathbf{b}$ is a spurious answer whenever $\mathbf{a}$ is, in which case it makes sense to check $\mathbf{a}$ using the fully-fledged reasoner before checking $\mathbf{b}$. This information is used by the heuristic planner to optimise the order in which the answers in G' are checked using HermiT. Finally, verified answers from G' are combined with the lower bound L^q.

	♯axioms	♯rules	♯∃-rules	♯∨-rules	♯facts
LUBM(n)	93	133	15	0	$n \times 10^5$
UOBM(n)	186	234	23	6	$2.6n \times 10^5$
FLY	14,447	18,013	8396	0	8×10^3
NPD	771	778	128	14	3.8×10^6
DBPedia$^+$	1,716	1,744	11	5	2.9×10^7
ChEMBL	2,593	2,960	426	73	2.9×10^8
Reactome	559	575	13	23	1.2×10^7
Uniprot	442	459	20	43	1.2×10^8

Table 1: Statistics for test datasets

3 Evaluation

We have evaluated our PAGOdA on a range of realistic and benchmark ontologies, datasets and queries. Experiments were conducted on a 32 core 2.60GHz Intel Xeon E5-2670 with 250GB of RAM, and running Fedora 20. All test ontologies, queries, and results are available online.[6]

3.1 Test Setting

Table 1 summarises our test data. Each column from left to right indicates the number of DL axioms, the number of rules after normalisation, the number of rules containing $\exists$, the number of rules containing $\vee$ in each ontology and the number of facts in each dataset.

LUBM and UOBM are widely-used reasoning benchmarks [6, 10]. To make the tests on LUBM more challenging, we extended the benchmark with 10 additional queries for which datalog lower-bound answers are not guaranteed to be complete (as is the case for the standard queries).

FLY is an ontology used in the Virtual Fly Brain tool.[7] Although the dataset is small, the ontology is rich in existentially quantified rules, which makes query answering challenging. We tested 6 CQs provided by the ontology developers.

NPD FactPages is an ontology describing petroleum activities in the Norwegian continental shelf. The ontology comes with a dataset containing 3.8 million facts. We tested all atomic queries over the signature of the ontology.

DBPedia contains information about Wikipedia entries. Although the dataset is rather large, the ontology axioms are simple and can be captured by OWL 2 RL. To provide a more challenging test, we have used LogMap [8] to extend DBPedia with a tourism ontology containing both existential and disjunctive rules. We again focused on atomic queries.

ChEMBL, Reactome, and Uniprot are ontologies that are available from the European Bioinformatics Institute (EBI) linked data platform.[8] They are

[6] `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/2015/jair/`
[7] http://www.virtualflybrain.org/site/vfb_site/overview.htm
[8] http://www.ebi.ac.uk/rdf/platform

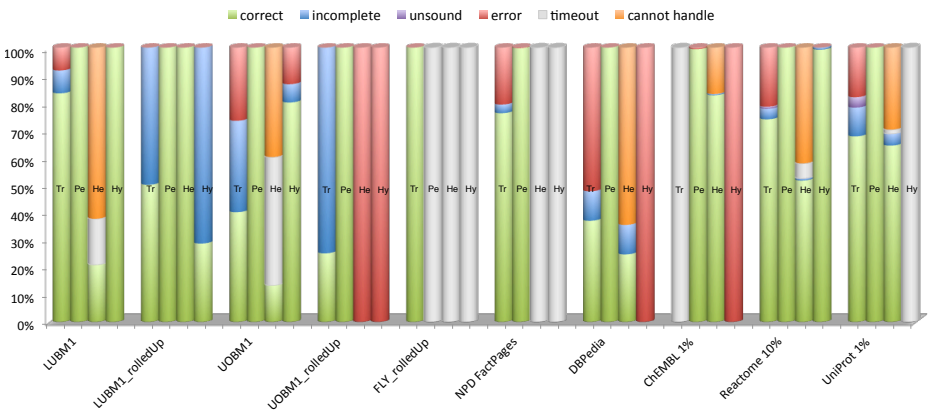


Fig. 2: Quality of the answers computed by each system. The four bars for each ontology represent Trowl, Pellet, HermiT and Hydrowl respectively.

rich in both existential and disjunctive rules, and the datasets are large. In order to test scalability, we computed subsets of the data using a data sampling algorithm based on random walks [9]. We tested example queries provided on the EBI website as well as all atomic queries over the relevant signatures.

3.2 Experiments and Results

Comparison with Other Systems We compared PAGOdA with HermiT (v.1.3.8), Pellet (v.2.3.1), TrOWL-BGP (v.1.2), and Hydrowl (v.0.2). Although TrOWL is incomplete for OWL 2, it has been included in the evaluation because, like PAGOdA, it exploits ontology approximation techniques.

In this test we used LUBM(1) and UOBM(1), 1% of the dataset for ChEMBL and UniProt, and 10% for Reactome; these are already hard for some systems, but can be processed by most. We rolled up all 6 queries into concepts wherever possible to get LUBM_rolledUp, UOBM_rolledUp and FLY_rolledUp. Since the answers to the FLY queries under SPARQL semantics are all empty, we only present results for FLY_rolledUp. We set timeouts of 20min for each individual query, and 5h for all the queries over a given ontology.

In Figure 2, each bar represents the performance of a particular reasoner w.r.t. a given ontology and set of test queries. We use green to indicate the percentage of queries for which the reasoner computed all the correct answers, where correctness was determined by majority voting, and blue (resp. purple) to indicate the percentage of queries for which the reasoner was incomplete (resp. unsound). Red, orange and grey indicate, respectively, the percentage of queries for which the reasoner reported an exception during execution, did not accept the input query, or exceeded the timeout. PAGOdA is not represented in the figure as it was able to correctly compute all answers for every query and test ontology within the given timeouts.

Figure 3 summarises the performance of each system relative to PAGOdA, but in this case we considered only those queries for which the relevant system

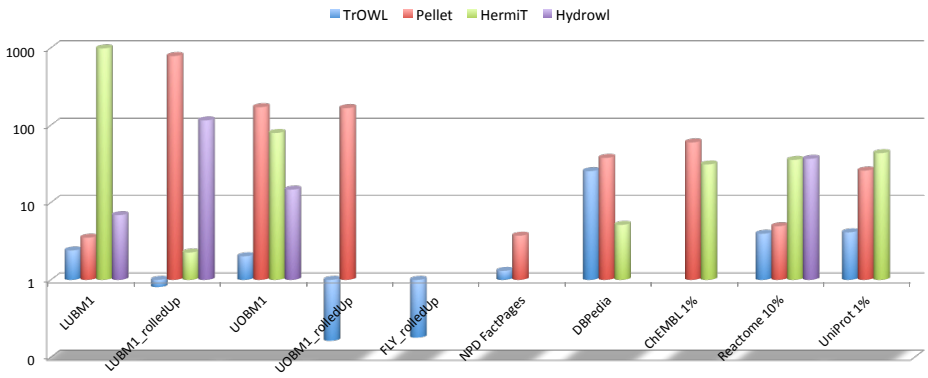


Fig. 3: Performance comparison with other systems.

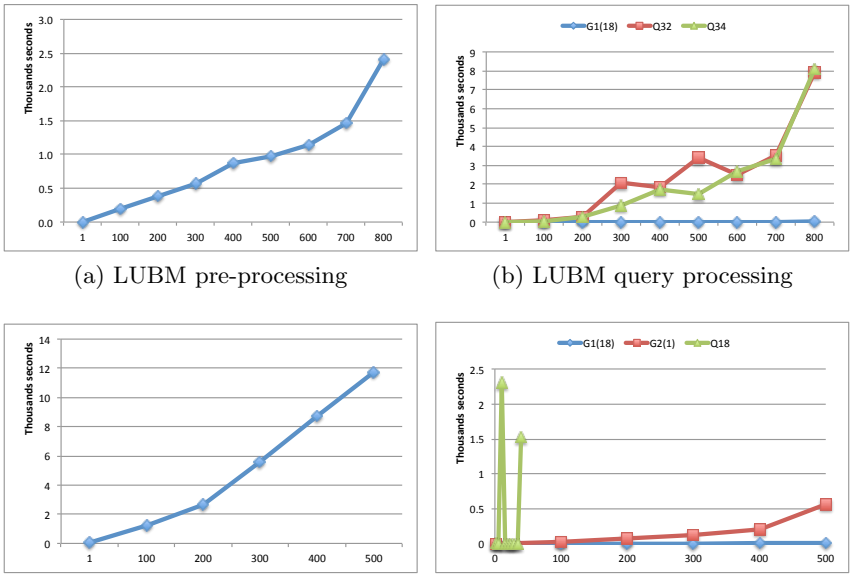
yields an answer (even if unsound and/or incomplete). This is not ideal, but we chose to consider all such queries (rather than only the queries for which the relevant system yields the correct answer) because *(i)* the resulting time measurement is obviously closer to the time that would be required to correctly answer all queries; and *(ii)* correctness is only relative as we do not have a gold standard. For each ontology and reasoner, the corresponding bar shows t_2/t_1 (on a logarithmic scale), where t_1 (resp. t_2) is the total time required by PAGOdA (resp. the compared system) to compute the answers to the queries under consideration; a missing bar indicates that the comparison system failed to answer any queries within the given timeout. Please note that two different bars for the same ontology are not comparable as they may refer to different sets of queries, so each bar needs to be considered in isolation.

TrOWL is faster than PAGOdA on LUBM_rolledUp, UOBM_rolledUp and FLY_rolledUp, but it is incomplete for 7 out of 14 LUBM queries and 3 out of 4 UOBM queries. For ChEMBL, TrOWL exceeds the timeout while performing the satisfiability check. For the remaining ontologies, PAGOdA is more efficient in spite of the fact that TrOWL is incomplete for some queries, and even unsound for several UniProt queries.

Pellet times out for the FLY ontology, but it succeeds in computing all answers in the remaining cases. We can observe, however, that in all cases Pellet is significantly slower than PAGOdA, sometimes by more than two orders of magnitude.

HermiT can only answer queries with one distinguished variable, so we could not evaluate binary queries. HermiT exceeds the timeout in many cases, and in the tests where it succeeds, it is significantly slower than PAGOdA.

Hydrowl is based on a theoretically sound and complete algorithm, but it was found to be incomplete in some of our tests. It also exceeded the timeout for three of the ontologies, ran out of memory for another two of the ontologies, and reported an exception for ChEMBL 1%. In the remaining cases, it was significantly slower than PAGOdA.



(a) LUBM pre-processing

(b) LUBM query processing

(c) UOBM pre-processing

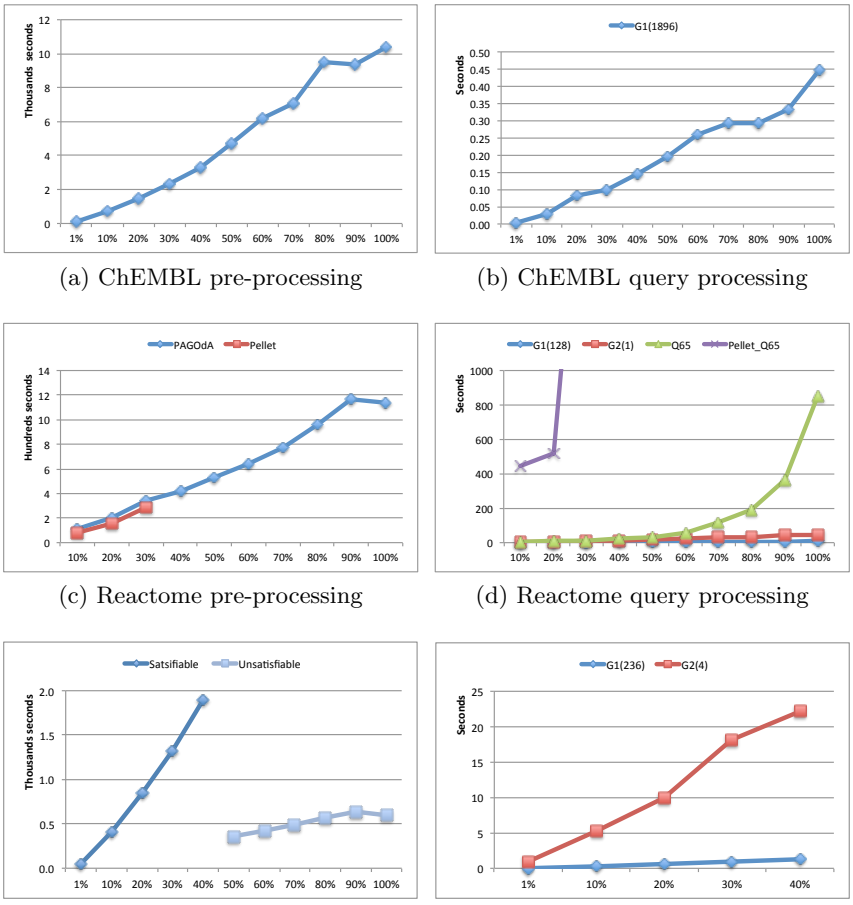
(d) UOBM query processing

Fig. 4: Scalability tests on benchmarks

Scalability Tests We tested the scalability of PAGOdA on LUBM, UOBM and the ontologies from the EBI platform. For LUBM we used datasets of increasing size with a step of $n = 100$. For UOBM we also used increasingly large datasets with step $n = 100$ and we also considered a smaller step of $n = 5$ for hard queries. Finally, in the case of EBI's datasets, we computed subsets of the data of increasing sizes from 1% of the original dataset up to 100% in steps of 10%. In each case we used the test queries described in Section 3.1. For each test ontology we measured *pre-processing time* and *query processing time* as described in Section 2. We organise the test queries into three groups: **G1**: queries for which the lower and upper bounds coincide; **G2**: queries with a non-empty gap, but for which summarisation is able to filter out all remaining candidate answers; and **G3**: queries where the fully-fledged reasoner is called over an ontology subset on at least one of the test datasets. We set a timeout of 2.5h for each individual query and 5h for all queries.

We also tested Pellet (the only other system found to be sound and complete for our tests) on Reactome, the only case were Pellet managed to process at least two datasets.

Our results are summarised in Figures 4 and 5. For each ontology, we plot time against the size of the input dataset, and for query processing we distinguish different groups of queries as discussed above. PAGOdA behaves relatively uniformly for queries in **G1** and **G2**, so we plot only the average time per query for these groups. In contrast, PAGOdA's behaviour for queries in **G3** is quite variable, so we plot the time for each individual query.



(a) ChEMBL pre-processing (b) ChEMBL query processing

(c) Reactome pre-processing (d) Reactome query processing

(e) Uniprot pre-processing (f) Uniprot query processing

Fig. 5: Scalability tests on EBI linked data platform

LUBM(n) All LUBM queries belongs to either **G1** or **G3** with the latter group containing two queries. The average query processing time for queries in **G1** never exceeds 13s; for the two queries in **G3** (Q32 and Q34), this reaches 8,000s for LUBM(800), most of which is accounted for by HermiT.

UOBM(n) As with LUBM, most test queries were contained in **G1**, and their processing times never exceeded 8 seconds. We found one query in **G2**, and PAGOdA took 569s to answer this query for UOBM(500). UOBM's randomised data generation led to the highly variable behaviour of Q18: it was in **G3** for UOBM(1) UOBM(10) and UOBM(50), causing PAGOdA to time out in the last case; it was in **G2** for UOBM(40); and it was in **G1** in all other cases.

ChEMBL All test queries were contained in **G1**, and average processing time was less than 0.5s in all cases.

	LUBM	UOBM	FLY	NPD	DBPedia	ChEMBL	Reactome	UniProt
Total	35	20	6	478	1247	1896	130	240
$L_1 + U_1$	26	4	0	442	1240	1883	82	204
$L_2 + U_1$	33	4	5	442	1241	1883	82	204
$L_2 + U_2$	33	12	5	442	1241	1883	98	204
$L_2 + U_{2\|3}$	33	16	5	473	1246	1896	128	236

Table 2: ♯Queries answered by different bounds

Reactome Groups **G2** and **G3** each contained one query, with all the remaining queries belonging to **G1**. Query processing time for queries in **G1** never exceeded 10 seconds; for **G2** processing time appeared to grow linearly in the size of datasets, and average time never exceeded 10 seconds; the **G3** query (Q65) is much more challenging, but it could still be answered in less than 900 seconds, even for the largest dataset.

On Reactome, Pellet is able to process the samples of size 10%, 20% and 30%, with pre-processing times comparable to PAGOdA. Average query-processing times for queries in **G1** and **G2** are slightly higher than those of PAGOdA, but times for query Q65 were significantly higher. This was due to PAGOdA's subset extraction technique, which is able to keep the input to the fully-fledged reasoner small, even for the largest datasets.

Uniprot In contrast to the other cases, Uniprot as a whole is unsatisfiable; however, our sampling technique can produce a satisfiable subset up to 40%. For larger subsets, pre-processing times drop abruptly as unsatisfiability can be efficiently detected in the lower bound. Query processing times were only considered for satisfiable samples. There were no queries in **G3**, and only four in **G2**, all of which were efficiently handled.

Effectiveness of Different Techniques We have evaluated the effectiveness of the various reasoning techniques implemented in PAGOdA by comparing the numbers of test queries that can be fully answered using the relevant technique.

Query bounds Table 2 illustrates the effectiveness of different combinations of upper and lower bounds in terms of the number of queries for which the bounds coincided for each test ontology and its smallest test datasets. In the table, we refer to the lower bound computed w.r.t. the datalog subset of the input knowledge base as L_1 and to the combined lower bound computed by PAGOdA as L_2. Similarly, we refer the naive upper bound computed using a datalog over-approximation of Σ as U_1; the upper bound computed w.r.t. M_2^U and M_3^U as U_2 and U_3; and the combined upper bound as $U_{2|3}$.

It can be seen that L_1 and U_1 suffice to answer most of the queries in many test ontologies. L_2 was very effective in the case of FLY, where the basic bounds did not match for any query, and also useful for LUBM, yielding matching bounds for 7 more queries. U_2, was especially effective for UOBM and Reactome, where many existentially quantified rules were already satisfied by the lower bound materialisation. Finally, the refined treatment of disjunctive rules in $U_{2|3}$ was instrumental in obtaining additional matching bounds for non-Horn ontologies.

	LUBM	UOBM	Fly	NPD	DBPedia	Reactome	Uniprot
Facts	0.5%	10.4%	7.3%	16.5%	9×10^{-5}%	5.2%	4×10^{-4}%
Rules	3.7%	10.9%	0.9%	18.4%	2.4%	5.3%	1.1%

Table 3: Size of the largest subsets given as percentage over input rules and facts.

	LUBM		UOBM			FLY	DBPedia	NPD	Reactome	UniProt	
$L_2 + U_{2\|3}$	26	14	264	112	1470	264	344	10	326	18	52	168
+ Sum	26	14	264	0	1444	264	344	0	0	0	52	0
+ Dep	1	1	1	0	1	1	7	0	0	0	37	0

Table 4: The number of hard calls to HermiT to fully answer each query

Subset extraction Table 3 shows, for each dataset, the maximum percentage of facts and rules that are included in the relevant subset over all test queries with non-matching bounds. We can observe that subset extraction is effective in all cases in terms of both facts and rules.

Summarisation and Dependencies The effectiveness of these techniques was measured by the number of 'hard' calls to HermiT that were required to fully answer each query, where a call is considered hard if the knowledge base passed to HermiT is not a summary. The first row of Table 4 shows the number of gap answers for each query where the L_2 and $U_{2|3}$ bounds don't match. Without optimisation, we would have to call HermiT this number of times to fully answer each query. Row 2 (resp. row 3) shows the number of hard calls to HermiT after applying summarisation (resp. summarisation plus dependency analysis).

4 Discussion

The reasoning techniques we have proposed here are very general and are applicable to a wide range of knowledge representation languages. Our main goal in practice, however, has been to realise our approach in a highly scalable and robust query answering system for OWL 2 DL ontologies, which we have called PAGOdA. Our extensive evaluation has not only confirmed the feasibility of our approach in practice, but also that our system PAGOdA significantly ourperforms state-of-the art reasoning systems in terms of both robustness and scalability. In particular, our experiments using the ontologies in the EBI linked data platform have shown that PAGOdA is capable of fully answering queries over highly complex and expressive ontologies and realistic datasets containing hundreds of millions of facts.

Acknowledgements. This work has been supported by the Royal Society under a Royal Society Research Fellowship, by the EPSRC projects MaSI^3, Score! and DBOnto, and by the EU FP7 project Optique.

References

1. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. Journal of Artificial Intelligence Research 48, 115–174 (2013), http://dx.doi.org/10.1613/jair.3873
2. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI 2007, Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada. pp. 299–304. AAAI Press (2007), http://www.aaai.org/Library/AAAI/2007/aaai07-046.php
3. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Scalable highly expressive reasoner (SHER). Journal of Web Semantics 7(4), 357–361 (2009)
4. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: LPNMR 2004, Proceedings of Logic Programming and Nonmonotonic Reasoning - 7th International Conference, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings. pp. 87–99 (2004)
5. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
7. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In: Kautz, H.A., Porter, B.W. (eds.) AAAI/IAAI 2000, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. pp. 399–404. AAAI Press / The MIT Press (2000), http://www.aaai.org/Library/AAAI/2000/aaai00-061.php
8. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and scalable ontology matching. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (eds.) ISWC 2011, The Semantic Web - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I. Lecture Notes in Computer Science, vol. 7031, pp. 273–288. Springer (2011), http://dx.doi.org/10.1007/978-3-642-25073-6_18
9. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD 2006, Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006. pp. 631–636 (2006)
10. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006, The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, Budva, Montenegro, June 11-14, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4011, pp. 125–139. Springer (2006), http://dx.doi.org/10.1007/11762256_12
11. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In: Brodley, C.E., Stone, P. (eds.) AAAI 2014, Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. pp. 129–137. AAAI Press (2014), http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8505
12. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (2009), http://dx.doi.org/10.1613/jair.2811

13. Zhou, Y., Nenov, Y., Cuenca Grau, B., Horrocks, I.: Complete query answering over horn ontologies using a triple store. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8218, pp. 720–736. Springer (2013)
14. Zhou, Y., Nenov, Y., Cuenca Grau, B., Horrocks, I.: Pay-as-you-go ontology query answering using a datalog reasoner. In: Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014. CEUR Workshop Proceedings, vol. 1193, pp. 352–364. CEUR-WS.org (2014)
15. Zhou, Y., Nenov, Y., Cuenca Grau, B., Horrocks, I.: Pay-as-you-go OWL query answering using a triple store. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (2014), http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8232

An Ontology-Based Archive for Historical Research

Giovanni Adorni[1], Marco Maratea[1], Laura Pandolfo[1], and Luca Pulina[2]

[1] DIBRIS, Università di Genova, Via Opera Pia, 13 – 16145 Genova – Italy
giovanni.adorni@unige.it, marco.maratea@unige.it,
laura.pandolfo@edu.unige.it
[2] POLCOMING, Università di Sassari, Viale Mancini n. 5 – 07100 Sassari – Italy
lpulina@uniss.it

1 Context and Motivation

The digitalization of cultural materials is doubtless a key-enabler for increasing accessibility of cultural heritage documents, e.g., historical texts. In the last decade Semantic Digital Libraries (see, e.g., [1]) have attracted the attention of research communities coming from different research areas, such as Cultural Heritage, History, and Knowledge Engineering. In order to find more innovative methods to improve search and retrieval operations, recently portals and digital libraries concerning Cultural Heritage have been enhanced with Semantic Web [2] technologies. Such technologies can offer effective solutions about design and implementation of user-friendly ways to access and query content and metadata [1]. In this context, a prominent example is the Europeana project [3][3].

Particularly, cultural heritage documents are characterized by being syntactically and semantically heterogeneous, multilingual, semantically rich, and highly interlinked. They are produced in a distributed, open fashion by organizations like museums, libraries, and archives, using their own established standards and best practices [4]. Historical documents represent an important component of the cultural heritage field, and they have been digitized and published on the web by means of several applications.

In this extended abstract we present STOLE[4], an ontology-based digital archive collecting some of the most relevant journal articles published between 1848 and 1946 concerning the legislative history of public administration in Italy. In STOLE we leverage ontologies for describing domain knowledge and providing semantic information integration among data in order to support historians' research. The documents stored in STOLE are regarded as a valuable source of information for historical research since through the study of these texts it is possible to trace the course of Italian history and often to find out some unexplored but useful aspects about a particular event or person. Currently, these historical sources

[3] http://europeana.eu

[4] STOLE is the acronym of the Italian "*STOria LEgislativa della pubblica amministrazione italiana*", that means "legislative history of the Italian public administration".

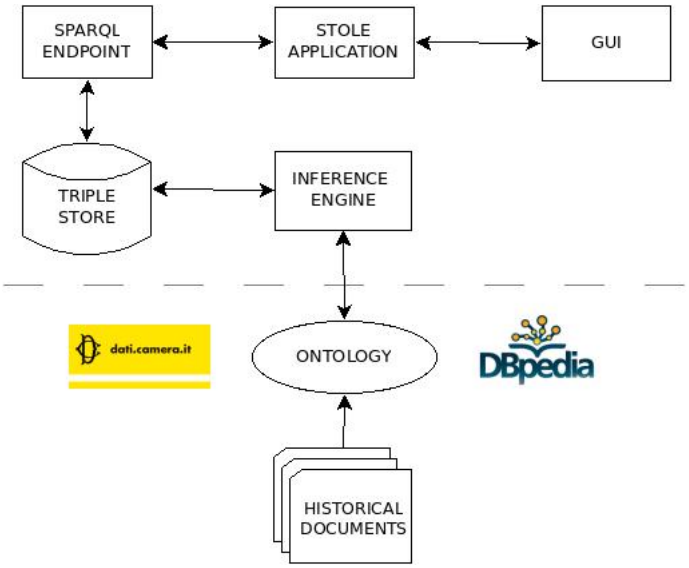


Fig. 1. The architecture of STOLE.

can be accessed going to the State Archives of Rome or to libraries that have a copy, but in both cases historians are not supported by information system during their research. Our ontology-based archive intends to handle this historical information providing search features and capabilities such that the user can search over this document collection. In the next section, we give details of the components of the system, while the abstract is concluded in Section 3 by listing the future work planned.

2 System Architecture

In Figure 1 we describe the architecture of STOLE; looking at the figure, we can see that it is composed of the modules listed in the following:

`Ontology` It represents the conceptual layer of our application. It is used to represent knowledge concerning journals, authors, cited people and events, as well as the relations between them. The DL expressivity if the STOLE[5] ontology is $\mathcal{ALCOIQ}(\mathcal{D})$, and it is composed of 2909 axioms. Actually, it is comprised of 342 individuals, its population is growing continuously. It has been developed building on existing standards and meta-data vocabularies, such as Dublin Core (`http://dublincore.org`), FOAF (`http://www.foaf-project.org`), Bio Vocabulary (`http://vocab.org/bio/0.1`) and the Bibliographic Ontology (`http://bibliontology.com`). Concerning the modeling ontology language, our choice falls to OWL2 DL [5]. We also

[5] The full documentation is available at `http://visionlab.uniss.it/STOLE_DOC`

considered some lightweight profiles, but we cannot avoid to introduce both cardinality restrictions and other role constraints, e.g., inverse object properties, in order to have proper expressivity for our application. The STOLE ontology has been populated leveraging on a set of annotated historical documents. Such semantic annotations were provided by a team of domain experts. Finally, we also developed a data integration layer in order to exploit information coming from external sources, such as DBpedia [6] and the Ontology of the Chamber of Deputies[6].

`Inference Engine` This module interacts with `Ontology` to check its consistency and to infer new knowledge to present to the user. Actually, we are using the HermiT reasoner [7].

`Triple Store` **and** `SPARQL Endpoint` They are the modules devoted to store and query the knowledge base, respectively. For these purposes, we are currently using Open Virtuoso[7].

`STOLE Application` In this module we implemented all functionalities related to query the data to `SPARQL Endpoint` and to process the answer in order to be presented to the user by means of the `GUI`.

`GUI` It is devoted to the user-system interaction and it aims at providing data representations widely used in the historical research field, such as timelines of historical events.

3 Conclusion and Future Work

We have presented an ontology-based archive and its application in the historical research domain, which includes numerous key aspects of Semantic Web. The current implementation of this framework can be extended in several ways. First of all, we are designing a Graphical User Interface to support the ontology population stage, in order to make this process of knowledge acquisition more interactive and dynamic. Finally, we are planning to introduce in STOLE a semantic indexing mechanism using Apache Solr[8], in order to extend keyword-based search functionalities.

[6] `http://dati.camera.it/data/en`
[7] `http://www.openlinksw.com/`
[8] `http://lucene.apache.org/solr`

References

1. Kruk, S.R., Westerki, A., Kruk, E.: Architecture of semantic digital libraries. In: Semantic Digital Libraries. Springer (2009) 77–85
2. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific american **284**(5) (2001) 28–37
3. Haslhofer, B., Isaac, A.: data. europeana. eu: The europeana linked open data pilot. In: International Conference on Dublin Core and Metadata Applications. (2011) 94–104
4. Ahonen, E., Hyvonen, E.: Publishing historical texts on the semantic web-a case study. In: Semantic Computing, 2009. ICSC'09. IEEE International Conference on, IEEE (2009) 167–173
5. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: Owl 2: The next step for owl. Web Semantics: Science, Services and Agents on the World Wide Web **6**(4) (2008) 309–322
6. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. Web Semantics: science, services and agents on the world wide web **7**(3) (2009) 154–165
7. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient owl reasoner. In: OWLED. Volume 432. (2008)
8. Huynh, D.F., Karger, D.R., Miller, R.C.: Exhibit: lightweight structured data publishing. In: Proceedings of the 16th international conference on World Wide Web, ACM (2007) 737–746

Reasoning in description logics with variables: preliminary results regarding the $\mathcal{EL}$ logic

Lakhdar Akroun, Lhouari Nourine, and Farouk Toumani

[1] INRIA, Grenoble, France
lakhdar.akroun@inria.fr
[2] LIMOS, CNRS, Blaise Pascal University, Clermont-Ferrand, France
nourine, ftoumani@isima.fr

Abstract. This paper studies the extension of description logics with variables ranging over infinite domains of concept names and role names. As a preliminary work, we consider more specifically the extension of the logic $\mathcal{EL}$ with variables and we investigate in this context two reasoning mechanisms, namely compliance (a kind of matching) and pattern containment. The main technical results are derived by establishing a correspondance between the $\mathcal{EL}$ logic and finite variable automata.

1 Introduction

We consider description logics augmented with variables ranged over concept names and role names. As an example, consider the following description:

$$B \equiv Person \sqcap \exists works\text{-}for.X \sqcap \exists graduated\text{-}from.X \qquad (1)$$

where the variable X takes its values from an infinite set of possible atomic concept names. B specifies the set of persons that work for the same type of organization they were graduated from. Specifications of type (1) are called hereafter a *pattern definition* and B is called a *pattern name* (or simply a pattern). An *instanciation* of a pattern is given by variable valuations. For example, if the variable X is assigned as value the atomic concept name *University* (respectively, *eSchool*), we obtain the following description B_1 (respectively, B_2) which is *compliant* with the pattern B:

$$B_1 \equiv Person \sqcap \exists works\text{-}for.University \sqcap \exists graduated\text{-}from.University$$

$$B_2 \equiv Person \sqcap \exists works\text{-}for.eSchool \sqcap \exists graduated\text{-}from.eSchool$$

Variables can also be used in role places as illustrated by the following pattern definition:

$$B_3 \equiv Person \sqcap \exists Y.Z \sqcap \exists .graduated\text{-}from.Z$$

The concept B_3 specifies the set of persons that have a relation (i.e., any kind of role) with the same type of organization they are graduated from. Indeed, the

concept B_1 is compliant with B_3 and it is also the case of the concept B_4 defined as follows:

$$B_4 \equiv Person \sqcap \exists evaluates.eSchool \sqcap \exists graduated\text{-}from.eSchool$$

Our framework supports terminological cycles as illustrated below with the pattern B_5 which specifies the persons that work for the same type of organization they are graduated from and have a relative who also work for the same type of organization she is graduated from.

$$B_5 \equiv Person \sqcap \exists works\text{-}for.X' \sqcap \exists graduated\text{-}from.X' \sqcap \exists has\text{-}relative.B_5$$

For a description logic $\mathcal{L}$, we denote by $\mathcal{L}_\mathcal{V}$ the obtained logic augmented with concept variables and role variables. We study the following reasoning mechanisms in this framework (formal definitions are given later in the paper):

- Compliance which asks whether a description E is *compliant* with a pattern C.
- Pattern containment which, given two pattern definitions C_1 and C_2, asks whether every description E compliant with C_1 is also compliant with C_2.

Indeed, the notion of a concept pattern (i.e., a concept description containing variables) is not new and has already been used, in particular, in the context of two non-standard reasonings, namely matching [1] and unification [2,4]. Given a concept pattern D and a concept description C, the matching problem asks whether there is a substitution σ of the variables by concept descriptions such that $C \sqsubseteq \sigma(D)$. Unification is a generalization of matching. It takes as input concept patterns and asks whether there exist a substitution of the variables by concept descriptions that makes the concept patterns equivalent. Our definition of concept patterns deviate from the one used in the literature with respect to the following features: (i) our definition of concept patterns is more liberal in the sense that we allow concept variables as well as role variables while usually only concept variables are allowed in concept patterns, (ii) we support cyclic pattern definitions and, inspired from the guarded variable automata theory, we consider two different types of semantics of variables (i.e., refreshing and not refreshing semantics), and (iii) our interpretation of variables is however more restrictive in this paper since we consider only *atomic variables* (i.e., we assume that variables take their values from an infinite set of atomic concept names and role names). An extension of this framework to variables that stand for descriptions would be an interesting research direction. From the reasoning perspective, our notion of compliance coincides with matching however, up to our knowledge, the notion of pattern containment has never been investigated in the literature.

We expect the proposed framework to be useful in various applications. For example, concept patterns have already been proven to be beneficial in applications where a given user is interested *'to search the knowledge base for concepts having a certain not completely specified form'* [6]. Targeting a similar purpose, we envision our framework to be useful as query language to: (i) formulate queries

over terminologies. For example, the concept B_3 given above can be viewed as a query over a given terminology while the concepts B_1 and B_2 are examples of *answers* to such a query, or (ii) to formulate ontological queries (i.e., queries over an ontology) in the context of an Ontology Based Data Access (OBDA) approach [7]. The reasoning mechanisms studied in this paper can prove particularly relevant to handle query evaluation and optimisation in such contexts.

Organization of the paper. Section 2 presents some preliminary notions regarding state machines and guarded variable automata. Section 3 recalls some basic notions of the $\mathcal{EL}$ description logic and describes the extension of this logic with variables, the obtained logic is called $\mathcal{EL}_{\mathcal{V}}$. Section 4 studies the compliance and the containment problems in the context of the $\mathcal{EL}_{\mathcal{V}}$ logic. We conclude and draw future research directions in section 5. Proofs are omitted and are included in the extended version of this paper [13].

2 Preliminaries

We first recall the notion of state machines [9] and simulation preorder between state machines. A State Machine M is a tuple $< \Sigma_M, Q_M, F_M, Q^0_M, \delta_M >$, where: Σ_M is a finite alphabet, Q_M is a set of states with $Q^0_M \subseteq Q_M$ the set of initial states and $F_M \subseteq Q_M$ the set of final states, $\delta_M \subseteq Q_M \times \Sigma_M \times Q_M$ is a set of labeled transitions. If Q_M is finite then M is called a finite state machine. If $q \in Q^0_M$ is an initial state of a machine M, we say that M is *rooted at* q.

Let $M =< \Sigma_M, Q_M, F_M, Q^0_M, \delta_M >$ and $M' =< \Sigma_{M'}, Q_{M'}, F_{M'}, Q^0_{M'}, \delta_{M'} >$ be two (eventually, infinite) state machines. A state $q_1 \in Q_M$ is simulated by a state $q'_1 \in Q_{M'}$, noted $q_1 \preceq q'_1$, iff the following two conditions hold : (i) $\forall a \in \Sigma_M$ and $\forall q_2 \in Q_M$ such that $(q_1, a, q_2) \in \delta_M$, there exists $(q'_1, a, q'_2) \in \delta_{M'}$ such that $q_2 \preceq q'_2$, and (ii) if $q_1 \in F_M$, then $q'_1 \in F_{M'}$. M is simulated by M', noted $M \preceq M'$, iff $\forall q_M \in Q^0_M, \exists q'_{M'} \in Q^0_{M'}$ s.t. $q_M \preceq q_{M'}$.

We briefly introduce now the notion of (Guarded) Variable Automata [5]. Let $\mathcal{X}$ be a finite set of variables and Σ an infinite alphabet of letters. The set $\mathcal{G}$ of guards is inductively defined as follows: $G := true \mid \alpha = \beta \mid \alpha \neq \beta \mid G \wedge G$ where $\alpha, \beta \in \Sigma \cup \mathcal{X}$. A GVA is a tuple $A = (\Sigma, \mathcal{X}, Q, Q_0, \delta, F, \mu)$, where Σ is an infinite set of letters, $\mathcal{X}$ is a finite set of variables, Q is a finite set states, $Q_0 \subseteq Q$ is a finite set of initial states, $\delta : Q \times (\Sigma_A \cup \mathcal{X}) \times \mathcal{G} \to 2^Q$ is a transition function where $\Sigma_A \subset \Sigma$ is a finite set of alphabet, $F \subseteq Q$ is a finite set of final states and $\mu : \mathcal{X} \to 2^Q$ is called the refreshing function.

In the sequel, we write (q, t, g, q') to denote a transition from q to q', when $q' \in \delta((q, t, g))$. At every point in time, a run of a GVA A is determined by its instantaneous description (or simply, configuration). A configuration of a GVA A is given by a pair $id = (q, \sigma)$ where $q \in Q$ is a state in A and $\sigma : \mathcal{X} \to \Sigma$ is a variable valuation. A valuation σ is extended to be the identity over the elements of Σ. The satisfaction of a guard g by a valuation σ, denoted $\sigma_i \models g$, is defined as usual. A run of A starts at an initial configuration $id_0 = (q_0, \sigma_0)$, with $q_0 \in Q_0$

an initial control state of A and σ_0 an arbitrary valuation of the variables of $\mathcal{X}$[3]. Then A moves from a configuration $id_i = (q_i, \sigma_i)$ to a configuration $id_j = (q_j, \sigma_j)$ over the letter $\sigma_i(t)$ if there is a transition $(q_i, t, g, q_j) \in \delta$ s.t. $\sigma_i \models g$ and $\sigma_i(x) = \sigma_j(x), \forall x \in \mathcal{X} \setminus \mu(q_j)$ (i.e., σ_i and σ_j coincide on the values of the variables that are not refreshed at the state q_j). Hence, given a GVA A, the runs of a A is captured by an infinite state machine called the extended machine of A and denoted $E(A)$. Roughly speaking, $E(A)$ is made of all the configurations of A and all the transitions between these configurations. Simulation between two guarded automata A and B, noted $A \preceq B$, is defined as simulation between their associated state machines, i.e., $A \preceq B$ iff $E(A) \preceq E(B)$.

3 Description logic with variables

Let N_A and N_R be respectively two disjoint and potentially infinite sets of atomic concept names and role names and let $\mathcal{L}$ be a description logic. Let N_C be an infinite set of concept names including the atomic concept names (i.e., $N_A \subseteq N_C$). Concept descriptions specified in the logic $\mathcal{L}$ (or $\mathcal{L}$-descriptions) are built from concept names N_C and role names N_R using the constructors of $\mathcal{L}$. The semantics of $\mathcal{L}$-descriptions is defined as usual. Let $N_{C_\mathcal{T}} \subset N_C$ and $N_{R_\mathcal{T}} \subset N_R$ be respectively two finite sets of concept names and role names. An *$\mathcal{L}$-terminology* $\mathcal{T}$ over the set of concept names $N_{C_\mathcal{T}}$ and the set of role names $N_{R_\mathcal{T}}$ is a set of concept definitions of the form $A \equiv D$, where $A \in N_{C_\mathcal{T}}$ is a concept name and D is an $\mathcal{L}$-description. Atomic concept names (i.e., elements of N_A) are prohibited from appearing in a left-hand side of a definition while the the concept names occurring of $N_{C_\mathcal{T}} \setminus N_A$, called *defined concepts*, must appear at the left-hand side of a definition. We assume that a terminology does not contain multiple definitions and we allow cyclic definitions. In this paper, we study reasoning in the context of a gfp-semantics [3].

Introducing variables. Let N_A, N_R and N_C defined as previously. For a description logic $\mathcal{L}$, we note by $\mathcal{L}_\mathcal{V}$ the corresponding description logic augmented with variables. To define the logic $\mathcal{L}_\mathcal{V}$, we extend the sets of concept names and roles names with variables. Let N_{cv} and N_{rv} be respectively the sets of concept and role variables. The sets N_{cv}, N_{rv} and $N_R \cup N_C$ are pairwise disjoints. In the sequel, we use the letters $X, Y, \ldots$ to denote variables. An $\mathcal{L}_\mathcal{V}$-terminology is defined over the set of concept terms $N_{C_\mathcal{T}} \subset N_C \cup N_{cv}$ and role terms $N_{R_\mathcal{T}} \subset N_R \cup N_{rv}$. Hence, $\mathcal{L}_\mathcal{V}$-patterns (or simply patterns), are built from concept terms $N_{C_\mathcal{T}}$ and role terms $N_{R_\mathcal{T}}$ using the $\mathcal{L}$-constructors. Therefore, an *$\mathcal{L}_\mathcal{V}$-terminology* is a set of pattern definitions of the form $C \equiv D$, where $C \in N_{C_\mathcal{T}} \setminus (N_A \cup N_{cv})$ is a concept name and D is an $\mathcal{L}_\mathcal{V}$-description. We allow indeed cyclic definitions in $\mathcal{L}_\mathcal{V}$-terminologies.

[3] Note that we adopt a slight different vision of configurations than [5] who considers, for example, a unique initial configuration $id_0 = (q_0, \emptyset)$.

Pattern valuation. Concept (respectively, role) variables take their values from the infinite set of atomic concept names (respectively, role names). This is captured by the notion of valuation. A variable valuation is a mapping $\sigma : N_{cv} \cup N_{rv} \rightarrow N_A \cup N_R$ which maps concept variables into atomic concepts and role variables into role names. We denote by $\mathcal{V}al$ the infinite set of all such possible valuations. Valuations are straightforwardly extended to $\mathcal{L}_\mathcal{V}$-patterns by considering that a valuation is the identity on elements of $N_C \cup N_R$. Continuing with the example of section 1, and taking a valuation σ_1 such that $\sigma_1(X) = University$, $\sigma_1(Y) = works\text{-}for$ and $\sigma_1(Z) = University$ we obtain $\sigma_1(B) \equiv B_1$ and $\sigma_1(B_3) \equiv B_1$. If we consider now a valuation σ_2 such that $\sigma_2(X) = \sigma_2(Z) = eSchool$ and $\sigma_2(Y) = evaluates$ we obtain $\sigma_2(B) \equiv B_2$ and $\sigma_2(B_3) \equiv B_4$. Hence, for an $\mathcal{L}_\mathcal{V}$-pattern C, $\sigma(C)$ is an $\mathcal{L}$-description. As a consequence, an $\mathcal{L}_\mathcal{V}$-pattern C describes a (potentially infinite) set of $\mathcal{L}$-descriptions (corresponding to the potentially infinite number of possible valuations). We extend the notion of valuation to $\mathcal{L}_\mathcal{V}$-teminologies as follows: given an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}$ and a variable valuation σ, we denote by $\sigma(\mathcal{T})$ the terminology made of the definitions of the form $A \equiv \sigma(D)$ such that $A \equiv D$ is a pattern definition in $\mathcal{T}$. Therefore, an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}$ specifies an (infinite) set of $\mathcal{L}$-terminologies $\sigma(\mathcal{T}), \forall \sigma \in \mathcal{V}al$.

Variants of variable semantics. Several possibilities exist to define the semantics of variables depending on the restrictions imposed on variable valuations. We borrow the notion of *non-deterministic reassignment* of variables from variable automata semantics [10,8,5], to define in this paper two classes of variable semantics: refreshing vs. non refreshing variable semantics. The demarcation between these two kinds of semantics lies in the valuation of variables that appear in the scope of a terminological cycle. A non refreshing semantics requires to have a unique valuation of such variables while a refreshing semantics enables to assign different values to the same variable for each *unfolding* of a cycle. To make the meaning of each semantics clear, let us consider again the pattern B_5 given at section 1. A one-step unfolding of B_5 leads to the following pattern description: $Person \sqcap \exists works\text{-}for.X' \sqcap graduated\text{-}from.X' \sqcap \exists has\text{-}relative.(Person \sqcap \exists works\text{-}for.X'' \sqcap graduated\text{-}from.X'' \sqcap \exists has\text{-}relative.B_5)$. A non-refreshing semantics allows only valuations σ that satisfy $\sigma(X') = \sigma(X'')$ while a refreshing semantics permits to have different valuations for X' and X'' (i.e., we may have $\sigma(X') \neq \sigma(X'')$). Indeed, in a non-refreshing semantics, the same variable can be used for both X' and X'' and hence the number of variables used in a (unfolded) definition is always finite. The case of refreshing semantics is different since in this case a cyclic pattern refers (implicitly) to an infinite number of variables. To keep the number of variables finite, we allow the possibility to refresh the values of a given variable during the unfolding of a given definition. For example, a valuation of the pattern B_5 using the assignment σ_i leads to the definition: $\sigma_i(B_5) \equiv Person \sqcap \exists works\text{-}for.\sigma_i(X') \sqcap graduated\text{-}from.\sigma_i(X') \sqcap \exists has\text{-}relative.\sigma_{i+1}(B_5)$.

Hence, the valuation of a $\mathcal{L}_\mathcal{V}$-pattern C is provided by a (potentially infinite) sequence of valuations $\sigma_0, \ldots, \sigma_n$, where σ_0 is the initial valuation and where each valuation σ_{i+1} coincides with the valuation σ_i on the non refreshed variables (i.e.,

$\sigma_i(X') = \sigma_{i+1}(X)'$ if X' is not refreshed) while σ_{i+1} assigns new values to the refreshed variables.

We use the following notation to distinguish between these two classes of semantics: r-semantics denotes refreshing variables semantics (valuations $\mathcal{V}al^r$) while nr-semantics denotes non refreshing variables semantics (valuations $\mathcal{V}al^{nr}$). Let $t \in \{r, nr\}$ and C be an $\mathcal{L}_\mathcal{V}$-pattern. We denote by $\mathcal{D}^t(C)$= $\{\sigma(C) \mid \forall \sigma \in \mathcal{V}al^t \}$ the infinite set of $\mathcal{L}$-descriptions obtained from the pattern C by the valuations of $\mathcal{V}al^t$.

Reasoning in description logics with variables. Let $\mathcal{T}$ and $\mathcal{T}'$ two $\mathcal{L}_\mathcal{V}$-terminologies. We say that $\mathcal{T}$ and $\mathcal{T}'$ are *coherent* if $N_{C_\mathcal{T}} \cap N_{C_{\mathcal{T}'}} = \emptyset$.

Let $t \in \{r, nr\}$. Let C and D be two $\mathcal{L}_\mathcal{V}$-patterns of an $\mathcal{L}_\mathcal{V}$-terminology $\mathcal{T}'$ and let E be an $\mathcal{L}$-description of an $\mathcal{L}$-termonilogy $\mathcal{T}''$ coherent with $\mathcal{T}'$. Let $\mathcal{T}$= $\mathcal{T}' \cup \mathcal{T}''$. In this paper we are interested by the following reasonings.

- E is compliant with C w.r.t. a t-semantics, noted $E \leqslant^t_\mathcal{T} C$, iff there exists a valuation $\sigma \in \mathcal{V}al^t$ such that $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(C)$ (i.e., E is subsumed by $\sigma(C)$ w.r.t. the terminology $\sigma(\mathcal{T})$).
- C is contained in D w.r.t. a t-semantics, noted $C \ll^t_{\mathcal{T}'} D$, iff for every $\mathcal{L}$-description E of an $\mathcal{L}$-termonilogy $\mathcal{T}''$, we have $E \leqslant^t_\mathcal{T} C$ implies $E \leqslant^t_\mathcal{T} D$ (i.e., every description E which is compliant with C w.r.t. a t-semantics is also compiant with D w.r.t. a t-semantics),

4 The case of the logic $\mathcal{EL}_\mathcal{V}$

In this paper, we study reasoning in the context of a gfp-semantics while we believe that our framework can be extended to descriptive and lfp-semantics as well. We are interested by classes of description logics in which reasoning w.r.t. a gfp-semantics can be characterized using a finite state machine (e.g., automata or graphs ([3,12]).

We provide below a characterization of compliance and pattern containment, w.r.t. a gfp-semantics, in $\mathcal{EL}_\mathcal{V}$. We explain briefly how to turn any $\mathcal{EL}_\mathcal{V}$-pattern definition C into a variable automata $\mathcal{A}^t_C$. To achieve this task, we adapt the $\mathcal{EL}$-normal form proposed in [3] to $\mathcal{EL}_\mathcal{V}$-patterns. Let $\mathcal{T}$ be an $\mathcal{EL}_\mathcal{V}$-terminology. An $\mathcal{EL}_\mathcal{V}$-pattern definition $C \equiv D \in \mathcal{T}$ is in a normal form if D is of the form

$$D \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$$

for $m, n \geq 0$ and each V_i is either an atomic concept name or a concept variable (i.e., $V_i \in N_A \cup N_{cv}$, for $i \in [1, m]$), and D_j is a defined concept name (i.e., $D_j \in N_{C_\mathcal{T}} \setminus (N_A \cup N_{cv})$, for $j \in [1, n]$ and each W_i is a role term (i.e., $R_i \in N_{R_\mathcal{T}}$ (R_i). A pattern terminology $\mathcal{T}$ is said *normalized* if all the pattern definitions it contains are in a normal form. W.o.l.g., we assume that two pattern definitions in a normalized terminology use disjoint sets of variables. It is worth noting that, since concept variables do not range over defined concept names, the

normalization process proposed in [3] can be straightforwardly extended to our context to transform any $\mathcal{EL}_{\mathcal{V}}$-terminology into a normalized one. As an example, a normalized terminology containing the defined concepts of our running example is given below:

$B \equiv Person \sqcap \exists works\text{-}for.D_1 \sqcap \exists graduated\text{-}from.D_1$
$B_1 \equiv Person \sqcap \exists works\text{-}for.D_2 \sqcap \exists graduated\text{-}from.D_2$
$B_2 \equiv Person \sqcap \exists works\text{-}for.eSchool \sqcap \exists graduated\text{-}from.eSchool$
$B_4 \equiv Person \sqcap \exists evaluates.D_3 \sqcap \exists graduated\text{-}from.D_3$
$B_5 \equiv Person \sqcap \exists works\text{-}for.D_4 \sqcap \exists graduated\text{-}from.D_4 \sqcap \exists has\text{-}relative.B_5$
$D_1 \equiv X$
$D_2 \equiv University$
$D_3 \equiv eSchool$
$D_4 \equiv X'$

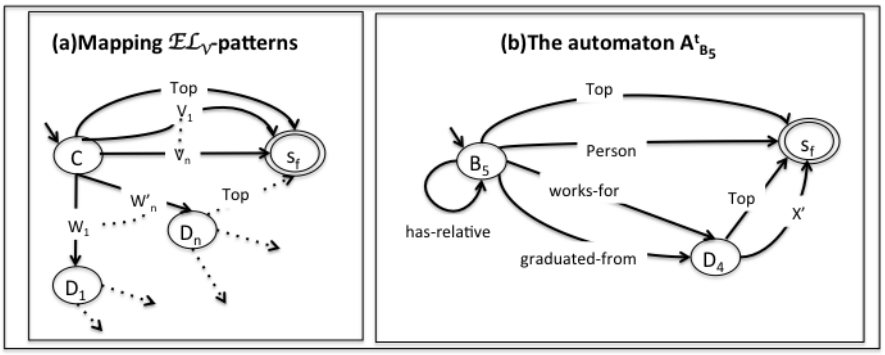


Fig. 1. Mapping $\mathcal{EL}_{\mathcal{V}}$-pattern definitions into variable automata.

Mapping $\mathcal{EL}_{\mathcal{V}}$-patterns into variable automata. Let $\mathcal{T}$ be a normalized $\mathcal{EL}_{\mathcal{V}}$-terminology. We explain below how to map defined concepts and patterns of $\mathcal{T}$ into variable automata. Let $Q_{\mathcal{T}} = N_{C_{\mathcal{T}}} \setminus (N_A \cup N_{cv})$ be a set of states made of the defined concepts of $\mathcal{T}$. Each concept definition or $\mathcal{EL}_{\mathcal{V}}$-pattern $C \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$ of $\mathcal{T}$ is turned into a variable automaton $A^t_C = (\Sigma_C, \mathcal{X}_C, Q_C, Q_0, \delta, F, \mu)$ defined as follows:

- the alphabet $\Sigma_C \subseteq N_A \cup N_R \cup \{Top\}$, is made of a subset of atomic concept names, role names and the Top concept,
- the set of variables $\mathcal{X}_C \subseteq N_{cv} \cup N_{rv}$, is made of a subset of concept and role variables,
- the set of states $Q_C \subseteq Q_{\mathcal{T}} \cup \{C, S_f\}$, is made of the states of $Q_{\mathcal{T}}$ reacheable from the state C,
- $Q_0 = \{C\}$ is the set of initial states of the automaton and $F = \{S_f\}$ is its set of final states,

– the transitions are unguarded (i.e., $\mathcal{G} = \emptyset$). The transition function δ is defined as follows:
  - $(q, Top, true, S_f) \in \delta$, $\forall q \in Q_C$, i.e., there is an edge labeled Top from every node q in Q_C to the final state S_f,
  - $(C, V_i, true, S_f) \in \delta$, $\forall i \in [1, m]$, i.e., each term V_i is turned into an edge, labeled V_i, from the node C to the final state S_f.
  - $(C, W_i, true, D_i) \in \delta$, $\forall i \in [1, n]$, i.e., each term $\exists W_i.D_i$ is turned into an edge, labeled W_i, from the node C to the node D_i.
– the refreshing function μ is defined as follows:
  - if $t = r$ then $\mu(x) = Q_C, \forall x \in \mathcal{X}$
  - if $t = nr$ then $\mu(x) = \emptyset, \forall x \in \mathcal{X}$

The proposed mapping of a pattern $C \equiv V_1 \sqcap \ldots \sqcap V_m \sqcap \exists W_1.D_1 \sqcap \ldots \sqcap \exists W_n.D_n$ into an automaton $A^t_C = (\Sigma_C, \mathcal{X}_C, Q_C, Q_0, \delta, F, \mu)$ is depicted at figure 1(a) while the figure 1(b) shows the variable automaton $A^t_{B_5}$ corresponding to the pattern B_5 of section 1. Figure 2(a) shows $E(A^r_{B_5})$, the extended automaton of $A^r_{B_5}$, for the case of a variable refreshing semantics (i.e. $t = r$). Note that,

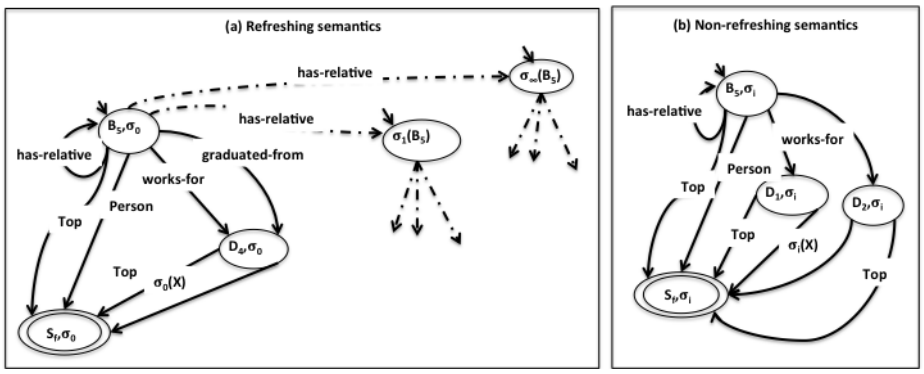


Fig. 2. The extended automaton $E(A^t_{B_5})$.

this machine includes an infinite set of initial states (i.e., the configurations (B_5, σ_i), $\forall \sigma_i \in \mathcal{V}al^r$). In the case of a refreshing semantics, the cyclic role has-relative relates a given state (B_5, σ_i) to all the other possible states (B_5, σ_j), $\forall \sigma_j \in \mathcal{V}al^r$, thereby making each machine rooted at an initial configuration (B_5, σ_i) an infinite state machine. In the case of a non-refreshing semantics (i.e. $t = nr$), $E(A^{nr}_{B_5})$ is made of an infinite set of, pairwise disconnected, finite state machines rooted at (B_5, σ_i), $\forall \sigma_i \in \mathcal{V}al^{nr}$ (c.f., figure 2(b)). The following lemma establishes a connection between subsumption in $\mathcal{EL}_\mathcal{V}$ and simulation between instances of variable automata.

Lemma 1. *Let $t \in \{r, nr\}$ and let C and D be two $\mathcal{EL}_\mathcal{V}$-patterns in a $\mathcal{EL}_\mathcal{V}$-terminology $\mathcal{T}$. Let $\sigma \subseteq \mathcal{V}al^t$, then $\sigma(C) \sqsubseteq_{\sigma(\mathcal{T})} \sigma(D)$ iff $\sigma(A^t_D) \preceq \sigma(A^t_C)$.*

Let $\mathcal{T}$ be an $\mathcal{EL}$-terminology $\mathcal{T}$. This lemma is derived from the observation that the set of all the automata $E(A^t_C)$, of the patterns C of $\mathcal{T}$, corresponds to a slight modification of the notion of $\mathcal{EL}$-description graph [3] of the $\mathcal{EL}$-terminology $\sigma(\mathcal{T})$, where: *(i)* a set of final states (S_f, σ) with $\sigma \in \mathcal{V}al^t$, marked as a final states, is added to the graph, and *(ii)* the atomic concepts are turned into edges from the nodes representing defined concepts to the final states (S_f, σ). It is clear that such a transformation of a description graph of a terminology $\mathcal{T}$ preserves the simulation relation between the nodes of the initial graph. Hence, the characterization of subsumption w.r.t. gfp-semantics using the simulation relation is still valid in our context.

Therefore reasoning over C can be reduced to reasoning over the corresponding variable automata A^t_C. However, despite the correspondance established by lemma 1, reduction of compliance and containment to simulation between variable automata is not straightforward and requires additional transformations. Consider for example the following pattern definitions in a terminology $\mathcal{T}= \{E \equiv \exists r_1.D_3, A \equiv \exists X.D_1 \sqcap \exists Y.D_2, D_i \equiv Top,\ \text{for } i \in \{1,2,3\}\}$. The corresponding automata A^t_E and A^t_A are depicted at figure 3(a). It is easy to check that E is compliant with A w.r.t. any t-semantics. Indeed, for any valuation σ which satisfy $\sigma(X) = \sigma(Y) = r_1$, we have $\sigma(A^t_A) \preceq A^t_E$ and hence $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(A)$ (which implies that E is compliant with A). However, we have clearly $A^t_A \not\preceq A^t_E$. As a witness of non simulation take any valuation σ' which satisfy $\sigma'(X) \neq r_1$ or $\sigma'(Y) \neq r_1$, and in this case we have $A^t_{\sigma'(A)} \not\preceq A^t_{\sigma'(E)}$ (which implies that $A^t_A \not\preceq A^t_E$). Despite this fact, it is still possible to reduce compliance to simulation after a transformation of the automata A^t_A and A^t_E into two new automata, denoted $\bar{A}^t_A$ and $\bar{A}^t_E$ (c.f., figure 3(b)). The main intuition underlying the proposed transformation is to construct new automata that satisfy the following property: $\bar{A}^t_A \preceq \bar{A}^t_E$ iff $\exists \sigma \in \mathcal{V}al^t$ s.t. $\sigma(A^t_A) \preceq A^t_E$. In the similar way, additional transformations are needed to reduce containment to simulation as stated in the following theorem.

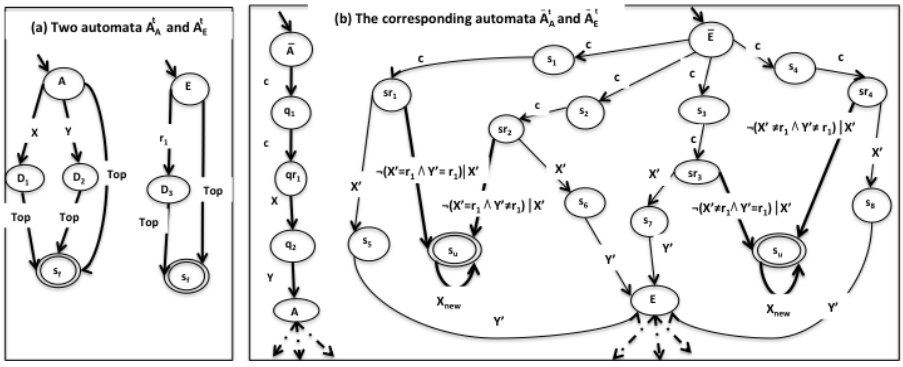


Fig. 3. Reducing compliance to simulation.

Theorem 1. *Let $\mathcal{T}'$ be an $\mathcal{EL}_{\mathcal{V}}$-terminology and C and D two $\mathcal{EL}_{\mathcal{V}}$-patterns in $\mathcal{T}$ and let E be a definition in and $\mathcal{EL}$-terminology $\mathcal{T}''$ coherent with $\mathcal{T}'$. Let $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$. Then there exists guarded variable automata $\hat{A}^t_C, \tilde{A}^t_D, \bar{A}^t_E$ and $\bar{A}^t_C$ such that:*

(i) $E \leqslant^t_{\mathcal{T}} C$ *iff* $\bar{A}^t_C \preceq \bar{A}^t_E$, *and*
(ii) $C \ll^t_{\mathcal{T}} D$ *iff* $\tilde{A}^t_D \preceq \hat{A}^t_C$.

The proof of this theorem is given in the extended version of this paper [13]. We explain below the case (i) (compliance). $E \leqslant^t_{\mathcal{T}} C$ iff $\bar{A}^t_C \preceq \bar{A}^t_E$. Figure 3(b) illustrates the construction of the automata $\bar{A}^t_A$ and $\bar{A}^t_E$ corresponding to the automata of A^t_A and A^t_E of figure 3(b). The main idea is to prefix the automata A^t_A and A^t_E with a set of transitions that enable to ensure that: $\bar{A}^t_A \preceq \bar{A}^t_E$ iff $\exists \sigma \in \mathcal{V}al^t$ s.t. $\sigma(A^t_A) \preceq A^t_E$. The new sr_i and qr_i states correspond to states where all the variables are refreshed. Simulation can be viewed as a game between a player P_E, which moves in the automaton $\bar{A}^t_E$ with an assigned goal to prove simulation, and a player P_A, which moves in the automaton $\bar{A}^t_A$ with an assigned goal to prove that there is no simulation. The rational behind the proposed construction is to enable P_E to force a choice of a valuation σ_t that satisfy simulation if such a valuation exists. This is achieved by the two first transitions labeled with the constant symbol c. At the beginning of the game, P_A choose an arbitrary valuation σ_0 and moves from the state $\bar{A}$ to q_1. The player P_E have a choice between several transitions c, that goes in the example from state $\bar{E}$ to one of the states $s_1, \ldots, s_4$. By this deterministic choice, P_E have the possibility to choose between several classes of valuations, each class being defined w.r.t. to the relation of the variables of the automata A^t_A with the constants that appear in A^t_E. Once such a choice is performed, the player P_A moves upon the constant c to the state qr_1 where he has the possibility to refresh the variables in order to comply with the class of valuation selected by P_E. The rest of the prefix construction enables the player P_E to synchronize his own variables with the variables of P_A in order to be aware of the valuation chosen by P_A. These variables are then used in the guards that enable P_E to enter a *'universal'* final state s_u in the cases where the player P_A cheats (i.e., at the state qr_1, the player P_A picks a valuation that do not belong to the class selected by P_E). Hence, $\bar{A}^t_A \preceq \bar{A}^t_E$ iff there is a valuation σ_t (that belongs to the class of valuations picked by P_E) such that $\sigma_t(A^t_A) \preceq A^t_E$.

Based on the previous theorem, we can provide the following result w.r.t. the considered reasoning in $\mathcal{EL}_{\mathcal{V}}$.

Theorem 2. *Let $t \in \{r, nr\}$. Then:*

(i) Compliance in $\mathcal{EL}_{\mathcal{V}}$ w.r.t. a t-semantics is NP-complete,
(ii) Pattern containment in $\mathcal{EL}_{\mathcal{V}}$ w.r.t.a t-semantics is in 2-Exptime.

Consider first the case (i). We recall that, E is compliant with C w.r.t. a t-semantics, noted $E \leqslant^t_{\mathcal{T}} C$, iff there exists a valuation $\sigma \in \mathcal{V}al^t$ such that $E \sqsubseteq_{\sigma(\mathcal{T})} \sigma(C)$. We know from our construction that $E \leqslant^t_{\mathcal{T}} C$ is equivalent to: $\exists \sigma$ s.t $\sigma(A^t_C) \preceq A^t_E$.

The *Compliance* problem is in *NP*: given an oracle which guess a valuation σ, then it is possible to check in polynomial time if $\sigma(A^t_C) \preceq A^t_E$.

The *NP-hardness* is proved by a reduction from graph 3-colorability problem [11]. Let $G = (V, E)$ be a graph. G is 3-colorable if it is possible to assign to each node in V one of the three colors in such a way that every two nodes connected by an edge have different colors. Starting from G and the three colors $\{r, g, b\}$, we construct a *compliance* problem such that the graph G is 3-colorable iff $E_{color} \leqslant^t_{\mathcal{T}} C_G$. Where E_{color} is a definition in $\mathcal{T}'$ and C_G a $\mathcal{EL}_{\mathcal{V}}$-pattern definition in $\mathcal{T}''$ and $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$.

Construction of the terminology $\mathcal{T}'$. For each color $c \in \{b, r, g\}$, we include in $\mathcal{T}'$ a concept definition E_c as follows:

- $E_b \equiv \exists b.E_{Top}$
- $E_r \equiv \exists r.E_{Top}$
- $E_g \equiv \exists g.E_{Top}$
- $E_{Top} \equiv Top$

In addition, $\mathcal{T}'$ contains the definition $E_{color} \equiv \exists r.E_b \sqcap \exists g.E_b \sqcap \exists r.E_g \sqcap \exists b.E_g \sqcap \exists b.E_r \sqcap \exists g.E_r$.

Construction of the $\mathcal{EL}_{\mathcal{V}}$-terminology $\mathcal{T}''$. Given a graph $G = (V, E)$ with $V = \{n_1, \ldots, n_m\}$, we include in $\mathcal{T}''$ the following set of $\mathcal{EL}_{\mathcal{V}}$-patterns:

- $\{N_i \equiv \exists X_i.N_{Top} \mid for\ i \in [1, m]\}$ and
- $N_{Top} \equiv Top$

In addition, $\mathcal{T}''$ includes the pattern: $C_G \equiv \prod_{(n_i,n_j)\in E} \exists X_i.N_j$.

Let $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$. It is then easy to check that G is 3-colorable iff $E_{color} \leqslant^t_{\mathcal{T}} C_G$.

Complexity of pattern containment (case (ii)) is obtained from theorem 1 which reduces an $\mathcal{EL}_{\mathcal{V}}$-pattern containment test $C \ll^t_{\mathcal{T}} D$ into a simulation test $\tilde{A}^t_D \preceq \hat{A}^t_C$ between two GVA. The automaton $\hat{A}^t_C$ is exponential in the size of C and D. Hence, knowing from [5] that simulation is in EXPTIME, we obtain an immediate 2-EXPTIME upper bound for the $\mathcal{EL}_{\mathcal{V}}$-pattern containment problem.

5 Conclusion

This paper addresses the problems of pattern compliance and containment for description logics with variables. It considers a framework that cater for cyclic terminologies and defines two semantics of variables which differ w.r.t. to the possibility or not to refresh the variables. The paper provides preliminary results regarding the description logic $\mathcal{EL}_{\mathcal{V}}$, obtained from an extension of $\mathcal{EL}$ with variables. Future research work will be devoted to the extension of the approach to more expressive logics.

References

1. F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
2. F. Baader and P. Narendran. Unification of concept terms in description logics. In *ECAI'98*, pages 331–335, 1998.
3. Franz Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI*, pages 325–330, 2003.
4. Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in $\mathcal{EL}$ towards general TBoxes. In *KR'12*, pages 568–572, 2012.
5. Walid Belkhir, Yannick Chevalier, and Michael Rusinowitch. Guarded Variable Automata over Infinite Alphabets. October 2013.
6. F. Baader and R. Küsters. Matching in description logics with existential restrictions. In *DL'99*, Sweden, 1999.
7. Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: Rewriting and optimization (extended version). *CoRR*, abs/1112.0343, 2011.
8. Orna Grumberg, Orna Kupferman, and Sarai Sheinvald. Variable automata over infinite alphabets. In *LATA*, pages 561–572, 2010.
9. J.E. Hopcroft and J.D. Ullman. Formal languages and their relation to automata. *ACM Classic Books Series*, 1969.
10. Michael Kaminski and Daniel Zeitlin. Finite-memory automata with nondeterministic reassignment. *Int. J. Found. Comput. Sci.*, 21(5):741–760, 2010.
11. R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, 1972.
12. R. Küsters. Characterizing the Semantics of Terminological Cycles in $\mathcal{ALN}$ using Finite Automata. In *KR'98*, pages 499–510, 1998.
13. L.Akroun, L. Nourine, and F. Toumani. Reasoning in description logics with variables: preliminary results (extended version). Technical report, Blaise Pascal University, http://www.isima.fr/ftoumani/dlvextended2015.pdf, 2015.

Integrating Ontologies and Planning for Cognitive Systems

Gregor Behnke[1], Pascal Bercher[1], Susanne Biundo[1], Birte Glimm[1], Denis Ponomaryov[2], and Marvin Schiller[1]

[1] Institute of Artificial Intelligence, Ulm University, Germany
[2] A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia

Abstract. We present an approach for integrating ontological reasoning and planning within cognitive systems. Patterns and mechanisms that suitably link planning domains and interrelated knowledge in an ontology are devised. In particular, this enables the use of (standard) ontology reasoning for extending a (hierarchical) planning domain. Furthermore, explanations of plans generated by a cognitive system benefit from additional explanations relying on background knowledge in the ontology and inference. An application of this approach in the domain of fitness training is presented.

1 Introduction

Cognitive systems aim to perform complex tasks by imitating the cognitive capabilities of human problem-solvers. This is typically achieved by combining specialised components, each of which is suited to fulfil a specific function within that system, such as planning, reasoning, and interacting with the user or the environment. Each component requires extensive knowledge of the domain at hand. Traditionally, several representations of knowledge are used by the different components, each well suited for their respective components. As a result, domain knowledge is distributed across various parts of such a system, often in different formalisms. Thus, redundancy and maintaining consistency pose a challenge.

In this paper, we present an approach for using an ontology as the central source of domain knowledge for a cognitive system. The main issue we address is how the planning domain (representing procedural knowledge) and other (ontological) domain knowledge can be suitably combined. The approach uses the ontology and ontological reasoning to automatically generate knowledge models for different components of a cognitive system, such as the planning component or an explanation facility. In particular, the planning domain is automatically extended using ontological background knowledge. The same ontology is used by an explanation mechanism providing coherent textual explanations for the generated plans and associated background knowledge to the user. The planning component uses Hierarchical Task Network (HTN) planning [7, 9], which is well-suited for human-oriented planning.

This paper is organised as follows. We commence with the relevant preliminaries in Section 2. A general outline of the approach and its foundations is presented in Section 3, illustrated by a case study using a real-world fitness training scenario. In

Section 4, the explanation mechanism is described by using accompanying examples from the case study. Related work is discussed in Section 5 and Section 6 concludes the paper.

2 Preliminaries

In the paper, we refer to the Description Logic $\mathcal{ALC}$ [24], which, as we assume, is familiar to the reader. We briefly introduce the relevant concepts of HTN planning that help to understand the context of our work. HTN planning [7, 9] is a discipline of automated planning where tasks are hierarchically organised; tasks are either "primitive" (they can be executed directly) or abstract (also called complex or compound in the literature), i.e., they must be decomposed into sets of subtasks which can in turn be abstract or primitive. Plans are represented by so-called task networks, which are partially ordered sets of tasks. Planning is done in a top-down manner, such that abstract tasks in a task network are refined stepwise into more concrete courses of action. This approach to planning is deemed similar to human problem solving, and thus considered appropriate for use in cognitive systems [4]. HTN planning problems consist of an initial task network, a planning domain – a set of operators (specifying the preconditions and effects of primitive actions) together with a set of decomposition methods (which specify how each abstract task can be decomposed by replacing it with a network of subtasks) – and an initial state (specifying the state of the world before the tasks in the plan are carried out). A decomposition method m is denoted as $\mathcal{A} \mapsto_{\prec} \mathcal{B}_1, \ldots, \mathcal{B}_n$, stipulating that the abstract task $\mathcal{A}$ may be decomposed into the plan containing the subtasks $\mathcal{B}_1$ to $\mathcal{B}_n$ adhering to the partial order $\prec$. The subscript $\prec$ is omitted if no order is defined on the subtasks. Applying a method m to a plan containing $\mathcal{A}$ refines it into a plan where $\mathcal{A}$ is replaced by the subtasks of m with the given order $\prec$. All orderings imposed on $\mathcal{A}$ are inherited by its subtasks $\mathcal{B}_i$. A plan is a solution for a planning problem if it consists of a fully executable network of tasks.

3 Integrating Ontologies and Planning

When bringing ontologies and planning together, a key challenge is to find a representation that suitably links general ontological knowledge with information in the planning domain. The aim is to enable both a specialised reasoner and a planner to play to their strengths in a common application domain of interest, while the consistency of the shared domain model remains ensured. The first part of our approach is to embed planning knowledge (a hierarchical planning domain) into a general ontology of the application domain. To address differences in the formalisms of planning and DL, we present suitable modelling patterns and translation mechanisms. As a second step, an off-the-shelf DL reasoner is applied to infer new knowledge about the planning domain in terms of new decomposition methods. Thanks to the integrated model, the structure of decompositions in the planning domain is accessible to DL reasoning, i.e., new methods are implied using knowledge of both domains. This modelling yields a standard planning domain such that an off-the-shelf planner can be used. We have applied this approach to a fitness scenario and have developed a system which creates a training

plan for a user pursuing some fitness objective. Such a plan defines a training schedule, comprising training and rest days, as well as the exercises and their duration which are necessary to achieve a certain goal, e.g., to train the lower body. A similar scenario is used by Pulido et al. [21], where exercises for physiotherapy of upper-limb injuries are arranged by a planner.

3.1 Embedding Planning Methods into Ontologies

In this section we describe how the knowledge contained in a hierarchical planning domain can be represented in an ontology such that its contents are described declaratively and thus become amenable to logical reasoning.

First, a link between the planning domain and corresponding/additional information in the ontology is defined by a common vocabulary – there is a distinguished set of *task concepts* in the ontology which correspond to planning tasks. If $\mathcal{T}$ is a planning task, then a corresponding task concept is denoted as T. In addition to task concepts, the ontology allows for concepts to model further aspects of the application domain. Task concepts are provided with definitions/told subsumptions in the ontology which are determined by the set of predefined decomposition methods for the counterpart planning tasks. Based on these predefined methods, new ones are derived from the ontology, as further described in Section 3.2. A simple decomposition method $\mathcal{A} \mapsto \mathcal{B}$ is interpreted as a subsumption $B \sqsubseteq A$ between task concepts A, B. This pattern, however, works only for such simple decomposition methods. In general, a method has the form $\mathcal{A} \mapsto \mathcal{B}_1, \ldots, \mathcal{B}_n$ and can be viewed as an instruction that $\mathcal{A}$ is achieved by "executing" the tasks $\mathcal{B}_1, \ldots, \mathcal{B}_n$. When representing such decomposition methods in an ontology one needs to take into account that a decomposition method is a precise definition of the tasks created. It specifies the subtasks required to achieve an abstract task, but simultaneously states that these tasks are also *sufficient*. Ontologies, on the other hand, are built on the open world assumption – representing a task decomposition by the tasks $\mathrm{B}_1, \ldots, \mathrm{B}_n$ is insufficient. The ontology must also contain the explicit information that only these tasks belong to the decomposition. For this purpose we use the `onlysome` construct (see, e.g., Horridge et al. [13]), which is a short-hand notation for a pattern combining the existential and universal restrictions to represent collections of concepts.

Definition 1 *Let r be a role, I an index set, and $\{C_i\}_{i \in I}$ concepts. We define the* `onlysome` *restriction $\mathfrak{O}r.(\{C_i\}_{i \in I})$ by $\mathfrak{O}r.(\{C_i\}_{i \in I}) := \bigsqcap_{i \in I} \exists r.C_i \sqcap \forall r.\left(\bigsqcup_{i \in I} C_i\right)$.*

With the `onlysome` restriction we can give a decomposition method in a definition stating that a collection of tasks corresponds to a task concept to be decomposed. Thus, a method $\mathcal{A} \mapsto \mathcal{B}_1, \ldots, \mathcal{B}_n$ can be expressed by using the axiom $\mathrm{A} \equiv \mathfrak{O}\mathrm{includes}.(\mathrm{B}_1, \ldots, \mathrm{B}_n)$.

Let us now turn to our application domain of fitness training. Here, elementary training exercises are represented as planning tasks whose preconditions and effects follow certain rules (e.g. muscles must be warmed up before being exercised, intense exercises must precede lighter ones, …). The ontology further encompasses concepts for training exercises, equipment, workouts, training objectives, and a part of the NCICB corpus [17], describing muscles etc. Contained in the ontology are four planning-related

types of concepts: exercises, workouts, workout templates, and trainings. *Workouts* are predefined (partially ordered) sets of exercises, modelled by a domain expert using `onlysome`-definitions. This implicitly defines decomposition methods for each workout. Workouts are, e.g., defined by:

$$\text{Workout1} \equiv \text{Ɔincludes.}(\text{FrontSquat}, \text{SumoDeadlift})$$

This axiom postulates that Workout1 includes front squats and sumo deadlifts but nothing else. At a more abstract level, *workout templates* serve to specify groups of workouts with similar properties. For example, there exist many similar variants of squats and deadlifts with similar training effects, which can be grouped together. For instance, consider WorkoutTemplate1:

$$\text{WorkoutTemplate1} \equiv \text{Ɔincludes.}(\text{Squat}, \text{Deadlift})$$

This workout template subsumes Workout1 wrt the ontology (since FrontSquat $\sqsubseteq$ Squat and SumoDeadlift $\sqsubseteq$ Deadlift). As detailed in the next subsection, these subsumptions are the basis for generating corresponding decomposition methods, e.g., in this case a method decomposing the task *WorkoutTemplate1* into *Workout1* which again is decomposed into its concrete subtasks. Finally, *trainings* define abstract training objectives, such as improving strength or training the lower body. Here, the requirements that need to be met are formulated using `onlysome` restrictions. For instance, the following axiom postulates that lower body training contains at least one and only exercises targeting muscles in the lower body:

$$\text{LowerBodyTraining} \equiv \text{Ɔincludes.}\exists\text{engages\_target.}(\exists\text{part\_of.LowerBody})$$

The cornerstone of our approach is to establish a correspondence between subsumptions among task concepts in the ontology and corresponding decompositions in the planning domain. For two task concepts representing collections of tasks using `onlysome`, we require that one is subsumed by the other if and only if the tasks defined by the first serve to achieve all requirements specified by the second. In more detail, a collection $\mathfrak{C}_1$ (representing a set of tasks) should be subsumed by a collection $\mathfrak{C}_2$ if and only if for any task concept (requirement) from $\mathfrak{C}_2$, there is a task concept in $\mathfrak{C}_1$, which achieves it (i.e., $\mathfrak{C}_1$ is subsumed by $\mathfrak{C}_2$), and there are only those task concepts in $\mathfrak{C}_1$ that meet some requirement from $\mathfrak{C}_2$. For this property to hold, we require that in `onlysome` restrictions $\text{Ɔ}r.(\{C_i\}_{i\in I})$ the role r is *independent* of concepts C_i, i.e., has no semantic relationship with them, as captured by the following definition.

Definition 2 *Let $\mathcal{O}$ be an ontology, r a role, and $C_1, \ldots, C_n$ concepts. We call r* independent of $C_1, \ldots, C_n$ wrt $\mathcal{O}$ *if, for any model $\mathcal{I}$ of $\mathcal{O}$ and any binary relation $[s]$ on the domain of $\mathcal{I}$, there is a model $\mathcal{J}$ of $\mathcal{O}$ with the same domain such that r is interpreted as $[s]$ in $\mathcal{J}$ and the interpretation of C_i, $1 \leq i \leq n$, in $\mathcal{I}$ and $\mathcal{J}$ coincides.*

Next, we show that the given intuition holds for the collections defined with the `onlysome` restriction.

Theorem 1 *Let $\mathcal{O}$ be an ontology, $C_1, \ldots, C_m$ concepts satisfiable wrt $\mathcal{O}$, $D_1, \ldots, D_n$ concepts, and r a role independent of $C_1, \ldots, C_m, D_1, \ldots, D_n$ wrt $\mathcal{O}$.*

Then it holds $\mathcal{O} \models \Im r.(C_1, \ldots, C_m) \sqsubseteq \Im r.(D_1, \ldots, D_n)$ *if and only if*
(1) $\forall i, 1 \leq i \leq m,\ \exists j, 1 \leq j \leq n$, *such that* $\mathcal{O} \models C_i \sqsubseteq D_j$ *and*
(2) $\forall j, 1 \leq j \leq n,\ \exists i, 1 \leq i \leq m$, *such that* $\mathcal{O} \models C_i \sqsubseteq D_j$.

Proof (Sketch). The if direction can be shown using monotonicity of existential/universal restrictions and Conditions (1) and (2). Using contraposition, we can show the only-if direction by constructing a countermodel for the subsumption. Since each C_i is satisfiable, there are models of $\mathcal{O}$ with some instance c_i of C_i. Using the negation of Condition (1), there is further a model with an instance x of some C_i that is not an instance of any D_j. We now build a new model as the disjoint union of these models and take an arbitrary element d in the constructed model. Using independence of r, we obtain a model in which r contains $\langle d, x\rangle$ and the tuples $\langle d, c_i\rangle$. We obtain the desired contradiction since d is an instance of $\Im r.(C_1, \ldots, C_m)$, but not an instance of $\forall r.(D_1 \sqcup \ldots \sqcup D_n)$ (due to $\langle d, x\rangle$) and, hence, of $\Im r.(D_1, \ldots, D_n)$. We can proceed similarly for the case of Condition (2) not holding. □

Until now, only the tasks contained in a decomposition method have been regarded, while their partial ordering was ignored. To incorporate it in the ontology, the notion of collections must be extended to partially ordered sets of concepts. Unfortunately, many DLs (also $\mathcal{ALC}$) are not well suited to represent partial orders, since their expressivity is limited by the tree-model property [28], stating that non-tree structures cannot fully be axiomatized. Since our aim is to completely represent the planning domain in the ontology, we propose a syntactic encoding for this information that is opaque to DL reasoners and has no influence on the semantics. A task $\mathcal{A}$ following a task $\mathcal{B}$ is expressed by replacing the concept A in `onlysome` expressions by $\mathrm{A} \sqcup (\bot \sqcap \exists \mathrm{after.B})$. The latter disjunct is trivially unsatisfiable and the given expression is semantically equivalent to just A. This makes order opaque to any reasoner. Preconditions and effects of primitive tasks are modelled using auxiliary roles, making them accessible for logical reasoning, too.

3.2 Extending Planning Domains by DL Inference

Embedding the planning domain into the ontology enables us to infer new decomposition methods using off-the-shelf DL reasoners. More precisely, subsumption relations between task concepts inferred from an ontology $\mathcal{O}$ result in decomposition methods being added to the planning domain. Suppose there are task concepts A and B such that $\mathcal{O} \models \mathrm{B} \sqsubseteq \mathrm{A}$ and there is no other task concept C such that $\mathcal{O} \models \{\mathrm{B} \sqsubseteq \mathrm{C}, \mathrm{C} \sqsubseteq \mathrm{A}\}$. Then, a decomposition method $\mathcal{A} \mapsto \mathcal{B}$ is created in analogy to the way such methods are encoded in the ontology. This simple scheme provides only for methods decomposing into a single subtask. Further, we interpret `onlysome`-definitions provided by the ontology as told decompositions that provide collections of tasks. Besides these two simple cases, we are also interested in knowing whether an abstract task $\mathcal{A}$ can be achieved by combining some task concepts $\mathrm{B}_1, \ldots, \mathrm{B}_n$ into a new decomposition method $\mathcal{A} \mapsto \mathcal{B}_1, \ldots, \mathcal{B}_n$. If so, some concept expression E describing this combination should be subsumed by A. In keeping with the principle of matching task collections described by Theorem 1, we consider combining concepts using `onlysome` restrictions.

The concepts $\mathrm{B}_1, \ldots, \mathrm{B}_n$ describe requirements, which another concept A might fulfil. If for some tasks $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_n$ it holds $\mathcal{O} \models \mathfrak{O}r.(B_1, \ldots, B_n) \sqsubseteq A$, then a decomposition method of $\mathcal{A}$ into the collection $\mathcal{B}_1, \ldots, \mathcal{B}_n$ is created. Let us once again consider our use-case to discuss an example. Consider the definition of lower body training introduced in the previous subsection:

$$\text{LowerBodyTraining} \equiv \mathfrak{O}\text{includes}.\exists\text{engages\_target}.(\exists\text{part\_of.LowerBody})$$

Since our ontology respects the requirement that the role includes is independent of concepts occurring in `onlysome` expressions, Theorem 1 applies and we know that any workout solely comprised of lower body exercises is subsumed by LowerBodyTraining. This results in decomposition methods for LowerBodyTraining into every possible workout for the lower body. Furthermore, consider the following definition of full body training:

$$\begin{aligned}\text{FullBodyTraining} \equiv \mathfrak{O}\text{includes}.(&\\ &\exists\text{engages\_target}.(\exists\text{part\_of.LowerBody}),\\ &\exists\text{engages\_target}.(\exists\text{part\_of.UpperBody}))\end{aligned}$$

Using reasoning, one can now establish whether combinations of different workouts achieve such a training objective, such that a corresponding decomposition method is automatically introduced into the planning domain. For instance, we can infer that a workout solely training the lower body and a workout solely training the upper body in combination constitute this training. Hence a decomposition method for a full body training into these two workouts is added to the planning domain. The following theorem clarifies the relationship between the obtained decomposition methods and entailed concept inclusions.

Theorem 2 *Let $\mathcal{A}$ be an abstract task, which can be refined into some plan $\mathcal{P}$ by applying decomposition methods created from the ontology $\mathcal{O}$. Let the plan $\mathcal{P}$ contain the tasks $\mathcal{B}_1, \ldots, \mathcal{B}_n$, $n \geq 1$. Then there is a concept P such that $\mathcal{O} \models P \sqsubseteq A$, $B_1, \ldots, B_n$ occur in P, and P is either of the form:*

1. *a task concept from $\mathcal{O}$, or*
2. *an expression $\mathfrak{O}r.(F_1, \ldots, F_m)$ with each F_i a concept of the form 1 or 2.*

Proof. The claim is proved by induction on the number m of (decomposition) steps made to obtain $\mathcal{P}$ using $\mathcal{O}$. In the induction base, for $m = 0$, A is the required concept. For $m > 0$, let $\mathcal{P}' = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ be a set of tasks obtained in $m - 1$ decomposition steps and let P' be a concept satisfying the claim for $\mathcal{P}'$. Let $\mathcal{P}$ be obtained from $\mathcal{P}'$ by decomposing some task $\mathcal{C}_i$, for $i \in \{1 \ldots, k\}$. Assume that a decomposition method for $\mathcal{C}_i$ was created from the concept inclusion $D \sqsubseteq C_i$ entailed by $\mathcal{O}$. Then D is one of the task concepts $B_1, \ldots, B_n$, we have $\mathcal{O} \models P'_{C_i/D} \sqsubseteq P'$, hence, $\mathcal{O} \models P'_{C_i/D} \sqsubseteq A$ and thus, $P'_{C_i/D}$ is the required concept (denoting P' with every occurrence of C_i substituted with D). If a decomposition method for $\mathcal{C}_i$ was created from a concept inclusion $\mathfrak{O}r.(D_1, \ldots, D_\ell) \sqsubseteq C_i$ entailed by $\mathcal{O}$ (and possibly obtained from an axiom $C_i \equiv \mathfrak{O}r.(D_1, \ldots, D_\ell)$), then every D_j, $j = 1, \ldots, \ell$, is a task concept among $B_1, \ldots, B_n$ and we have $\mathcal{O} \models P'_{C_i/\mathfrak{O}r.(D_1,\ldots,D_\ell)} \sqsubseteq A$, so $P'_{C_i/\mathfrak{O}r.(D_1,\ldots,D_\ell)}$ is the required concept. □

Taking into account all possible combinations of tasks in the ontology presents a problem, since there are exponentially many. We propose a pragmatic solution. First, the maximal number of task concepts to be combined in `onlysome` expressions can be restricted by some number k. Second, most real-world domains (including our application example) have restrictions on which tasks can be combined. In our case study using the fitness training scenario, we only considered combinations of two task concepts defined by an `onlysome` axiom. This already enabled a considerable number of methods to be inferred.

Essentially, the ontology used in our system consists of two parts $\mathcal{O}_1$ and $\mathcal{O}_2$, with $\mathcal{O}_1$ containing definitions as above and $\mathcal{O}_2$ representing the core knowledge about the subject domain: exercises (being task concepts), training equipment, body anatomy, etc. The ontology is built in such a way that it guarantees independence of the role includes from any concept occurring under $\exists$includes, which means that the principle of matching task collections outlined in Section 3.1 correctly applies. The general shape of our ontology is formally described in the following theorem, where the role includes is abbreviated as r.

Theorem 3 *Let r be a role and $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$ an ontology such that $\mathcal{O}_1$ is an acyclic terminology consisting of definitions $A \equiv \exists r.(C_1, \ldots, C_m)$, where r does not occur in $C_1, \ldots, C_m$ and A, r do not occur in $\mathcal{O}_2$. Then the role r is independent of any concepts appearing under $\exists r$ in $\mathcal{O}_1$.*

Proof (Sketch). Let $\mathcal{A}$ be the set of concept names occurring on the left-hand side of axioms in $\mathcal{O}_1$. Let $\mathcal{I}$ be a model of $\mathcal{O}$ and $[s]$ a binary relation on the domain of $\mathcal{I}$. Let $\mathcal{I}'$ be an interpretation obtained from $\mathcal{I}$ by changing the interpretation of r to $[s]$. Since r and $\mathcal{A}$-concepts do not occur in $\mathcal{O}_2$, we have $\mathcal{I}' \models \mathcal{O}_2$. It remains to consider an expansion of the terminology $\mathcal{O}_1$ (cf. [2, Prop. 2.1]) to verify that there exists a model $\mathcal{J}$ of $\mathcal{O}_1 \cup \mathcal{O}_2$ obtained from $\mathcal{I}'$ by changing the interpretation of $\mathcal{A}$-concepts (and leaving the interpretation of other symbols unchanged). For any concepts $C_1, \ldots, C_m$ appearing under $\exists r$ in $\mathcal{O}_1$, their interpretations in $\mathcal{J}$ and $\mathcal{I}$ coincide, which shows the required statement. □

The initial planning domain of our case-study scenario encompasses 310 different tasks and a few methods. Its formalisation in the ontology consists of 1 230 concepts and 2 903 axioms, of which 613 concepts and 664 axioms are imported from the NCICB corpus. Initially, 9 different training objectives and 24 workout templates are specified. For extending the ontology with inferred decompositions, we employ the OWL reasoner FaCT++ [27], which requires 3.6 seconds on an up-to-date laptop computer (Intel® Core™ i5-4300U). After being extended, the planning domain contains 471 tasks and 967 methods, of which 203 are created based on subsumptions between workouts and workout templates, and further three methods have been created by `onlysome`-combinations of task concepts. In addition, 59 decomposition methods for training objectives into workout templates are created of which 24 are `onlysome`-combinations of task concepts.

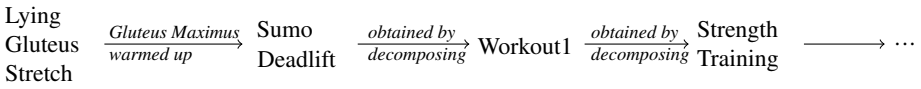


Fig. 1. Example of a plan explanation for the task *lying gluteus stretch*

4 Explanations

Cognitive systems that interact with users need to be able to adequately communicate their solutions and actions. In the field of HCI and dialog modelling, it was shown that systems that provide additional explanations receive increased trust from their users [16, 20]. Bercher et al. [3] empirically investigated the role of plan explanations in an interactive companion system in a real-world scenario.

In our integrated approach, both DL reasoning and planning work together, thereby using the procedural and declarative information contained in the integrated knowledge model, and generating new information artefacts of different flavours (in particular, inferred facts and refined plans). We now describe how explanations are generated from this information by combining techniques from plan explanation (specifically, an approach for explaining hybrid plans [25]) and an approach for explaining ontological inferences [23]. Together they offer complimentary views on a given application domain.

Plan explanation focuses on dependencies between tasks in a plan, in particular how tasks are decomposed into subtasks, and how these tasks are linked by preconditions and effects. To provide an explanation why a particular task is part of a generated plan, the information relevant to justify its purpose is extracted from the causal dependencies and the decompositions applied to the plan. Technically, this information (which internally is represented in a logic formalism) is considered the "explanation". It is guaranteed that plan explanations are always linear chains of arguments, each based on its predecessor. For instance, consider that in our application scenario, the user asks to justify a particular action, for example "Why do I have to do a lying gluteus stretch?" Fig. 1 shows the dependencies that establish the use of the lying gluteus stretch within a plan that achieves a strength training. This information is further converted into text to be communicated to the user by using simple templates. Causal dependencies between two tasks A and B, where A provides a precondition l for B, are verbalised as "A is necessary as it ensures that l, which is needed by B." Similarly, decompositions are justified by patterns such as "Task A is necessary, since it is part of B." In the running example, this yields:

```
The lying gluteus stretch is necessary as it ensures that
the gluteus maximus is warmed up, which is needed by the sumo
deadlift. The sumo deadlift is necessary, since it is part
of the Workout1. The Workout1 is necessary, since it is part
of the strength training. ...
```

The mechanism presented so far represents the part of "traditional" plan explanation. Here, method decompositions are treated as facts (e.g. "Workout1 is part of strength

$$R_{\sqsubseteq\text{Def}}\ \frac{X \sqsubseteq Y \qquad Y \equiv Z}{X \sqsubseteq Z}\ \begin{array}{l}\text{Y is an}\\ \text{atomic}\\ \text{concept}\end{array}$$

"... $\langle\!\langle X \sqsubseteq Y \rangle\!\rangle$. Thus, $\langle\!\langle X \sqsubseteq Z \rangle\!\rangle$ according to the definition of $\langle\!\langle Y \rangle\!\rangle$."

Fig. 2. Sample inference rule (left) together with corresponding text template (right). Guillements indicate the application of a function translating DL formulas into text.

training"); however, in our approach, they can be justified further, since they correspond to subsumptions inferred from background knowledge in the ontology. Therefore, the reasoning behind the subsumption can be used as an explanation to justify the decomposition. For example, the user may ask the question "Why is Workout1 a strength training?" For this purpose, we use the second explanation mechanism, which has been implemented as a prototype. Its aim is to generate stepwise textual explanations for ontological inferences. In the running example, it outputs:

```
According to its definition, Workout1 includes front squat
and sumo deadlift. Furthermore, since sumo deadlift is an
isotonic exercise, it follows that Workout1 includes an isotonic
exercise. Given that something that includes an isotonic
exercise has strength as an intended health outcome, Workout1
has strength as an intended health outcome. Thus, Workout1
is a strength training according to the definition of strength
training.
```

To generate such explanations, first a consequence-based inference mechanism is used to construct a derivation tree for a given subsumption. This is done in two stages. First the relevant axioms in the ontology (the "justifications") are identified using the implementation provided by Horridge [12], which is done efficiently using a standard tableau-based reasoner. Then, a (slower) proof search mechanism using consequence-based rules is applied for building a derivation tree. This tree is then linearised to yield a sequence of explanation steps. The ordering of the explanation steps corresponds to a post-order traversal in the tree structure of inference rule applications (where the inference step that yields the conclusion is taken to be the root). As an example of a consequence-based inference rule used for explanation generation and its corresponding text template, consider Fig. 2. This template generates the last statement in the sample text shown above. Note that even though the presented inference rule is quite simple, it represents a (logical) shortcut, since the conclusion $X \sqsubseteq Z$ could also be obtained in two steps by inferring $Y \sqsubseteq Z$ from the equivalence axiom and then using the transitivity of $\sqsubseteq$. For the generation of explanations, this (logically redundant) rule is given precedence over the "standard" inference rules, in the interest of smaller proofs and greater conciseness of the generated text. By contrast, some inference rules are specified to never generate output, since it would be considered uninformative for a user. Consider the rule deriving $X \sqsubseteq (Y \sqcup Z)$ from $X \sqsubseteq Y$, which is always ignored during text generation. Such considerations provide ample scope for further work into adjusting the generated texts according to pragmatics and user preferences.

To answer the general question how well people understand automatically generated verbalisations of ontological axioms and inferences, studies have already been per-

formed as part of related work. For example, in experiments, Nguyen [18] found that the understandability of verbalised inference steps depends on the employed inference rules, where some kinds of inference rules were found considerably more difficult than others. Furthermore, if the more difficult-to-understand inference rules were verbalised in a more elaborated manner, understanding was improved. Such work hints at a general challenge for empirical evaluations of the general "usefulness" of such verbalisations (to be addressed as future work); their accessibility partially depends on the complexity of the formalised domain and on the prerequisites of the user.

5 Related Work

Past research on coupling ontological reasoning and planning mainly addressed the aim of increasing expressivity/efficiency. There is a large body of research on integrating ontology and action formalisms, see e.g., the overview in Calvanese et al. [6], which aims at bringing together the benefits of static and dynamic knowledge representation. Gil [10] provides a survey of approaches joining classical planning and ontologies and names a number of planners that use ontological reasoning to speed-up plan generation. Hartanto and Hertzberg [11] use a domain-specific ontology to prune a given HTN model. However, additional content in the domain can not be inferred in their paradigm. A number of approaches use ontologies to enrich the structure of the planning domain. Typically, ontologies provide hierarchies of tasks and plans and are often used to represent states under the open world assumption [22, 26]. For a survey, we refer to Sirin [26, Chapter 8]. Further approaches (e.g. [14, 8]) use OWL as a representation language for HTN planning domains but do not employ ontology reasoning to extend these domains. Sirin [26] describes HTN-DL, combining HTN and description logics to solve Web Service composition problems. HTN-DL planning domains are encoded in an ontology by representing tasks as concepts and decomposition methods as individuals. Both are augmented with axioms describing their preconditions and effects. Here Sirin's view of methods differs from standard HTN, as they define a partially ordered list of actions, but not an abstract task they decompose. Further, preconditions and effects are assigned to methods, which are not necessarily related to the contents of the plan. Although Sirin's and our approach are similar in the idea of using an ontology and DL reasoning to generate planning domains, there are conceptual differences. In HTN-DL all decomposition methods are provided in the domain by a modeller, while the presented approach infers new decomposition methods. Sirin applies reasoning to determine whether a decomposition method can be applied to an abstract task, using their preconditions and effects. His approach does not allow inference based on the tasks in a method nor their decompositions or other properties, which is the cornerstone of ours.

Related work on the generation of explanations from ontologies encompasses a number of approaches to verbalise the formalised contents with the goal of imitating natural language. Systems targeting non-expert users include, e.g., the *NaturalOWL* system [1] and the *ontoVerbal* verbaliser [15]. By contrast, the generation of explanations for entailments is addressed only by some approaches, e.g., by Horridge [12], whose approach targets expert users. The generation of stepwise explanations of entailments for non-expert users is considered by Borgida et al. [5], Nguyen et al. [19, 18] and

Schiller and Glimm [23], whose work provided a basis for the approach to explanations presented in this paper.

6 Conclusion

We presented an approach to integrating hierarchical planning knowledge into ontologies that encompass a general representation of the application domain. A central issue of this paper is to establish the semantic correspondence between the constructs of the planning domain (in particular, decomposition methods) and their representation in the ontology. Our results enable a cognitive system to use a coherent knowledge model for both planning and reasoning, which at the same time enables coherent and detailed explanations for the user, as demonstrated in the application scenario. Our investigation also highlights avenues for future work. One such topic is incorporating mixed-initiative planning into the approach, such that communication with and participation of the user would benefit from explanations by the system. Second, the explanations generated by the system raise the issue of selecting the right level of verbosity. Future work should address this from the viewpoint of pragmatics (e.g. that explanations for "obvious" inferences should generally be omitted) and user modelling (e.g. taking prior knowledge of the user into account). While our approach enables the hierarchical structure of a planning domain to be exploited by DL reasoning, the partial order of tasks is not amenable to DL reasoning. The question how this limitation could be addressed can be taken up by future work.

Acknowledgements. This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "A Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

Bibliography

[1] Androutsopoulos, I., Lampouras, G., Galanis, D.: Generating natural language descriptions from OWL ontologies: The NaturalOWL system. JAIR 48, 671–715 (2013)

[2] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)

[3] Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: Plan, repair, execute, explain - How planning helps to assemble your home theater. In: Proc. of the Int. Conf. on Automated Planning and Scheduling. pp. 386–394. AAAI Press (2014)

[4] Biundo, S., Bercher, P., Geier, T., Müller, F., Schattenberg, B.: Advanced user assistance based on AI planning. Cognitive Systems Research 12(3-4), 219–236 (2011), Special Issue on Complex Cognition

[5] Borgida, A., Franconi, E., Horrocks, I.: Explaining ALC subsumption. In: Proc. of the European Conf. on Artificial Intelligence. pp. 209–213. IOS Press (2000)

[6] Calvanese, D., De Giacomo, G., Montali, M., Patrizi, F.: Verification and synthesis in description logic based dynamic systems. In: Proc. of the 7th Int. Conf. on Web Reasoning and Rule Systems (RR 2013). LNCS, vol. 7994, pp. 50–64. Springer (2013)

[7] Erol, K., Hendler, J.A., Nau, D.S.: Complexity results for HTN planning. Annals of Mathematics and Artificial Intelligence 18(1), 69–93 (1996)

[8] Freitas, A., Schmidt, D., Panisson, A., Meneguzzi, F., Vieira, R., Bordini, R.H.: Semantic representations of agent plans and planning problem domains. In: Engineering Multi-Agent Systems, pp. 351–366. Springer (2014)

[9] Geier, T., Bercher, P.: On the decidability of HTN planning with task insertion. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 1955–1961. AAAI Press (2011)

[10] Gil, Y.: Description logics and planning. AI Magazine 26(2), 73–84 (2005)

[11] Hartanto, R., Hertzberg, J.: Fusing DL reasoning with HTN planning. In: KI 2008: Advances in Artificial Intelligence. LNCS, vol. 5243, pp. 62–69. Springer (2008)

[12] Horridge, M.: Justification Based Explanations in Ontologies. Ph.D. thesis, University of Manchester, Manchester, UK (2011)

[13] Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.: The Manchester OWL syntax. In: Proc. of the Workshop on OWL Experiences and Directions. Athens, GA, USA (2006)

[14] Ko, R.K.L., Lee, E.W., Lee, S.G.: Business-OWL (BOWL) – A hierarchical task network ontology for dynamic business process decomposition and formulation. IEEE Transactions on Services Computing 5(2), 246–259 (2012)

[15] Liang, S.F., Scott, D., Stevens, R., Rector, A.: OntoVerbal: a generic tool and practical application to SNOMED CT. Int. J. of Advanced Computer Science and Applications 4(6), 227–239 (2013)

[16] Lim, B.Y., Dey, A.K., Avrahami, D.: Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: Proc. of the SIGCHI Conf. on Human Factors in Comp. Systems. pp. 2119–2128 (2009)
[17] NCICB, N.: (2015), http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl (accessed February 9, 2015)
[18] Nguyen, T.A.T.: Generating Natural Language Explanations For Entailments In Ontologies. Ph.D. thesis, The Open University, Milton Keynes, UK (2013)
[19] Nguyen, T.A.T., Power, R., Piwek, P., Williams, S.: Predicting the understandability of OWL inferences. In: The Semantic Web: Semantics and Big Data, LNCS, vol. 7882, pp. 109–123. Springer (2013)
[20] Nothdurft, F., Richter, F., Minker, W.: Probabilistic human-computer trust handling. In: Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue. pp. 51–59. Association for Computational Linguistics (2014)
[21] Pulido, J.C., González, J.C., González-Ferrer, A., García, J., Fernández, F., Bandera, A., Bustos, P., Suárez, C.: Goal-directed generation of exercise sets for upper-limb rehabilitation. In: Proc. of the Workshop on Knowledge Engineering for Planning and Scheduling (2014)
[22] Sánchez-Ruiz, A.A., González-Calero, P.A., Díaz-Agudo, B.: Abstraction in knowledge-rich models for case-based planning. In: Case-Based Reasoning Research and Development, LNCS, vol. 5650, pp. 313–327. Springer (2009)
[23] Schiller, M., Glimm, B.: Towards explicative inference for OWL. In: Proc. of the Int. Description Logic Workshop. vol. 1014, pp. 930–941. CEUR (2013)
[24] Schmidt-Schauss, M., Smolka, G.: Attributive concept descriptions with complements. AIJ 48, 1–26 (1991)
[25] Seegebarth, B., Müller, F., Schattenberg, B., Biundo, S.: Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In: Proc. of the Int. Conf. on Automated Planning and Scheduling. pp. 225–233. AAAI Press (2012)
[26] Sirin, E.: Combining Description Logic Reasoning with AI Planning for Composition of Web Services. Ph.D. thesis, University of Maryland at College Park (2006)
[27] Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: Proc. of the Third Int. Joint Conf. on Automated Reasoning (IJCAR). pp. 292–297. Springer (2006)
[28] Vardi, M.Y.: Why is modal logic so robustly decidable? In: Descriptive Complexity and Finite Models. vol. 31, pp. 149–184. American Mathematical Society (1997)

Optimized Construction of Secure Knowledge-Base Views

P. A. Bonatti, I. M. Petrova and L. Sauro

Dept. of Electrical Engineering and Information Technologies
Università di Napoli "Federico II"

Abstract. In this paper we examine a confidentiality framework for constructing safe KB views with respect to the object-level and the meta-level background knowledge that users may exploit to reconstruct secrets. In particular, we will present a first implementation of our framework equipped with several optimization techniques that are assessed experimentally in a concrete e-health scenario.

1 Introduction

Recently, the Semantic Web has been increasingly used to encode sensible knowledge on individuals, companies and public organizations. As reasoning techniques make it possible to extract implicit information, any access control method that does not deal with inference fails to ensure privacy [1, 10].

The most popular security criterion is that the published view of a knowledge base should not entail any secret sentence [3, 9, 14]. However, such a model guarantees confidentiality just in the case the filtered knowledge base is the only source of information. On the contrary, various sources of background knowledge can be exploited to reconstruct secrets. Background knowledge can be object-level knowledge of the domain of interest, e.g. auxiliary ontologies, as well as meta knowledge about which kind of information the knowledge base is expected to represent. For instance, suppose a hospital allows to know whether a patient has been hospitalized but omits to reveal where, if she is in the infective disease ward. Since a hospital's *KB* is expected to have complete knowledge about which patients are in which ward, from the fact that John has been admitted to the hospital and yet he does not appear to be located in any ward, a user can reconstruct he is affected by some infection.[1]

To tackle the vulnerabilities arising from these scenarios, [7] has provided a fully generic formalization of object-level and meta-level background knowledge, a confidentiality model which neutralizes the inference-based attacks that exploit such knowledge, and – since the user's background knowledge is not directly possessed by the knowledge engineer – a rule-based methodology to safely approximate it.

As the works in [11, 12], our model is inspired by the literature on *Controlled Query Evaluation* ([4, 5, 6]). However, the two approaches differ in many aspects, including: *(i)* [11, 12] focus on conjunctive queries, while we focus on subsumption and instance

[1] For further details see the analogous Example 1 in [7]. In general, meta knowledge helps in preventing *attacks to complete knowledge* and *attacks to the signature*.

checking; *(ii)* in our framework secrets can be both intensional and extensional axioms, whereas in [11, 12] they can only be extensional facts; *(iii)* although [11, 12] can deal with object-level background knowledge, meta knowledge is not taken into account.

Regarding complexity issues, in [7] it has been shown that by using Horn rules to encode the user's meta knowledge, if the underlying DL is tractable, then the filtering secure function is tractable too.[2] Although such promising theoretical properties suggest that the framework can be practically used, they are still to be assessed experimentally. In this paper, we present SOVGen, a first prototype suited for a concrete e-health scenario. In particular, extensional data is encoded in realistic electronic health records conforming to the standard HL7 v.3 - CDA Rel.2. We approximate the user's background knowledge with the SNOMED-CT ontology, together with an ontology establishing the mapping between SNOMED-CT concepts and ICD-9CM codes that occur in the records. The user's meta knowledge, on the other hand, consists of *(i)* bridge metarules that permit to identify SNOMED-CT concepts starting from the specific encoding of the records required by CDA, as well as *(ii)* metarules that establish relationships between medications, diseases, medical procedures, etc.

Sec. 2 will provide a general overview on the theoretical model; due to space limitations we refer to [7] for technical proofs. In Sec. 3 we will describe the algorithm underlying SOVGen together with its optimizations. Sections 4 and 5 describe the experimental settings and performance analysis, respectively. Sec. 6 concludes the paper.

2 The Model

We assume the reader to be familiar with description logics, and refer to [2] for all definitions and results. We assume a fixed, denumerable signature Σ, specifying the names of *concepts*, *roles*, and *individuals*, and a reference logical language $\mathcal{L}$ is generated from Σ by the grammar of a DL. Unless stated otherwise, by *axioms* we mean members of $\mathcal{L}$; a *knowledge base* is any finite subset of $\mathcal{L}$. The notion of *logical consequence* is the classical one; for all $K \subseteq \mathcal{L}$, the logical consequences of K will be denoted by $Cn(K)$ ($K \subseteq Cn(K) \subseteq \mathcal{L}$).

Let KB be a knowledge base and $S \subseteq \mathcal{L}$ a set of secrecies. Generally speaking, the confidentiality of S is preserved if a user cannot expect to discover any secret by querying the system. A possible attempt to protect the secrets is to use a view KB' which is a maximal subset of KB that entails no secret, $Cn(KB') \cap S = \emptyset$.

Unfortunately, in case some background knowledge is available to the user, this mechanism could not ensure confidentiality. Frequently, part of the domain knowledge is not axiomatized in KB. In such cases a user can import some external ontology or RDF repository BK to infer more than she is allowed to. Moreover, she may possess some meta knowledge about KB. For instance, a hospital's KB is expected to have complete knowledge about its patients; a company's KB is likely to encode complete information about its employees, etc. Such meta knowledge can be represented epistemically as a set of possible knowledge bases PKB, queries can be then used to narrow PKB until the user is able to reconstruct a secret.

[2] Non-Horn metarules can be safely approximated with Horn metarules; the price to pay is a loss of *cooperativeness*, i.e. a reduction of the information available to the user.

Summarizing, we introduce a general confidentiality model which takes into account object-level and meta-level background knowledge.

Definition 1 ([7]). *A* bk-model *is a tuple* $\mathcal{M} = \langle KB, f, S, PKB, BK\rangle$ *where* KB *is a knowledge base,* $f : \wp(\mathcal{L}) \to \wp(\mathcal{L})$ *is a* filtering function *mapping each knowledge base* K *on a view* $f(K) \subseteq Cn(K)$*,* $S \subseteq \mathcal{L}$ *is a set of secrecies,* $BK \subseteq \mathcal{L}$ *is a set of axioms encoding the users' object-level knowledge,* $PKB \subseteq \wp(\mathcal{L})$ *is a set of possible knowledge bases encoding users' meta knowledge.*

The view of KB released to a user is $f(KB)$. Intuitively, f is secure if for each secret s there exists a possible knowledge base $K \in PKB$ such that *(i)* KB and K have the same observable behavior, that is, as far as the user knows, the knowledge base might be K, and *(ii)* K and the object-level background knowledge BK do not suffice to entail s.

Definition 2. *A filtering function* f *is* secure *(w.r.t.* $\mathcal{M}$*) iff for all* $s \in S$*, there exists* $K \in PKB$ *such that 1)* $f(K) = f(KB)$ *and 2)* $s \notin Cn(K \cup BK)$*.*

In the rest of the paper we focus on concrete scenarios where all the components of bk-models are finite. Moreover, we tacitly assume that no secret is violated a priori, that is, for all secrets $s \in S$ there exists $K \in PKB$ such that $s \notin Cn(K \cup BK)$.[3]

Clearly, Definition 2 just formalizes our desiderata, consequently the next step is to exhibit a *secure filtering function*. This function is formulated as an iterative process where for each axiom that, according to the user's meta knowledge, may possibly occur in the knowledge base a *censor* decides whether it should be obfuscated to protect confidentiality. The iterative construction manipulates pairs $\langle X^+, X^-\rangle \in \wp(\mathcal{L}) \times \wp(\mathcal{L})$ that represent a meta constraint on possible knowledge bases: we say that a knowledge base K *satisfies* $\langle X^+, X^-\rangle$ iff K entails all the sentences in X^+ and none of those in X^- (formally, $Cn(K) \supseteq X^+$ and $Cn(K) \cap X^- = \emptyset$).

Let PAX (the set of *possible axioms*) be the set of all axioms occurring in at least one possible knowledge base, i.e. $PAX = \bigcup_{K' \in PKB} K'$. Let $\nu = |PAX|$ and $\alpha_1, \ldots, \alpha_i, \ldots, \alpha_\nu$ be any enumeration of PAX. The secure view construction for a knowledge base K in a bk-model $\mathcal{M}$ consists of the following, inductively defined sequence of pairs $\langle K_i^+, K_i^-\rangle_{i\geq 0}$:

- $\langle K_0^+, K_0^-\rangle = \langle \emptyset, \emptyset\rangle$, and for all $1 \leq i < \nu$, $\langle K_{i+1}^+, K_{i+1}^-\rangle$ is defined as follows:
  - if $censor_{\mathcal{M}}(K_i^+, K_i^-, \alpha_{i+1}) = true$ then let $\langle K_{i+1}^+, K_{i+1}^-\rangle = \langle K_i^+, K_i^-\rangle$;
  - if $censor_{\mathcal{M}}(K_i^+, K_i^-, \alpha_{i+1}) = false$ and $K \models \alpha_{i+1}$ then $\langle K_{i+1}^+, K_{i+1}^-\rangle = \langle K_i^+ \cup \{\alpha_{i+1}\}, K_i^-\rangle$;
  - otherwise let $\langle K_{i+1}^+, K_{i+1}^-\rangle = \langle K_i^+, K_i^- \cup \{\alpha_{i+1}\}\rangle$.

Finally, let $K^+ = \bigcup_{i\leq \nu} K_i^+$, $K^- = \bigcup_{i\leq \nu} K_i^-$, and $f_{\mathcal{M}}(K) = K^+$. The iterative construction aims at finding maximal sets K^+ and K^- that *(i)* partly describe what does / does not follow from K (as K satisfies $\langle K^+, K^-\rangle$ by construction), and *(ii)* do not trigger the censor (the sentences α_{i+1} that trigger the censor are included neither in K^+ nor in K^-).

In order to define the censor we need an auxiliary definition that captures all the consequences of the background knowledge BK and the meta knowledge PKB refined by a constraint $\langle X^+, X^-\rangle$. Let $Cn_{\mathcal{M}}(X^+, X^-)$ be the set of all axioms $\alpha \in \mathcal{L}$ such that

$$\text{for all } K' \in PKB \text{ such that } K' \text{ satisfies } \langle X^+, X^-\rangle, \alpha \in Cn(K' \cup BK)\,. \tag{1}$$

[3] Conversely, no filtering function can conceal a secret that is already known by the user.

Now the censor is defined as follows. For all $X^+, X^- \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$,

$$censor_{\mathcal{M}}(X^+, X^-, \alpha) = \begin{cases} true & \text{if there exists } s \in S \text{ s.t. either } s \in Cn_{\mathcal{M}}(X^+ \cup \{\alpha\}, X^-) \\ & \text{or } s \in Cn_{\mathcal{M}}(X^+, X^- \cup \{\alpha\}); \\ false & \text{otherwise.} \end{cases} \quad (2)$$

In other words, the censor checks whether telling either that α is derivable or not to a user – aware that the knowledge base satisfies $\langle X^+, X^- \rangle$ – restricts the set of possible knowledge bases enough to conclude that a secret s is entailed by the knowledge base enriched with the background knowledge *BK*.

Note that the censor obfuscates α_{i+1} if *any* of its possible answers entail a secret, independently of the actual contents of *K* (the possible answers "yes" and "no" correspond to conditions $s \in Cn_{\mathcal{M}}(X^+ \cup \{\alpha\}, X^-)$ and $s \in Cn_{\mathcal{M}}(X^+, X^- \cup \{\alpha\})$, respectively). This way, roughly speaking, the knowledge bases that entail *s* are given the same observable behavior as those that don't. Thm 1 in [7] shows that $f_{\mathcal{M}}$ is secure w.r.t. $\mathcal{M}$.

Remark 1. Observe that our method is inspired by CQE based on lies and/or refusals ([4, 5, 6] etc). Technically we use *lies*, because rejected queries are not explicitly marked. However, our censor resembles the classical refusal censor, so the properties of $f_{\mathcal{M}}$ are not subsumed by any of the classical CQE methods. For example (unlike the CQE approaches that use lies), $f_{\mathcal{M}}(KB)$ encodes only correct knowledge (i.e. entailed by *KB*), and it is secure whenever users do not initially know any secret (while lies-based CQE further require that no *disjunction* of secrets should be known a priori). Unlike the refusal method, $f_{\mathcal{M}}$ can handle *cover stories* because users are not told that some queries are obfuscated. As an additional advantage, our method needs not to adapt existing engines to handle nonstandard answers like *mum*. Finally, the CQE approaches do not deal specifically with DL knowledge bases, nor meta knowledge.

Of course, the actual confidentiality of a filtering $f(KB)$ depends on a careful definition of the user's background knowledge, that is, *PKB* and *BK*. If background knowledge is not exactly known by the knowledge engineer then it can be safely overestimated. More background knowledge means larger *BK* and smaller *PKB*, which leads to the following comparison relation $\leq_k$ over bk-models:

Definition 3. *Let* $\mathcal{M} = \langle KB, f, S, PKB, BK \rangle$ *and* $\mathcal{M}' = \langle KB', f', S', PKB', BK' \rangle$ *be two bk-models, we write* $\mathcal{M} \leq_k \mathcal{M}'$ *iff* $KB = KB'$, $f = f'$, $S = S'$, $PKB \supseteq PKB'$ *and* $BK \subseteq BK'$.

Then, it is easy to see that $\mathcal{M}'$ is a safe approximation of $\mathcal{M}$, that is if f is secure w.r.t. $\mathcal{M}'$, then it is also secure w.r.t. $\mathcal{M}$ (Proposition 2, [7]).
Consequently, a generic advice for estimating *BK* consists in *(i)* including public ontologies and triple stores formalizing relevant knowledge and *(ii)* modeling as completely as possible the integrity constraints satisfied by the data, as well as role domain and range restrictions and disjointness constraints.

While *BK* can be represented with standard languages (e.g. OWL, RDF, etc.), user's meta knowledge requires an ad-hoc language for defining *PKB*. Here we express *PKB* as the set of all theories that are contained in a given set of *possible axioms PAX* and satisfy a finite set *MR* of *metarules* like:

$$\alpha_1, \ldots, \alpha_n \Rightarrow \beta_1 \mid \ldots \mid \beta_m \quad (n \geq 0, m \geq 0)\,, \quad (3)$$

where all α_i and β_j are in $\mathcal{L}$ ($1 \leq i \leq n$, $1 \leq j \leq m$). For all metarules r, let $body(r) = \{\alpha_1, \ldots, \alpha_n\}$ and $head(r) = \{\beta_1, \ldots, \beta_m\}$.

Informally, (3) means that if KB entails $\alpha_1, \ldots, \alpha_n$ then KB entails also some of $\beta_1, \ldots, \beta_m$. Sets of similar metarules can be succintly specified using *metavariables*; they can be placed wherever individual constants may occur, that is, as arguments of assertions, and in nominals. A metarule with such variables abbreviates the set of its *ground instantiations*: Given a $K \subseteq \mathcal{L}$, let $ground_K(MR)$ be the ground instantiation of MR where metavariables are uniformly replaced by the individual constants occurring in K in all possible ways.

A set of axioms $K \subseteq \mathcal{L}$ *satisfies* a ground metarule r if either $body(r) \nsubseteq Cn(K)$ or $head(r) \cap Cn(K) \neq \emptyset$. In this case we write $K \models_m r$. Moreover, if K satisfies all the metarules in $ground_K(MR)$ then we write $K \models_m MR$. Therefore the formal definition of PKB now becomes:

$$PKB = \{K \mid K \subseteq PAX \wedge K \models_m MR\}\,. \tag{4}$$

In this paper, we assume that MR consists of Horn metarules ($|head(r)| \leq 1$) and $PAX = KB \cup \bigcup_{r \in ground_{KB}(MR)} head(r)$. Under such hypothesis, it can be shown that if all the axioms in KB, PKB, BK, and S belong to a tractable DL, and the number of distinct variables in MR is bounded by a constant, then $f_{\mathcal{M}}$ can be calculated in polynomial time.

3 Implementation overview

In this section we introduce SOVGen, the prototypical implementation of the confidentiality model illustrated in Section 2 based on Horn metarules. By standard logic programming techniques, a minimal $K \subseteq PAX$ satisfying the set of metarules and the constraints K^+ can be obtained with the following polynomial construction:

$$K_0 = K^+\,, \quad K_{i+1} = K_i \cup \bigcup \{\, head(r) \mid r \in ground_{K_i}(MR) \wedge body(r) \subseteq Cn(K_i) \,\}$$

It can be proved that the sequence limit $K_{|PAX|}$ satisfies $\langle K^+, K^- \rangle$ as well if $K_{|PAX|}$ does not entail an axiom in K^-. Then, for all $s \in S$, s activates the censor iff s is a consequence of $K_{|PAX|} \cup BK$. For further details refer to [7].

Algorithm 1 represents the abstract algorithm underlying SOVGen. The sets M_M and M_G constitute a partition of MR based on the metarules' type (ground or containing metavariables). Iterating over the axioms $\alpha \in PAX$ (*lines 6-25*), at each step K collects all the axioms of PAX that does not contribute to the entailment of secrets. The repeat-until loop (*lines 9-17*) computes the deductive closure K' of K under the set of metarules MR. In particular, for each ground metarule (*lines 10-13*) we evaluate a conjunctive query (encoded in *line 11*) in order to check if m body is satisfied by the current K'. Similarly, for each metarule containing metavariables (*lines 14-16*), we obtain all possible bindings for the metavariables in the body of m by means of a conjunctive query evaluation (*line 15*). The sequence of steps described above is iterated until a fixpoint is reached (*line 17*). At this point the condition $Cn(K') \cap K^- \models \emptyset$ is verified (*line 18*). It is now possible to determine the value of the censor for α. We first check that no secret is entailed from the minimal K (*line 19*) enreached with BK. Finally, we can safely include α in the view only if it is entailed by KB (*line 21*). Otherwise, the set K^-

Algorithm 1:

Data: KB, S, MR, BK.

1 $K_i^+, K_i^- \leftarrow \emptyset$;
2 $M_M \leftarrow \{r_i | r_i \in MR$ and r_i metarule containing metavariables$\}$;
3 $M_G \leftarrow \{r_i | r_i \in MR$ and r_i ground metarule$\}$;
4 $PAX \leftarrow KB \cup \bigcup_{r \in ground_{KB}(MR)} head(r)$;
5 $K \leftarrow BK$;
6 **forall** $\alpha \in PAX$ **do**
7 $K' \leftarrow K \cup \{\alpha\}$;
8 $M'_G \leftarrow M_G$;
9 **repeat**
10 **forall** $m \in M'_G$ **do**
11 **if** $K' \models body(m)$ **then**
12 $K' \leftarrow K' \cup \{head(m)\}$;
13 $M'_G \leftarrow M'_G \setminus \{m\}$;
14 **forall** $m \in M_M$ **do**
15 **forall** $(a_0, \ldots, a_n) \mid K' \models body(m, [X_0/a_0, \ldots, X_n/a_n])$ **do**
16 $K' \leftarrow K' \cup \{head(m, [X_0/a_0, \ldots, X_n/a_n])\}$;
17 **until** *No element is added to* K';
18 **if** $\{\beta \in K^- \mid K' \models \beta\} = \emptyset$ **then**
19 **if** $\{s \in S \mid K' \cup BK \models s\} = \emptyset$ **then**
20 **if** $KB \models \alpha$ **then**
21 $K^+ \leftarrow K^+ \cup \{\alpha\}$;
22 $K \leftarrow K'$;
23 $M_G \leftarrow M'_G$;
24 **else**
25 $K^- \leftarrow K^- \cup \{\alpha\}$;
26 **return** K_i^+

is updated (*line 25*). Note that, due to the monotonicity of reasoning, at each iteration we can safely remove from M_G all the ground rules already satisfied at the previous iterations (*lines 13, 23*).

A careful analysis of the algorithm immediately points out: *(1)* the opportunity to apply a process of modularization designed to reduce the size of very large background knowledge bases (such as SNOMED-CT). In fact, many of the axioms in a large BK are reasonably expected to be irrelevant to the given view; *(2)* the need of techniques for effective conjunctive query evaluation.[4]

With respect to point (1), we investigate the use of *module extractors* [17, 16] on the background knowledge bases in order to make reasoning focus on relevant knowledge only. Experimental results show that the modules extracted are on average two

[4] Straightforward evaluation of metarules in the presence of metavariables with an OWL reasoner would need to consider all possible ways of uniformly replacing metavariables by individual constants occurring in the ontology.

or three orders of magnitude smaller than the initial BKs which drastically improves performance.

With respect to point (2), the presence of technologies that permit native conjunctive query evaluation reveals fundamental to achieve efficient framework implementation. Nowadays SPARQL[5], constitute a de facto standard when it comes to conjunctive query answering. It has been recently extended with the OWL Direct Semantics Entailment Regime in order to permit reasoning over OWL ontologies. Unfortunately, only few tools provide support to this new semantics. Among those our choice fell on *Apache Jena Semantic Web Toolkit*[6] (for more information and motivations see [8]). A valid alternative to the consolidated SPARQL engines proves to be *OWL-BGP*[7], a relatively new framework for parsing SPARQL basic graph patterns (BGPs) to OWL object representation and their assessment under the OWL Direct Semantics Entailment Regime. *OWL-BGP* incorporates various optimization techniques [15] including query rewriting and a cost-based model[8] for determining the order in which conjunctive query atoms are evaluated. As we will see in Section 5 the performance of the query evaluation module of SOVGen is unacceptable when Jena is used and not quite satisfactory when *OWL-BGP* is adopted[9]. As an alternative to the above frameworks for conjunctive query evaluation we propose an hoc module, called *Metarule Evaluation Engine (MEE)*, that aims to take advantage of the specific nature of the Horn metarules and incremental reasoning techniques of ELK [13].

Metarule Evaluation Engine (MEE). The evaluation algorithm is based on direct calls to an incremental reasoner. In the following we provide a brief description of the procedure employed for the evaluation of the different types of metarules.

The evaluation of a ground metarule r requires checking that all the axioms $\alpha_1, \ldots, \alpha_n$ in $body(r)$ are entailed by $K^{'}$. The algorithm takes advantage of *short circuit evaluation* techniques that permit to end the evaluation as soon as $K^{'} \not\models \alpha_i$ and memoization of the atoms α_i satisfied in previous iterations in order to avoid their re-evaluation.

The evaluation of metarules with metavariables, on the other hand, comprises a preprocession step that partition the atoms $\alpha_1, \ldots, \alpha_n$ in the metarule body in sets of *connected components*. Within a component, atoms (that in this case can be viewed as axiom templates) share common metavariables, while there are no metavariables shared between atoms belonging to different *connected components*. Evaluating together templates belonging to non-related components increases unnecessarily the amount of intermediate results, whereas it is sufficient to combine the results for the single components. Furthermore, for some types of templates, such as $C(X)$, it is possible to retrieve the solutions directly from the reasoner, instead of verifying the satisfiability of each compatible mapping for the metavariable X. Although this can trigger some internal controls, most of the methods of reasoners are highly optimized. Other more complex

[5] http://www.w3.org/TR/sparql11-overview/

[6] http://jena.apache.org/

[7] https://code.google.com/p/owl-bgp/

[8] The cost calculation is based on information about instances of concepts and roles extrapolated from an abstract model built by reasoners that implement Tableaux reasoning algorithms.

[9] Note that evaluation of ground metarules results in *SPARQL ASK* query (line11 of Alg.1), while evaluation of metarules with metavariables in *SPARQL SELECT* query (line15 of Alg.1).

templates, like the property assertions $R(X, Y)$, do not allow the evaluation via dedicated reasoning tasks and require satisfiability check for each possible instantiation. Consequently, within each connected component, the evaluation is performed considering first all atoms of the type $C(X)$ for the purpose of restricting as much as possible the compatible mappings for the metavariables, then the atoms $R(X, Y)$ (X or Y may possibly be an individual constant) are considered.

Note that, unlike the previous engines, MEE does not need to initialize the inference model on each step of the repeat-until loop. In fact, the queries are evaluated through a number of calls to the ELK reasoner, that make it possible to exploit the characteristics of incremental classification.

4 Experimental Settings

In this section we present synthetic test cases which have been specifically designed to simulate the employment of SOVGen in a e-health scenario. In particular, each test case represents the encoding of sensitive data in a CDA-compliant electronic health record.[10]

According to the theoretical framework each test case comprises four different components: the ontology *KB* that contains confidential data to be protected; an ontology *MR* encoding the user meta knowledge with a set of metarules; a set S of secrets; a series of ontologies representing the user's object-level background knowledge *BK*.

KB generation. KB is generated as a set of assertions instantiating the PS ontology. PS encodes a patient summary clinical document following the HL7 Implementation Guide for CDA Rel.2 Level 3: Patient Summary. As it can be seen in Figure 1, PS currently provide a support for encoding information about (i) history of assumed medications; (ii) clinical problem list including diagnosis, diagnostic hypothesis and clinical findings; (iii) history of a family member disease; (iv) list of the procedures the patient has undergone; (v) list of relevant diagnostic tests and laboratory data. Note that, according to the CDA standards a disease in the PS ontology is represented by a ICD-9CM code, while pharmaceutical products and procedures are represented by a SNOMED CT codes. For example, <code code="64572001" codeSystemName="SNOMED CT"/> stands for an instance of the SNOMED CT concept Disease (SCT_64572001). The type of sections to be generated are randomly chosen among those mentioned above. A disease (resp. product, procedure, test) code to associate to the entries is chosen as a random leaf of the corresponding Disease (resp. Pharmaceutical/biologic product, Procedure by site, Measurement procedure, Imaging) concept of the SNOMED CT ontology. In case a disease code is needed, the ICD-9CM code corresponding to the SNOMED CT one is retrieved and the equivalence is added to a background knowledge ontology named EQIV-RL.

Metarule generation. The knowledge encoded in KB gives rise to several possible types of metarules. Bridge metarules associate a ICD-9CM/SNOMED CT code to the concept in the respective ontology. For instance,

$$\text{CD}(C),\ \text{dtpCode}(C, 64572001),\ \text{dtpCodeSystem}(C, \text{SNOMED-CT}) \Rightarrow \text{SCT\_64572001}(C)$$

[10] Clinical Document Architecture (CDA) is a standard for information exchange, based on the Health Level 7 Reference Information Model.

makes it possible to derive that a code instance C is in fact an instance of the Disease concept in SNOMED CT.

The second type of metarules concerns the pharmaceutical products. The presence of a drug in the history of medication use implies that the patient suffers (certainly or with a great probability) from a specific pathology or has undertaken a specific procedure. Consider the following example of metarule which says that the presence of a medicine with active ingredient Phenytoin (SCT_40556005) indicates that the patient suffers from some kind of Epilepsy (SCT_84757006):

$$\begin{aligned}&\text{Patient}(P),\ \text{SubstanceAdministration}(SA),\ \text{Consumable}(C),\ \text{hasConsumable}(SA, C),\\ &\text{ManufacturedProduct}(MP),\ \text{hasManufacturedProduct}(C, MP),\ \text{Material}(M),\\ &\text{hasManufacturedMaterial}(MP, M),\ \text{SCT\_40556005}(CD),\ \text{hasCode}(M, CD)\\ &\Rightarrow \exists\text{suffer.SCT\_84757006}(P)\end{aligned}$$

The third type of metarules concerns the problems section. In particular the presence of a diagnosis (resp. diagnostic hypothesis) indicates that the patient suffer (resp. possibly suffer) a certain pathology.

Other types of metarules apply to the family history – e.g. a patient could be subject to a family members' disease – and the procedures section. For instance, the metarule

$$\text{Patient}(P),\ \text{Procedure}(I),\ \text{SCT\_77465005}(C),\ \text{hasCode}(I, C) \Rightarrow \text{subject}(P, C)$$

allows to entail that the presence of an organ transplantation (SCT_77465005) in the procedure section indicates that the patient is subject to transplantation.

Note that the generation of MR is not completely random for a part of the metarules. In order to obtain a nontrivial reasoning, during the KB generation, together with the creation of a section' entry is also created one or more corresponding bridge metarules and a metarule corresponding to the section in question. A second part of metarules are constructed by randomly selecting appropriate SNOMED CT concepts as needed. The adoption of such approach guarantees that al least part of metarules are actually fired during the secure ontology view generation. Furthermore, observe that the there are actually two levels of metarules, the bridge metarules constitute a precondition for the activation of the others.

Secrets generation. The ontology S is randomly generated as a set of assertions of the types:

$$\exists\text{suffer}.X(p) \quad \exists\text{possiblySuffer}.X(p) \quad \exists\text{possibleSubject}.X(p) \quad \exists\text{subject}.Y(p)$$

where X (resp. Y) is chosen as a random subconcept of the Disease (resp. Procedure) concept of the SNOMED CT ontology.

Background knowledge. The background knowledge *BK* is approximated by means of the PS, SNOMED-CT and the previously mentioned EQIV-RL ontologies.

5 Performance Analysis

In this section we present a performance analysis of SOVGen. Scalability evaluations have been carried out on synthetic test cases as described in Section 4. The size of *KB* is given by the parameter *KB-size* as the number of assertions occurring in the ontology.

```
<clinicalDocument>
   <recordTarget>
      <patientRole>
         <patient> . . . </patient>
      </patientRole>
   </recordTarget>
   <structuredBody>
      <section> <code code='10160-0' codeSystemName='LOINC'/> <!-- HISTORY OF MEDICATION USE -->
         <entry> . . . </entry>
      </section>
      <section> <code code='11450-4' codeSystemName='LOINC'/> <!-- CLINICAL PROBLEM LIST -->
         <entry> . . . </entry>
      </section>
      <section> <code code='10157-6' codeSystemName="LOINC"/> <!-- FAMILY MEMBER DISEASES -->
         <entry> . . . </entry>
      </section>
      <section> <code code='47519-4' codeSystemName='LOINC'/> <!-- HISTORY OF PROCEDURES -->
         <entry> . . . </entry>
      </section>
      <section> <code code='30954-2' codeSystemName="LOINC"/> <!-- RELEVANT DIAGNOSTIC TESTS -->
         <entry> . . . </entry>
      </section>
   </structuredBody>
</clinicalDocument>
```

Fig. 1. HL7 CDA Rel.2 Patient Summary

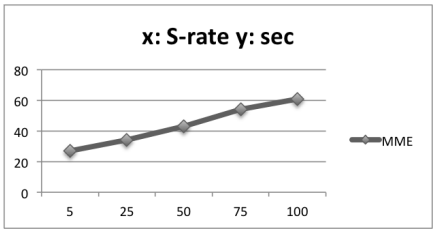
Then, the size of MR, *MR-rate*, is the ratio between the number of metarules and the number of assertions in *KB*. Finally, the size of S is determined by the parameter *S-rate* that specifies the ratio $|S|/|KB|$.

The experiments were performed on an Intel Core i7 2,5GHz laptop with 16GB and OS X 10.10.1, using Java 1.7 configured with 8GB RAM and 4GB stack space. Each reported value is the average execution time of five runs over five different ontologies. Note that given the amount of background knowledge (consider that SNOMED-CT describes about 300K concepts) the use of module extraction techniques improves the computation time of two–three orders of magnitude at a cost of about 30 sec of overhead.

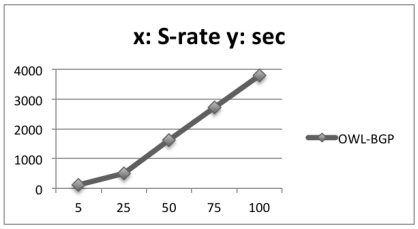
In Figure 2, the left (resp. right) column shows the experimental results obtained by using MEE (resp. OWL-BGP) to evaluate metarules – no result for Jena is reported as the execution time on all the test cases exceeded 1 hour time-out. Figures 2(a) and 2(b) report the execution time as the amount of secrets grows. Both *MR-rate* and *KB-size* are fixed, respectively to 10% and 200 assertions. Note that, MEE outperforms OWL-BGP of 1–2 orders of magnitude. Figures 2(c) and 2(d) show the impact of MR-rate when *KB-size* is fixed to 200 and *S-size* to 10%. Here, MEE runs about 10 times faster than OWL-BGP. Finally, Figures 2(e) and 2(f) illustrate the way the execution time changes as the the size of *KB* increases. Again MEE is 10^2 faster than OWL-BGP.

6 Conclusions

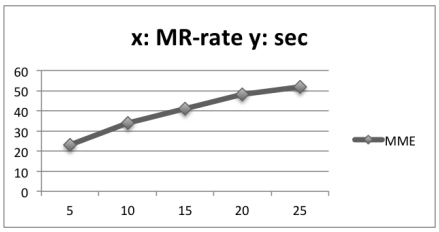
In [7] a novel confidentiality model has been introduced which adapts Controlled Query Evaluation to the context of Description Logics, and extends it by taking into account object-level and meta background knowledge. Here, we have presented SOVGen, a first implementation of this methodology that has been specialized to deal with a concrete



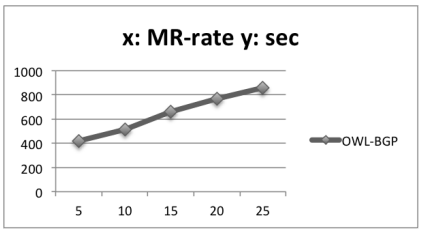
(a) KB-size=200, MR-rate=10%



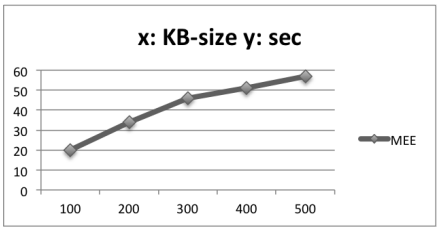
(b) KB-size=200, MR-rate=10%



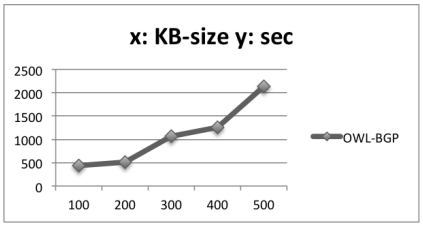
(c) KB-size=200, S-rate=10%



(d) KB-size=200, S-rate=10%



(e) MR-rate=10%, S-rate=25%



(f) MR-rate=10%, S-rate=25%

Fig. 2. Secure view construction time with MEE e OWL-BGP on variation of the parameters S-rate, MR-rate and KB-rate

e-health application. In order to maximize performance, we have compared different reasoning tools and designed several optimization techniques. Then, we assessed SOVGen experimentally by using realistic electronic health records that refer to SNOMED-CT concepts, and Horn rules to represent meta knowledge. In particular, we observed that module extraction techniques and a suitable, ad-hoc metarule evaluation engine – which intensively exploit ELK incremental reasoning – largely outperform general conjunctive query evaluation engines.

Considering that secure views are constructed off-line – so that no overhead is placed on user queries – performance analysis shows that SOVGen is close to meet practical use in this application scenario. In future work, we aim at improving the system with new optimizations, and extending it to general rules.

Bibliography

[1] F. Abel, J. L. D. Coi, N. Henze, A. W. Koesling, D. Krause, and D. Olmedilla. Enabling advanced and context-dependent access control in RDF stores. In K. Aberer et al., editor, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, volume 4825 of *LNCS*, pages 1–14. Springer, 2007.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In *International Semantic Web Conference*, pages 49–64, 2009.

[4] J. Biskup and P. A. Bonatti. Lying versus refusal for known potential secrets. *Data Knowl. Eng.*, 38(2):199–222, 2001.

[5] J. Biskup and P. A. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *Int. J. Inf. Sec.*, 3(1):14–27, 2004.

[6] J. Biskup and P. A. Bonatti. Controlled query evaluation for known policies by combining lying and refusal. *Ann. Math. Artif. Intell.*, 40(1-2):37–62, 2004.

[7] P. A. Bonatti and L. Sauro. A confidentiality model for ontologies. In H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2013.

[8] P. A. Bonatti, L. Sauro, and I. Petrova. A mechanism for ontology confidentiality. In L. Giordano, V. Gliozzi, and G. L. Pozzato, editors, *Proceedings of the 29th Italian Conference on Computational Logic, Torino, Italy, June 16-18, 2014.*, volume 1195 of *CEUR Workshop Proceedings*, pages 147–161. CEUR-WS.org, 2014.

[9] Eldora, M. Knechtel, and R. Peñaloza. Correcting access restrictions to a consequence more flexibly. In R. Rosati, S. Rudolph, and M. Zakharyaschev, editors, *Description Logics*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[10] G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling access to RDF graphs. In A.-J. Berre, A. Gómez-Pérez, K. Tutschku, and D. Fensel, editors, *FIS*, volume 6369 of *Lecture Notes in Computer Science*, pages 107–117. Springer, 2010.

[11] B. C. Grau, E. Kharlamov, E. V. Kostylev, and D. Zheleznyakov. Controlled query evaluation over OWL 2 RL ontologies. In H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of *Lecture Notes in Computer Science*, pages 49–65. Springer, 2013.

[12] B. C. Grau, E. Kharlamov, E. V. Kostylev, and D. Zheleznyakov. Controlled query evaluation over lightweight ontologies. In M. Bienvenu, M. Ortiz, R. Rosati, and M. Simkus, editors, *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, volume 1193 of *CEUR Workshop Proceedings*, pages 141–152. CEUR-WS.org, 2014.

[13] Y. Kazakov, M. Krötzsch, and F. Simancik. The incredible ELK - from polynomial procedures to efficient reasoning with el ontologies. *J. Autom. Reasoning*, 53(1):1–61, 2014.

[14] M. Knechtel and H. Stuckenschmidt. Query-based access control for ontologies. In P. Hitzler and T. Lukasiewicz, editors, *RR*, volume 6333 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2010.

[15] I. Kollia and B. Glimm. Optimizing SPARQL query answering over OWL ontologies. *CoRR*, abs/1402.0576, 2014.

[16] F. Martin-Recuerda and D. Walther. Axiom dependency hypergraphs for fast modularisation and atomic decomposition. In M. Bienvenu, M. Ortiz, R. Rosati, and M. Simkus, editors, *Proceedings of the 27th International Workshop on Description Logics (DL'14)*, volume 1193 of *CEUR Workshop Proceedings*, pages 299–310, 2014.

[17] U. Sattler, T. Schneider, and M. Zakharyaschev. Which kind of module should I extract? In B. Cuenca Grau et al., editor, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

Conjunctive Query Answering with Finitely Many Truth Degrees*

Stefan Borgwardt[1], Theofilos Mailis[2],
Rafael Peñaloza[3]**, and Anni-Yasmin Turhan[4]

[1] Chair for Automata Theory, Theoretical Computer Science, TU Dresden, Germany
[2] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
[3] KRDB Research Centre, Free University of Bozen-Bolzano, Italy
[4] Department of Computer Science, University of Oxford, UK

1 Introduction

Fuzzy description logics (FDLs) have arisen as suitable formalisms for representing and reasoning with the vague or imprecise knowledge that is intrinsic to many application domains. They extend classical description logics by allowing additional truth degrees that lie between the classical "true" and "false" values. These truth degrees typically belong to a subset of the interval $[0, 1]$.

For example, in a cloud computing environment, one might be interested in modeling the notion of an *overused* component. This is a typical example of an imprecise concept, since it is impossible to give a precise point where a component starts being overused. Instead, in FDLs, all components are assigned the degree to which they are being overused, where a higher degree implies a more extensive usage. For example, an idle component is overused with degree 0, while a component running at half its capacity might be overused to degree 0.8. The axioms

$$\langle \mathsf{Overused}(\mathsf{cpuA}) \geq 0.8 \rangle,$$
$$\langle \mathsf{Server} \sqcap \exists \mathsf{hasPart}.\mathsf{Overused} \sqsubseteq \mathsf{ServerWithLimitedResources} \geq 0.9 \rangle$$

express that object cpuA is overused to a degree of at least 0.8, and that every server that has an overused part is a server with limited resources with a degree of at least 0.9, respectively. The different concept constructors are interpreted by a t-norm and its associated operators [13]. One important t-norm is the Łukasiewicz t-norm, which is defined by $x \otimes y := \max\{x + y - 1, 0\}$.

Since dealing with infinitely many truth degrees easily leads to undecidability of reasoning [1, 7, 12], we focus on finitely valued FDLs, where the degrees are

* This work was partially supported by DFG under grant BA 1122/17-1 'FuzzyDL' (S. Borgwardt), the European project Optique (T. Mailis), the Cluster of Excellence 'cfAED' (R. Peñaloza), and EPSRC grant EP/J0083-46/1 'PrOQAW: Probabilistic Ontological Query Answering on the Web' (A.-Y. Turhan).

** The work was developed while the author was still affiliated with TU Dresden and the Center for Advancing Electronics Dresden, Germany.

ordered in a finite chain. In this case, standard reasoning in expressive FDLs has been shown to be decidable, and in the same complexity class, as reasoning in their classical counterparts [9–11]. One proposed method for reasoning in finitely valued FDLs is based on *crispification*. The idea of this method is to transform the fuzzy ontology into a classical ontology that preserves all the information about the truth degrees expressed in the original ontology. This is achieved through new concept and role names like $\mathsf{Fast}_{\geq 0.8}$ that intuitively contain all the elements that belong to Fast to a degree of at least 0.8. In this way, one can reduce reasoning in fuzzy DLs to reasoning in classical DLs, for which highly optimized reasoners exist. However, this approach only works for DLs that include at least the expressivity of $\mathcal{ALCH}$.

2 Types of Fuzzy Queries

A reasoning problem extensively studied for DLs over the last years is (conjunctive) query answering, together with the associated query entailment problem. Briefly, a conjunctive query q is a finite set of concept and role atoms, which intuitively are ABox assertions that might contain variables in place of individuals. An ontology $\mathcal{O}$ *entails* the query q if every model $\mathcal{I}$ of $\mathcal{O}$ has a match for q; that is, if all the variables in q can be mapped to elements of the domain of $\mathcal{I}$ in a way that all the atoms are satisfied. In FDLs, the matches of an atom need not be absolute, but might also hold with a truth degree between 0 and 1.

The existence of intermediate truth degrees gives rise to two different notions of conjunctive queries that can be entailed by a fuzzy ontology. The first one, called *threshold conjunctive query*, extends the notion of an atom to express additionally the least degree to which the atom must be satisfied in each model. Thus, for example, we can ask whether $\mathsf{server1}$ is fast (to degree 0.8) and has an overused (to degree 0.6) component through the threshold query

$$\{\mathsf{Fast}(\mathsf{server1}) \geq 0.8, \quad \mathsf{hasPart}(\mathsf{server1}, x) \geq 1, \quad \mathsf{Overused}(x) \geq 0.6\}. \tag{1}$$

Such a query is entailed by $\mathcal{O}$ if every model of $\mathcal{O}$ has a match with at least the given degrees. Notice that the result of a threshold query entailment check is either "yes" (if the query is entailed by $\mathcal{O}$) or "no." There are no intermediate degrees associated with these answers.

The second type of query, called *fuzzy conjunctive query*, asks for the *best entailment degree*; i.e., the largest possible degree d such that every model of the ontology has a match to degree at least d. For example, using the fuzzy conjunctive query

$$\{\mathsf{Fast}(\mathsf{server1}), \quad \mathsf{hasPart}(\mathsf{server1}, x), \quad \mathsf{Overused}(x)\}, \tag{2}$$

we can find the best degree to which $\mathsf{server1}$ is fast and has an overused component, where the conjunction between the atoms is interpreted using a t-norm. In the case of fuzzy conjunctive queries, there is only one degree that is global for the whole match of the query. Thus, it is possible that the threshold query above is not entailed (i.e., answers "no") while this fuzzy conjunctive query returns a positive degree.

3 Results

We propose a query answering procedure based on the crispification approach. In addition to crispifying the ontology, we also translate a threshold query into a classical conjunctive query that preserves the semantics w.r.t. the crispified ontology. For example, the threshold query (1) is crispified into the classical conjunctive query

$$\{\mathsf{Fast}_{\geq 0.8}(\mathsf{server1}), \quad \mathsf{hasPart}_{\geq 1}(\mathsf{server1}, x), \quad \mathsf{Overused}_{\geq 0.6}(x)\}.$$

Recall that $\mathsf{Fast}_{\geq 0.8}$ is a classical concept name from the crispified ontology. Thus, deciding entailment of this conjunctive query suffices for deciding entailment of the original threshold query.

A similar translation is used for fuzzy conjunctive queries, except that the result is a union of conjunctive queries, where each conjunctive query considers a certain combination of individual degrees whose combination leads to the entailment of the fuzzy query. For example, to decide whether the fuzzy conjunctive query (2) is entailed to a degree of at least 0.8, say in the presence of the degree set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$, one has to consider a union of several CQs such as

$$\{\mathsf{Fast}_{\geq 0.8}(\mathsf{server1}), \quad \mathsf{hasPart}_{\geq 1}(\mathsf{server1}, x), \quad \mathsf{Overused}_{\geq 1}(x)\} \text{ and}$$
$$\{\mathsf{Fast}_{\geq 1}(\mathsf{server1}), \quad \mathsf{hasPart}_{\geq 1}(\mathsf{server1}, x), \quad \mathsf{Overused}_{\geq 0.8}(x)\},$$

where the t-norm of the individual degrees is equal to 0.8. After the translation, one can use any existing query answering system for classical DLs.

While studying the crispification approach for expressive FDLs, we encountered two issues. First, we noticed that some of the previous crispification approaches, such as those in [4,5], do not treat number restrictions correctly when the Łukasiewicz t-norm is used. Second, the previously known crispifications (see also [2, 6]) produce an exponential blow-up, which makes almost any instance of the problem infeasible. We solved the second issue by introducing a linear normalization step that ensures a polynomial bound on the size of the crispification. Essentially, the normalization process introduces abbreviations that avoid copying complex concepts during the crispification step.

Using this normalization step, we are able to prove tight complexity bounds for answering threshold conjunctive queries; the complexity is always the same as for classical conjunctive query answering (in DLs more expressive than $\mathcal{ALCH}$ that do not have number restrictions). Unfortunately, the translation of fuzzy conjunctive queries causes an exponential blow-up, which is avoided when the simple Gödel t-norm $x \otimes y := \min\{x, y\}$ is used. Moreover, the *data complexity* of classical query answering in DLs is not affected when considering finitely valued semantics, as the reduction of the ABox (the data) is linear. Finally, our method for fuzzy query answering can be applied to any crispification approach, i.e. also in the cases that correctly handle number restrictions [3].

More details can be found in [8], which has been submitted to a journal. A preliminary version of these results appeared in [14].

References

1. Baader, F., Penaloza, R.: Are fuzzy description logics with general concept inclusion axioms decidable? In: Fuzzy Systems (FUZZ), 2011 IEEE International Conference on. pp. 1735–1742. IEEE (2011)
2. Bobillo, F., Delgado, M., Gómez-Romero, J.: Crisp representations and reasoning for fuzzy ontologies. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 17(4), 501–530 (2009)
3. Bobillo, F., Delgado, M., Gómez-Romero, J., Straccia, U.: Fuzzy Description Logics under Gödel semantics. International Journal of Approximate Reasoning 50(3), 494–514 (2009)
4. Bobillo, F., Delgado, M., Gómez-Romero, J., Straccia, U.: Joining Gödel and Zadeh fuzzy logics in fuzzy description logics. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 20(4), 475–508 (2012)
5. Bobillo, F., Straccia, U.: Reasoning with the finitely many-valued Łukasiewicz fuzzy Description Logic $\mathcal{SROIQ}$. Information Sciences 181(4), 758–778 (2011)
6. Bobillo, F., Straccia, U.: Finite fuzzy Description Logics and crisp representations. In: Uncertainty Reasoning for the Semantic Web II, pp. 99–118. Springer (2013)
7. Borgwardt, S., Distel, F., Peñaloza, R.: The limits of decidability in fuzzy description logics with general concept inclusions. Artificial Intelligence 218, 23–55 (2015)
8. Borgwardt, S., Mailis, T., Peñaloza, R., Turhan, A.Y.: Answering fuzzy conjunctive queries over finitely valued fuzzy ontologies (2015), under submission to a journal.
9. Borgwardt, S., Peñaloza, R.: The complexity of lattice-based fuzzy description logics. Journal on Data Semantics 2(1), 1–19 (2013)
10. Borgwardt, S., Peñaloza, R.: Consistency reasoning in lattice-based fuzzy description logics. International Journal of Approximate Reasoning 55(9), 1917–1938 (2014)
11. Bou, F., Cerami, M., Esteva, F.: Finite-valued Lukasiewicz modal logic is PSPACE-complete. In: Walsh, T. (ed.) Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 774–779. AAAI Press (2011)
12. Cerami, M., Straccia, U.: On the (un)decidability of fuzzy Description Logics under Łukasiewicz t-norm. Information Sciences 227, 1–21 (2013)
13. Hájek, P.: Metamathematics of Fuzzy Logic (Trends in Logic). Springer-Verlag (2001)
14. Mailis, T., Peñaloza, R., Turhan, A.Y.: Conjunctive query answering in finitely-valued fuzzy description logics. In: Kontchakov, R., Mugnier, M.L. (eds.) Proceedings of the 8th International Conference on Web Reasoning and Rule Systems (RR 2014). vol. 8741, pp. 124–139. Springer (2014)

Query Answering in Bayesian Description Logics

İsmail İlkan Ceylan*

Theoretical Computer Science, TU Dresden, Germany
ceylan@tcs.inf.tu-dresden.de

Abstract. The Bayesian Description Logic (BDL) $\mathcal{BEL}$ is a probabilistic DL, which extends the lightweight DL $\mathcal{EL}$ by defining a joint probability distribution over $\mathcal{EL}$ axioms with the help of a Bayesian network (BN). In the recent work, extensions of standard logical reasoning tasks in $\mathcal{BEL}$ are shown to be reducible to inferences in BNs.
This work concentrates on a more general reasoning task, namely on conjunctive query answering in $\mathcal{BEL}$ where every query is associated to a probability leading to different reasoning problems. In particular, we study the probabilistic query entailment, top-k answers, and top-k contexts as reasoning problems. Our complexity analysis suggests that all of these problems are tractable under certain assumptions.

1 Introduction

Description Logics (DLs) [3], as a successful family of knowledge representation (KR) formalisms, have been employed in various application domains such as conceptual modeling, databases, bio-medical ontologies, natural language processing, configuration, and the semantic web[1]. Arguably, all these domains, as is real world, are subject to imprecision; may it be an assertion about an individual or a terminological statement, it often comes along with a degree of uncertainty.

The fact that classical DLs had severe limitations in representing and reasoning under uncertainty led to a body of work [20] tailored towards this goal. Several extensions to DLs have been proposed with different characteristics in terms of their logical expressivity, their semantics, and their independence assumptions.

BDLs [6] have been proposed as a means of representing the uncertainty over DL axioms that are being asserted. In BDLs, every axiom is associated with a probability, which is encoded with the help of a BN. This family of logics provides a compact and easy way of encoding probabilities over DL axioms. Two important features of BDLs are that they do not force any independence assumptions, and they are based on the so-called multiple world semantics.

The focus of this work is the DL $\mathcal{BEL}$ [7], a Bayesian extension of the lightweight DL $\mathcal{EL}$ [2] for which several probabilistic reasoning tasks have been

* Supported by DFG within the Research Training Group "RoSI" (GRK 1907).
[1] http://www.w3.org/TR/owl2-overview/

Table 1: Syntax and Semantics of $\mathcal{EL}$

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Exist. Rest.	$\exists r.C$	$\{d \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in r^{\mathcal{I}}, e \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

studied such as the probabilistic entailment, or finding most likely context (sub-ontology) for an entailment. In fact, tight complexity bounds have been obtained for these problems [8].

Nevertheless, problems related to query answering, and in particular conjunctive query (CQ) answering, has not been studied in the context of BDLs, so far. In this paper, we close this gap and focus on i) probabilistic query entailment: "What is the probability of a query to be entailed?" ii) probabilistic query answering: "What are the *top-k answers* to a query?" and finally iii) the most likely context: "What are the *top-k contexts* that entail a query?"

Consequently, we argue that these problems generalize the reasoning problems that have been considered so far. Unsurprisingly, reasoning in $\mathcal{BEL}$ is intractible as is CQ answering in $\mathcal{EL}$ and inference in BNs. Further analysis shows that tractability can be regained by fixing the BN and the query.

2 Conjunctive Query Answering in $\mathcal{EL}$

We briefly review the DL $\mathcal{EL}$ [5] and query answering in $\mathcal{EL}$, which constitute the basis of this paper. Formally, let $\mathsf{N_I}$, $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint sets of *individual-*, *concept-* and *role-names*, respectively. $\mathcal{EL}$ *concept language* is defined by the grammar rule $C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

The *semantics* of $\mathcal{EL}$ is given by an *interpretation*: that is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps every individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; every concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to $\mathcal{EL}$ concepts as shown in the upper part of Table 1.

The domain knowledge is encoded through a set of axioms, which restrict the interpretation domain of the concepts. A *TBox* $\mathcal{T}$ is a finite set of *general concept inclusions (GCIs)* of the form $C \sqsubseteq D$, where C, D are concepts. An *ABox* is a finite set of *concept assertions* $C(a)$ and *role assertions* $r(a,b)$, where $a, b \in \mathsf{N_I}$, C is a concept and $r \in \mathsf{N_R}$. A *knowledge base* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. We use the term *axiom* as a general expression for GCIs and assertions.

The interpretation $\mathcal{I}$ *satisfies* an axiom λ iff it satisfies the conditions on the lower part of Table 1. It is a *model* of the TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$ and a *model* of the ABox $\mathcal{A}$ if it satisfies all the assertions in $\mathcal{A}$. An interpretation is a *model* of the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ iff it is a model of both $\mathcal{T}$ and $\mathcal{A}$. For the rest of this paper we will denote as $\mathsf{N_I}(\mathcal{A})$ the set of all individual names that appear in the ABox $\mathcal{A}$.

CQA is an important reasoning task for DLs that has been investigated in the context of $\mathcal{EL}$. Let NV be a set of *variables* disjoint from $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$. An *atom* is an expression of the form $A(\chi)$ or $r(\chi, \psi)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $\chi, \psi \in \mathsf{N_I} \cup \mathsf{NV}$. A *conjunctive query* (CQ) q is a non-empty set of atoms associated to a set $\mathsf{DV}(\mathsf{q}) \subseteq \mathsf{NV}$ of *distinguished variables*. If $\mathsf{DV}(\mathsf{q}) = \emptyset$, then q is called a *Boolean CQ*. A special case of a CQ is an *instance query (IQ)*, which consists of only one atom $A(\chi)$ with $A \in \mathsf{N_C}$.

Let q be a Boolean CQ and $\mathsf{IV}(\mathsf{q})$ be the set of all individual names and variables appearing in q. The interpretation $\mathcal{I}$ *satisfies* q if there exists a function $\pi : \mathsf{IV}(\mathsf{q}) \to \Delta^{\mathcal{I}}$ such that (i) $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{N_I} \cap \mathsf{IV}(\mathsf{q})$, (ii) $\pi(\chi) \in A^{\mathcal{I}}$ for all $A(\chi) \in \mathsf{q}$, and (iii) $(\pi(\chi), \pi(\psi)) \in r^{\mathcal{I}}$ for all $r(\chi, \psi) \in \mathsf{q}$. In this case, we call π a *match* for $\mathcal{I}$ and q. The ontology $\mathcal{O}$ *entails* q ($\mathcal{O} \models \mathsf{q}$) iff every model of $\mathcal{O}$ satisfies q. For an arbitrary CQ q, a function $\mathfrak{a} : \mathsf{DV}(\mathsf{q}) \to \mathsf{N_I}(\mathcal{A})$ is an *answer* to q w.r.t. $\mathcal{O}$ iff $\mathcal{O}$ entails the Boolean CQ $\mathfrak{a}(\mathsf{q})$ obtained by replacing every distinguished variable $\chi \in \mathsf{DV}(\mathsf{q})$ with $\mathfrak{a}(\chi)$. *Conjunctive query answering* (CQA) is the task of finding all answers of a CQ, and query entailment is the problem of deciding whether an ontology entails a given Boolean CQ by replacing every distinguished variable $\chi \in \mathsf{DV}(\mathsf{q})$ with $\mathfrak{a}(\chi)$.

It is well known that query entailment in $\mathcal{EL}$ is polynomial w.r.t. data and KB complexity, but NP-complete w.r.t. combined complexity [23]. Notice that, $\mathcal{EL}$ does not enjoy the so-called full *first order rewritability* which has been considered as a key feature for CQA, since it allows one to reduce the problem to standard tasks in Relational Database Management Systems (RDMSs). Yet, CQA in $\mathcal{EL}$ can be successfully employed using a combined approach as described in [22].

3 The Bayesian Description Logic $\mathcal{BEL}$

The Bayesian DL $\mathcal{BEL}$ [7] has been introduced as a probabilistic extension of the light-weight DL $\mathcal{EL}$. In $\mathcal{BEL}$ probabilities are encoded through a *Bayesian network* (BN) [11]; that is, a pair $\mathcal{B} = (G, \Phi)$, where $G = (V, E)$ is a finite directed acyclic graph (DAG) whose nodes represent (boolean) random variables, and Φ contains, for every node $x \in V$, a conditional probability distribution $P_{\mathcal{B}}(x \mid \pi(x))$ of x given its parents $\pi(x)$. If V is the set of nodes in G, we say that $\mathcal{B}$ is a BN *over* V.

BNs are widely studied probabilistic graphical models where the underlying graph $G = (V, E)$ encodes a series of conditional independence assumptions between the random variables. Every variable $x \in V$ is known to be conditionally independent of its non-descendants given its parents. Thus, every BN $\mathcal{B}$ defines

a unique joint probability distribution (JPD) over V given by

$$P_{\mathcal{B}}(V) = \prod_{x \in V} P_{\mathcal{B}}(x \mid \pi(x)).$$

The concept language of $\mathcal{BEL}$ is the same as the $\mathcal{EL}$ concept language. The difference appears in encoding the domain knowledge, i.e. in forming axioms. $\mathcal{BEL}$ generalizes classical TBoxes (resp. ABoxes) by annotating the GCIs (resp. assertions) with a context defined by a set of literals belonging to a BN.

Formally, let $\mathsf{N_I}$ be a set of individual names and V a finite set of boolean variables. A *V-context* is a conjunction of literals over V. A *V-restricted general concept inclusion* (V-GCI) is an expression of the form $\langle C \sqsubseteq D : \kappa \rangle$ where C, D are $\mathcal{BEL}$ concepts and κ is a V-context. A *V-restricted assertion* (V-assertion) is an expression of the form $\langle C(a) : \kappa \rangle$, or $\langle r(a,b) : \kappa \rangle$ where $a, b \in \mathsf{N_I}$, C, D are $\mathcal{BEL}$ concepts and κ is a V-context. A *V-TBox (resp.V-ABox)* is a finite set of V-GCIs (resp.V-assertions). A $\mathcal{BEL}$ *knowledge base* (KB) is a tuple $\mathcal{K} = (\mathcal{B}, \mathcal{T}, \mathcal{A})$ where $\mathcal{B}$ is a BN over V, $\mathcal{T}$ is a V-TBox and $\mathcal{A}$ is a V-ABox.

We will sometimes speak of *contextual axioms* to address both V-GCIs and V-assertions. The intuition behind the contextual axioms is to enforce an axiom to hold within a given context, but not necessarily in others. The semantic of such axioms is realized with the so-called *contextual interpretations*, which differently from the classical interpretations also evaluate the context variables. Formally, given a finite set of Boolean variables V, $(\mathcal{I}, \mathcal{V}^{\mathcal{I}})$ is a *contextual interpretation* where $\mathcal{V}^{\mathcal{I}}$ is a *propositional interpretation* over V, and $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a classical $\mathcal{EL}$ interpretation. We will usually ignore the prefix and speak simply of e.g. a *KB*, a *TBox*, an *ABox*, or an *interpretation*.

The interpretation function $\cdot^{\mathcal{I}}$ is extended to arbitrary $\mathcal{BEL}$ concepts as in $\mathcal{EL}$, i.e. using the rules in Table 1. We say that the contextual interpretation $(\mathcal{I}, \mathcal{V}^{\mathcal{I}})$ is a *model* of an axiom $\langle \lambda : \kappa \rangle$ denoted as $(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) \models \langle \lambda : \kappa \rangle$, iff either (i) $\mathcal{V}^{\mathcal{I}} \not\models \kappa$, or (ii) $\mathcal{I} \models \lambda$. It is a *model* of the TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) iff it is a model of all the axioms in $\mathcal{T}$ (resp. $\mathcal{A}$).

A contextual interpretation $(\mathcal{I}, \mathcal{V}^{\mathcal{I}})$ needs to satisfy only the axioms asserted within a context κ for which it holds that $\mathcal{V}^{\mathcal{I}} \models \kappa$. Formally, let $\mathcal{K} = (\mathcal{B}, \mathcal{T}, \mathcal{A})$ be a $\mathcal{BEL}$ KB: Given a contextual interpretation $(\mathcal{I}, \mathcal{V}^{\mathcal{I}})$ where $\mathcal{V}^{\mathcal{I}} = \mathcal{W}$, we define the $\mathcal{EL}$ KB $\mathcal{K}_{\mathcal{W}} = (\mathcal{T}_{\mathcal{W}}, \mathcal{A}_{\mathcal{W}})$ that needs to be satisfied by $\mathcal{I}$ as:

$$\mathcal{T}_{\mathcal{W}} := \{C \sqsubseteq D \mid \langle C \sqsubseteq D : \varphi \rangle \in \mathcal{T},\ \mathcal{W} \models \varphi\},$$
$$\mathcal{A}_{\mathcal{W}} := \{C(a) \mid \langle C(a) : \varphi \rangle \in \mathcal{A}, \mathcal{W} \models \varphi\} \cup \{r(a,b) \mid \langle r(a,b) : \varphi \rangle \in \mathcal{A}, \mathcal{W} \models \varphi\}.$$

In $\mathcal{BEL}$, uncertainty is represented through a BN that describes a joint probability distribution over the context variables. Semantically, $\mathcal{BEL}$ is linked to this distribution with the so called *multiple world semantics*: A *probabilistic interpretation* defines a probability distribution over a set of (contextual) interpretations; this distribution is required to be consistent with the joint probability distribution provided by the BN. Formally, a *probabilistic interpretation* is a pair $\mathcal{P} = (\mathfrak{I}, P_{\mathfrak{I}})$, where $\mathfrak{I}$ is a set of contextual interpretations and $P_{\mathfrak{I}}$ is a probability distribution over $\mathfrak{I}$ such that $P_{\mathfrak{I}}(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) > 0$ only for finitely many interpretations

$(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) \in \mathfrak{I}$. $\mathcal{P}$ is a *model* of the TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) if every $(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) \in \mathfrak{I}$ is a model of $\mathcal{T}$ (resp. $\mathcal{A}$). $\mathcal{P}$ is *consistent* with the BN $\mathcal{B}$ if for every possible valuation $\mathcal{W}$ of the variables in V it holds that

$$\sum_{(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) \in \mathfrak{I},\ \mathcal{V}^{\mathcal{I}} = \mathcal{W}} P_{\mathfrak{I}}(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) = P_{\mathcal{B}}(\mathcal{W}).$$

The probabilistic interpretation $\mathcal{P}$ is a *model* of the KB $(\mathcal{B}, \mathcal{T}, \mathcal{A})$ iff it is a (probabilistic) model of $\mathcal{T}$, $\mathcal{A}$ and consistent with $\mathcal{B}$.

To provide a fine-grained analysis of the complexity of reasoning in $\mathcal{BEL}$, we use different measures for the size of the input. In *data complexity*, we measure only the size of the ABox, and consider the rest of the KB and the query fixed. For *ontology complexity* we use the size of the TBox and the ABox; in *network complexity* the relevant input is the BN, while the *combined complexity* considers the size of the whole input.

4 Probabilistic Query Entailment

Different reasoning tasks have been studied in the context of Bayesian DLs; perhaps the most prominent one being the *probabilistic entailment* [6]. Although, probabilistic entailment has been considered generally, its focus was on entailments of simple consequences, i.e. consequences of the form subsumption, instance checking etc., all of which are tasks that can be decided in time polynomial in $\mathcal{EL}$. Thus, the class of problems based on entailments of simple consequences has lead tight complexity bounds in $\mathcal{BEL}$ [8].

Here we generalize these results and study *probabilistic query entailment*. In this setting, we are not just interested in the entailment of a query q but also in the probability of such entailment.

Definition 1 (probabilistic query entailment). *Let $\mathcal{K} = (\mathcal{B}, \mathcal{T}, \mathcal{A})$ be a $\mathcal{BEL}$ KB over V and $\mathcal{P} = (\mathfrak{I}, P)$ a probabilistic interpretation. $\mathcal{P}$ defines a probability distribution $P_{\mathcal{P}}$ over all conjunctive queries q given by*

$$P_{\mathcal{P}}(\mathsf{q}) := \sum_{(\mathcal{I}, \mathcal{V}^{\mathcal{I}}) \in \mathfrak{I},\ \mathcal{I} \models \mathsf{q}} P(\mathcal{I}, \mathcal{V}^{\mathcal{I}}).$$

The probability of the query q *w.r.t.* $\mathcal{K}$ *is* $P_{\mathcal{K}}(\mathsf{q}) := \inf_{\mathcal{P} \models \mathcal{K}} P_{\mathcal{P}}(\mathsf{q})$. *A query q is entailed with probability $p \in (0, 1]$ iff $P_{\mathcal{K}}(\mathsf{q}) \geq p$.*

Recall that every valuation $\mathcal{W}$ defines an $\mathcal{EL}$ ontology that contains all the axioms that must be satisfied by any contextual interpretation using the valuation $\mathcal{W}$. Given a Boolean CQ q, we can build a probabilistic model $\mathcal{P}_{\mathsf{q}} = (\mathfrak{I}, P)$ of $\mathcal{K}$ such that for every valuation $\mathcal{W}$ there is exactly one contextual interpretation $\mathcal{I}_{\mathcal{W}} \in \mathfrak{I}$, and it satisfies that $\mathcal{I}_{\mathcal{W}} \models \mathsf{q}$ iff $\mathcal{K}_{\mathcal{W}} \models \mathsf{q}$. It is easy to see that every other model $\mathcal{P}$ of $\mathcal{K}$ is such that $P_{\mathcal{P}}(\mathsf{q}) \geq P_{\mathcal{P}_{\mathsf{q}}}(\mathsf{q})$, which yields the following theorem.

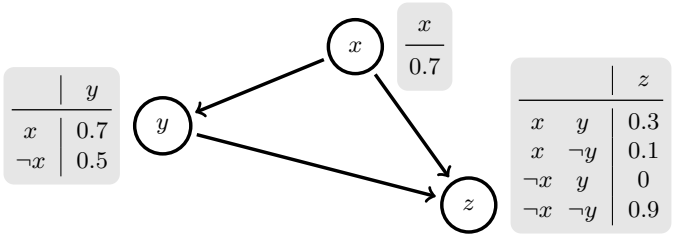


Fig. 1: The BN $\mathcal{B}_{\texttt{ABC}}$ over the variables $\{x, y, z\}$

Theorem 2. *For every $\mathcal{BEL}$ KB $\mathcal{K}$ and Boolean CQ $\mathtt{q}$ $P_{\mathcal{K}}(\mathtt{q}) = \sum_{\mathcal{K}_{\mathcal{W}} \models \mathtt{q}} P_{\mathcal{B}}(\mathcal{W})$.*

Given Theorem 2, one can compute the probability of any query by summing up the probabilities of the worlds that entail the query $\mathtt{q}$. We illustrate probabilistic query entailment with a simple example.

Example 3. Consider the $\mathcal{BEL}$ KB $\mathcal{K} = ((\mathcal{T}_{\texttt{ABC}}, \mathcal{A}_{\texttt{ABC}}), \mathcal{B}_{\texttt{ABC}})$ where

$$\mathcal{T}_{\texttt{ABC}} := \{ \langle \mathtt{A} \sqsubseteq \exists \mathtt{r}.\mathtt{B} : \{y\} \rangle , \langle \mathtt{B} \sqsubseteq \mathtt{C} : \{x\} \rangle \}$$
$$\mathcal{A}_{\texttt{ABC}} := \{ \langle \mathtt{A}(\mathtt{a}) : \{\mathtt{x}\} \rangle , \langle \mathtt{r}(\mathtt{a}, \mathtt{b}) : \{\mathtt{z}\} \rangle , \langle \mathtt{C}(\mathtt{b}) : \{x, z\} \rangle , \langle \mathtt{A}(\mathtt{c}) : \{\mathtt{y}\} \rangle \}$$

$\mathcal{B}_{\texttt{ABC}}$ is the BN given in Figure 1 and the Boolean CQ $\mathtt{q} = \{A(\chi), r(\chi, \psi), C(\psi)\}$. Clearly, $\mathcal{K}_{\mathcal{W}} \models \mathtt{q}$ only for worlds $\mathcal{W}$ such that $\mathcal{W} \models (x \wedge y) \vee (x \wedge z)$. Hence, we get $P_{\mathcal{K}}(\mathtt{q}) = P_{\mathcal{B}_{\texttt{ABC}}}((x \wedge y) \vee (x \wedge z)) = 0.411$.

Clearly the number of worlds might be exponential in $|V|$. In fact, this corresponds to exponentially many query entailment tests, which can be performed using polynomial space only.

Theorem 4. *Probabilistic query entailment is polynomial w.r.t. data and ontology complexity; and in* PSPACE *w.r.t. network and combined complexity.*

The bounds for network and combined complexity can be improved if we restrict the queries to instance queries only. It is then possible to use a novel structure, called the *proof structure* such as the one presented in [8]. The general idea is to reduce probabilistic reasoning in $\mathcal{BEL}$ knowledge bases to standard inferences in a BN. In essence, a proof structure compactly describes the class of contexts that entail the wanted consequence. Using this proof-structure, it is possible to construct a BN from which the probability of such consequence can be computed. Importantly, it has been shown that such reduction can be performed in polynomial time.

In a nutshell, a proof structure is a directed acyclic hyper-graph, in which every node represents an axiom. It is constructed in a bottom up manner with the help of a set of deduction rules. Starting from an initial set of axioms given by the KB, it adds new nodes for the axioms resulting from 1-step application of the deduction rules. Edges are used for denoting the axioms that have been used for the deduction. This process continues until the rules are saturated under

the set of axioms. This structure enables us to trace back all the causes for a consequence. Thus, once transformed into a BN, it represents all contexts for a consequence, the probability of which can then be computed via the BN. For the details, we refer to [8].

To provide a better complexity bound for probabilistic query entailment, we extend the proof structure to also handle the assertional knowledge, which was not present so far. Following a naïve approach it is possible to introduce a new set of deduction rules; instead, we make use of nominals to handle the assertions.

Briefly, the DL $\mathcal{ELO}$ extends $\mathcal{EL}$ with nominals; that is, it allows special types of concepts of the form $\{a\}$ with the semantics $\{a^{\mathcal{I}}\}$. It is well-known that in the presence of nominals, $\mathcal{EL}$ KBs can be represented without an ABox. Thus, for an $\mathcal{ELO}$ KB it is possible to benefit from the deduction rules presented in [19] to construct a proof structure. Using the approach in [8] with the new rules given in [19] over an $\mathcal{ELO}$ KB, we construct a proof structure for an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ that is guaranteed to contain the information of all possible causes for a consequence to follow from $\mathcal{K}$. Moreover, this hypergraph is acyclic and has polynomially many nodes, on the size of $\mathcal{K}$, by the properties of the rules and their applications.

A $\mathcal{BEL}$ KB can be transformed into a $\mathcal{BELO}$ KB in the obvious way. Let $\mathcal{K} = \{\mathcal{B}, \mathcal{T}, \mathcal{A}\}$ be a $\mathcal{BEL}$ KB, we construct the $\mathcal{BELO}$ KB $\mathcal{K}' = \{\mathcal{B}, \mathcal{T}'\}$ where

$$\begin{aligned}\mathcal{T}' \;=\; \mathcal{T} \;\cup\; \{\; \langle \{a\} \sqsubseteq C : \kappa \rangle \;\mid\; \langle C(a) : \kappa \rangle \in \mathcal{A} \;\} \\ \cup\; \{\; \langle \{a\} \sqsubseteq \exists . r\{b\} : \kappa \rangle \;\mid\; \langle r(a,b) : \kappa \rangle \in \mathcal{A} \;\}.\end{aligned}$$

Clearly, $\mathcal{K} \models c$ iff $\mathcal{K}' \models c$ for any consequence c. Hence, for any ABox assertion, it is possible to construct a proof structure of polynomial size. To check the probability of an instance query $C(\chi)$ we construct a BN using the proof structures of $C(a)$ where a is an individual appearing in the ABox. Observe that the number of proof structures is bound with the individuals available in the ABox and we obtain a polynomial construction w.r.t. the size of the input.

Together with the hardness of probabilistic entailment of simple consequences in $\mathcal{BEL}$ without ABoxes, we get the following result.

Lemma 5. *Probabilistic query entailment restricted to IQs is* PP*-complete w.r.t. the combined complexity.*

5 Probabilistic Query Answering

Query answering is the problem of finding mappings for a query, i.e. one is not just interested whether a query is entailed or not, but also with the witnesses of such entailment. Typically, data is assumed to be large and it is not always very feasible to return all answers to a query q to the user. One of the most important applications of query answering is returning the top-k answers to a given query q w.r.t. a measure. By this way users do not only get a feasible number of answers but also a fine grained view over the data. In the context of probabilities, we are interested in finding the answers that are most likely.

Let q be a query with the distinguished variables $\mathsf{DV}(\mathsf{q})$, and $\mathcal{K} = (\mathcal{B}, \mathcal{T}, \mathcal{A})$ a $\mathcal{BEL}$ KB. We denote by $\mathsf{Ind}(\mathcal{A})$ the set of all individual names appearing in $\mathcal{A}$. Recall that every function $\mathfrak{a} : \mathsf{DV}(\mathsf{q}) \to \mathsf{Ind}(\mathcal{A})$ defines a CQ obtained by replacing every $\chi \in \mathsf{DV}(\mathsf{q})$ in q with $\mathfrak{a}(\chi)$. Abusing the notation, we call this query $\mathfrak{a}(\mathsf{q})$. We call any function $\mathfrak{a} : \mathsf{DV}(\mathsf{q}) \to \mathsf{Ind}(\mathcal{A})$ an *answer* to q w.r.t. $\mathcal{K}$, and define its probability as $P_{\mathcal{K}}(\mathfrak{a}) := P_{\mathcal{K}}(\mathfrak{a}(\mathsf{q}))$. Since an answer defines a boolean CQ, all complexity results for CQs transfer immediately. Every answer to a query q, has a probability, which we use as a measure to distinguish the answers. We refine the set of answers w.r.t. their probabilities and return top answers only.

Definition 6 (top-k answer). *Let q be a query, $\mathcal{K}$ be a $\mathcal{BEL}$ KB, and $k \in \mathbb{N}$. A* top-k answer *to q w.r.t. $\mathcal{K}$ is a tuple $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k)$ of different answers to q w.r.t. $\mathcal{K}$ such that (i) for all $i, 1 \leq i < k$, $P_{\mathcal{K}}(\mathfrak{a}_i) \geq P_{\mathcal{K}}(\mathfrak{a}_{i+1})$, and (ii) for every other answer $\mathfrak{a}$, $P_{\mathcal{K}}(\mathfrak{a}_k) \geq P_{\mathcal{K}}(\mathfrak{a})$.*

In other words, a top-k answer is an ordered tuple of the k answers with the highest probability. We assume that k is a constant that is fixed *a priori*. Thus, it is not considered part of the input of the problem. Obviously, since different answers may have the same probability, top-k answers are not unique. Here we are only interested in finding one of them. Stating it as a decision problem, we want to verify whether a given tuple is a top-k answer.

Example 7. Consider the $\mathcal{BEL}$ KB $\mathcal{K} = ((\mathcal{T}_{\texttt{ABC}}, \mathcal{A}_{\texttt{ABC}}), \mathcal{B}_{\texttt{ABC}})$ provided in Example 3 and the query $\mathsf{q} = \{A(\chi)\}$ with $\chi \in \mathsf{DV}$. We are interested in identifying the top-1 answer to q w.r.t. $\mathcal{K}$. Notice that both $\mathfrak{a}_0 : \chi \mapsto a$ and $\mathfrak{a}_1 : \chi \mapsto c$ are answers to q with positive probability. Clearly, $\mathfrak{a}_0$ is the top-1 answer since $P_{\mathcal{K}}(\mathfrak{a}_0) > P_{\mathcal{K}}(\mathfrak{a}_1)$.

Assuming that the size of q and the BN $\mathcal{B}$ are fixed, there are polynomially many answers to q w.r.t. $\mathcal{K}$, and for each answer $\mathfrak{a}$, we can compute $P_{\mathcal{B}}(\mathfrak{a})$ performing polynomially many $\mathcal{EL}$ query entailment tests. Thus, it is possible to verify whether $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k)$ is a top-k answer in polynomial time w.r.t. ontology complexity.

If we consider the combined complexity, the problem can be decided as follows. For every answer to the query, we only keep track of those answers that are best by checking the probabilities $P_{\mathcal{B}}(\mathfrak{a})$ of the individual answers iteratively. Since the latter can be done in PSPACE, we obtain an upper bound.

Theorem 8. *Let $\mathfrak{A} = (\mathfrak{a}_1, \ldots, \mathfrak{a}_k)$ be a tuple of answers to q w.r.t. $\mathcal{K}$. Deciding whether $\mathfrak{A}$ is a top-k answer is polynomial w.r.t. data and ontology complexity, in* PSPACE *w.r.t. network complexity and combined complexity.*

We show a lower bound for this problem w.r.t. the combined complexity. by providing a reduction from the decision version of the *maximum a-posteriori* (D-MAP) problem for BNs [11]. Formally, given a BN $\mathcal{B}$ over V, a set $Q \subseteq V$, a context κ, and $p > 0$, the D-MAP problem consists of deciding whether there exists a valuation μ of the variables in Q such that $P_{\mathcal{B}}(\kappa \wedge \mu) > p$.

Consider an arbitrary but fixed instance of D-MAP described by the BN $\mathcal{B} = ((V, E), \Phi)$, the context κ, $Q \subseteq V$, and $p > 0$. We introduce a new Boolean random variable z not appearing in V. Using this variable, we construct a new DAG (V', E) with $V' = V \cup \{z\}$ and a new BN $\mathcal{B}' = ((V', E), \Phi')$, where $P_{\mathcal{B}'}(v \mid \pi(v)) = P_{\mathcal{B}}(v \mid \pi(x))$ for all $v \in V$, and $P_{\mathcal{B}'}(z) = p$. Consider the $\mathcal{BEL}$ KB $\mathcal{K} = (\mathcal{B}', \emptyset, \mathcal{A})$ where

$$\mathcal{A} := \{\langle A_x(a_x) : x\rangle, \langle A_x(b_x) : \neg x\rangle, \langle A_x(c) : z\rangle \mid x \in Q\} \cup \{\langle B(a) : \kappa\rangle, \langle B(c) : z\rangle\},$$

and query $\mathfrak{q} := \{A_x(\chi_x) \mid x \in Q\} \cup \{B(\chi)\}$, where all the variables are distinguished; i.e., $\mathsf{DV}(\mathfrak{q}) = \{\chi_x \mid x \in Q\} \cup \{\chi\}$. It is easy to see that the mapping $\mathfrak{a}_0 : \mathsf{DV}(\mathfrak{q}) \to \{c\}$ is an answer to this query and $P_{\mathcal{K}}(\mathfrak{a}_0) = p$. Moreover, any other answer that maps any variable to c will have the probability at most p, since it can only be entailed in contexts satisfying z. Suppose that there is an answer $\mathfrak{a}$ such that $P_{\mathcal{K}}(\mathfrak{a}) > p$. This answer must map every variable χ_x to either a_x or b_x and χ to a. Let $\mu_{\mathfrak{a}} := \bigwedge_{\mathfrak{a}(\chi_x)=a_x} x \wedge \bigwedge_{\mathfrak{a}(\chi_x)=b_x} \neg x$. By construction, $\mu_{\mathfrak{a}}$ is a valuation of the variables in Q, $P_{\mathcal{B}}(\kappa \wedge \mu_{\mathfrak{a}}) > p$, and $\mathfrak{a}(\mathfrak{q})$ is only entailed by valuations satisfying the context $\kappa \wedge \mu_{\mathfrak{a}}$. Overall this means that $\mathfrak{a}_0$ is *not* a top-1 answer iff there is a valuation μ of the variables in Q such that $P_{\mathcal{B}}(\kappa \wedge \mu) > p$.

Theorem 9. *Deciding whether a tuple $\mathfrak{A}$ is a top-k answer is co*$\mathrm{NP}^{\mathrm{PP}}$*-hard w.r.t. combined complexity.*

Notice that the proof uses a very simple query which is in fact acyclic. Thus, contrary to classical $\mathcal{EL}$ [4], restricting to acyclic queries does not suffice for reducing the complexity of reasoning. Clearly, if we consider IQs this hardness might not hold any more.

Obtaining most probable answers for a query is a crucial task for the domains where imprecise characterizations of knowledge is necessary. The next section is dedicated to another reasoning task that can be seen dual to top-k answers, namely top-k contexts.

6 Most Likely Contexts for a Query

Dually to finding the most likely answers to a query, we are also interested in finding the k most likely contexts that entail a given Boolean query $\mathfrak{q}$. More precisely, suppose that we have already observed that the query $\mathfrak{q}$ holds; then, we are interested in finding out which is the current context. As in the previous section, we do not consider one, but search for a fixed number of contexts that are the most likely to hold.

To define this reasoning task formally, we must generalize the notion of the ontology $\mathcal{K}_{\mathcal{W}}$ defined to consider arbitrary contexts κ, which we denote as $\mathcal{K}_\kappa$. For any contextual interpretation $(\mathcal{I}, \mathcal{V}^{\mathcal{I}})$ with $\mathcal{V}^{\mathcal{I}} \models \kappa$ it must hold that $\mathcal{I} \models \mathcal{K}_\kappa$. If $\mathcal{K}_\kappa$ entails the Boolean query $\mathfrak{q}$, then we say that $\mathfrak{q}$ holds in context κ. We are interested in finding out the most likely contexts in which a given query holds.

Definition 10 (top-k contexts). *Let* q *be a CQ,* $\mathcal{K}$ *a* $\mathcal{BEL}$ *KB, and* $k \in \mathbb{N}$. $\kappa_1, \ldots, \kappa_k$ *are* top-k contexts *for* q *w.r.t.* $\mathcal{K}$ *if* $\mathcal{K}_{\kappa_i}$ *entails* q *for all* $i, 1 \leq i \leq k$*;* $P_{\mathcal{B}}(\kappa_i) \geq P_{\mathcal{B}}(\kappa_{i+1})$ *for all* $i, 1 \leq i \leq k$*; and there is no other context* κ *such that* $\mathcal{K}_\kappa \models \mathsf{q}$ *and* $P_{\mathcal{B}}(\kappa) > P_{\mathcal{B}}(\kappa_k)$.

We illustrate top-k mlc with our continuing example. In this case, we are interested in finding out the 2 most likely context that entail the query.

Example 11. Consider the $\mathcal{BEL}$ KB $\mathcal{K} = ((\mathcal{T}_{\texttt{ABC}}, \mathcal{A}_{\texttt{ABC}}), \mathcal{B}_{\texttt{ABC}})$ and query q provided in Example 3. Clearly all contexts κ that entail q are such that $\kappa \models \{x, y\} \vee \{x, z\}$. The top-2 contexts are then $\langle \{x, y\}, \{x, z\} \rangle$ since $P_{\mathcal{B}_{\texttt{ABC}}}(\{x, y\}) > P_{\mathcal{B}_{\texttt{ABC}}}(\{x, z\})$.

The problem of finding one most likely context has been studied for simple queries. In those special cases, it was shown to be coNP^{PP}-complete problem w.r.t. combined complexity [8]. The coNP^{PP} upper bound holds also for top-k contexts w.r.t. combined complexity: if a tuple is not a top-k mlc, then guess a new context κ and show using a PP oracle that $\mathcal{K}_\kappa \models \mathsf{q}$ and $P_{\mathcal{B}}(\kappa) > P_{\mathcal{B}}(\kappa_k)$. If the BN is fixed, then the number of contexts is constant, and they can be ordered w.r.t. their complexity in constant time. The top-k mlc problem is then solved by applying a constant number of $\mathcal{EL}$ CQ entailment tests, yielding a polynomial upper bound w.r.t. ontology complexity. All these complexity results are summarized in the following theorem.

Theorem 12. *Deciding whether* $\kappa_1, \ldots, \kappa_k$ *are top-k mlc for* q *w.r.t. the KB* $\mathcal{K}$ *is polynomial w.r.t. data, and ontology complexity,* PP*-hard and in* NP^{PP} *w.r.t. network complexity, and* NP^{PP}*-complete w.r.t. combined complexity.*

Given the hardness of deciding top-k contexts, we consider a special case of this problem: Suppose now that all contexts are of a special form, i.e. they are valuations, we call this problem *top-k worlds*. In this case, we need to guess a world $\mathcal{W}$ and decide whether i) $\mathcal{K}_\mathcal{W} \models q$ and ii) $P_\mathcal{K}(\mathcal{W}) > P_\mathcal{K}(\mathcal{W}_k)$, where the former requires an NP oracle whereas the latter can be decided in polynomial time using tha standard chain rule of BNs.

Notice that, top-k contexts and top-k answers are dual to each other, but they do not necessarily overlap. Consider for instance the case, where all top-k answers to a query q are retrieved from the same context κ. In this case, top-k contexts for q will contain other contexts than κ with the assumption that $k > 1$. Deciding top-k contexts is particularly informative for cases where the diversity of knowledge is important.

We have discussed several reasoning problems in $\mathcal{BEL}$ w.r.t. CQs which we considered as natural problems that could arise in several domains. For a summary of the results, see Table 2.

7 Related Work

The literature on probabilistic extensions of DLs consists of various formalisms, each of which with different characteristics [20]. Despite the fact that probabilistic query answering has been studied widely in relational databases [15, 13, 9],

Table 2: $\mathcal{BEL}$ reasoning problems and their complexity

Problem	**data**	**ontology**	**network**	**combined**
probabilistic CQ entailment	P	P	PP-c	PP/PSPACE
probabilistic IQ entailment	P	P	PP-c	PP-c.
top-k answer	P	P	PP/PSPACE	coNP^{PP}/PSPACE
top-k contexts	P	P	$\text{PP}/\text{coNP}^{\text{PP}}$	coNP^{PP}/PSPACE
top-k worlds	P	P	coNP-c	coNP/Π_2^p

RDF graphs [16] and XML databases [1, 17], only few of the probabilistic DLs considered CQA as a reasoning task.

In the probabilistic extension of Datalog+/- [14] authors are interested in retrieving the answers that are above a threshold value that is set *a priori*. In contrast to $\mathcal{BEL}$, in probabilistic Datalog+/- the underlying semantics is based on Markov logic networks. The Prob-DL family [21] extends classical DLs with subjective probabilities, also known as Type II probabilities [18]. The main difference with our logic is that Prob-$\mathcal{EL}$ introduces probabilities as a concept constructor, whereas we allow only probabilities over axioms. More closely related to $\mathcal{BEL}$ is BDL-Lite [10]. As is in $\mathcal{BEL}$, BDL-Lite only allows probabilities over axioms and conditional dependencies are represented faithfully. However, as it has been pointed before [8], the authors use a closed world assumption, which easily leads to inconsistencies for the Bayesian extension of $\mathcal{EL}$.

8 Conclusions

We have studied probabilistic query entailment, top-k answers and top-k contexts as reasoning problems. Though not being complete, for each of these problems, we provided a complexity analysis. Moreover, we have shown that assuming that the given BN and query are relatively small, all problems become tractable. Removing this assumption immediately results in the loss of tractability, which is not surprising given the intractability results in BNs and CQA in $\mathcal{EL}$.

As a future work, we want to obtain tight bounds w.r.t. all measures provided. We have shown tight complexity bounds for the query entailment problem of IQs. Restricting our attention to IQs, other problems might also get easier under widely accepted assumptions of complexity theory. It should be reminded that this is unfortunately not the case for acyclic queries.

On the practical side, we will consider optimizing the reasoning mechanisms and we will implement a system for reasoning in $\mathcal{BEL}$ that will benefit both from techniques in DLs, such as module extraction, query processing and from techniques in reasoning with lifted BNs, mainly based on logic programming as in [12].

References

1. Abiteboul, S., Senellart, P.: Querying and updating probabilistic information in XML. In: Proc. of EDBT'06. LNCS, vol. 3896. Springer Verlag (2006)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI'05. Morgan Kaufmann Publishers (2005)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2007)
4. Bienvenu, M., Ortiz, M., Šimkus, M., Xiao, G.: Tractable queries for lightweight description logics. In: Proc. of IJCAI'13. AAAI Press (2013)
5. Brandt, S.: Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and—What Else? In: Proc. of ECAI'04. vol. 110. IOS Press (2004)
6. Ceylan, İ.İ., Peñaloza, R.: Bayesian Description Logics. In: Proc. of DL'14. CEUR Workshop Proceedings, vol. 1193. CEUR-WS (2014)
7. Ceylan, İ.İ., Peñaloza, R.: The Bayesian Description Logic $\mathcal{BEL}$. In: Proc. of IJCAR'14. LNCS, vol. 8562. Springer Verlag (2014)
8. Ceylan, İ.İ., Peñaloza, R.: Tight Complexity Bounds for Reasoning in the Description Logic $\mathcal{BEL}$. In: Proc. of JELIA'14. LNCS, vol. 8761. Springer Verlag (2014)
9. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB Journal 16(4) (2007)
10. D'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable Reasoning with Bayesian Description Logics. In: Proc. of SUM'08. LNCS, vol. 5291. Springer Verlag (2008)
11. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
12. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic prolog and its application in link discovery. In: Proc. of IJCAI'07. Morgan-Kaufmann Pub. (2007)
13. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM TOIS'97 15(1) (1997)
14. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in datalog +/- ontologies. Ann. Math. AI 69(1) (2013)
15. Grädel, E., Gurevich, Y., Hirsch, C.: The complexity of query reliability. In: Proc. ACM SIGACT-SIGMOD-SIGART'98 (1998)
16. Huang, H., Liu, C.: Query evaluation on probabilistic RDF databases. In: WISE09,. LNCS, vol. 5802. Springer Verlag (2009)
17. Hung, E., Getoor, L., Subrahmanian, V.S.: PXML: A probabilistic semistructured data model and algebra. In: Proc. ICDE'03 (2003)
18. Joseph, H.: An Analysis of First - Order Logics of Probability. In: Proc. of IJCAI'89. Morgan Kaufmann Publishers (1989)
19. Kazakov, Y., Krötzsch, M.R., Simančík, F.: Practical Reasoning with Nominals in the EL Family of Description Logics. In: Proc. of KR'12. AAAI Press (2012)
20. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 291–308 (Nov 2008)
21. Lutz, C., Schröder, L.: Probabilistic Description Logics for Subjective Uncertainty. In: Proc. of KR'10. AAAI Press (2010)
22. Lutz, C., Toman, D., Wolter, F.: Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ Using a Relational Database System. In: Proc. of IJCAI'09. AAAI (2009)
23. Rosati, R.: On conjunctive query answering in EL. In: Proc. of DL'07. CEUR Workshop Proceedings, vol. 250. CEUR-WS (2007)

Answering $\mathcal{EL}$ Queries in the Presence of Preferences

İsmail İlkan Ceylan[1⋆], Thomas Lukasiewicz[2], and Rafael Peñaloza[3⋆⋆]

[1] Theoretical Computer Science, TU Dresden, Germany
ceylan@tcs.inf.tu-dresden.de
[2] Department of Computer Science, University of Oxford, UK
Thomas.Lukasiewicz@cs.ox.ac.uk
[3] KRDB Research Centre, Free University of Bozen-Bolzano, Italy
rafael.penaloza@unibz.it

Conjunctive query (CQ) answering is an important reasoning task in description logics (DLs). Its goal is to retrieve the tuples of individuals that satisfy a conjunctive query; i.e., a finite set of atomic queries. These tuples are called answers. Clearly, a given CQ may have a considerable number of answers, specially if the set of individual names appearing in the ABox is large, as is the case for many existing DL ontologies. In order to manage all these answers in a structural manner, one can try to extend query answering with preference criteria, in such a way that the most preferred answers are returned first.

Possibilistic networks (PNs) have arisen as a way of representing conditional preferences over a finite set of events in a compact way [1]. The general idea is to provide a possibility degree to each conditional event which is proportional to the preference given to that event. We apply this idea to model the preferences of query answers indirectly, by modeling the preferences over the contexts that entail them. In a nutshell, we divide an $\mathcal{EL}$ knowledge base (KB) into *contexts*, and use a possibilistic network to describe the joint possibility distribution over these contexts. Our formalism is based on ideas previously presented for reasoning under probabilistic uncertainty described by a Bayesian network [3]. The preference of an answer to the query is the possibility degree of the best context that entails this answer. Dually, we also compute, given a query, the most preferred source; that is, the context with the highest degree that entails this query.

Similar to Bayesian networks [4], PNs are graphical models providing a compact representation of a discrete possibility distribution, through some independence assumptions [2]. A *possibility distribution* over a set Ω is a function $\mathsf{Pos} : \Omega \to [0, 1]$ that intuitively provides a degree of how possible is an event $\omega \in \Omega$ to happen. This function is extended to sets $\Gamma \subseteq \Omega$ by defining $\mathsf{Pos}(\Gamma) = \sup_{\omega \in \Gamma} \mathsf{Pos}(\omega)$. The *product conditional distribution* which is defined by the equation $\mathsf{Pos}(\Gamma \cap \Theta) = \mathsf{Pos}(\Gamma \mid \Theta) \cdot \mathsf{Pos}(\Theta)$.

Possibilistic networks decompose a possibility distribution into a product of conditional probability distributions that depend on the structure of a graph. A

⋆ Supported by DFG within the Research Training Group "RoSI" (GRK 1907).
⋆⋆ The work was developed while the author was still affiliated with TU Dresden and the author has been partially supported by the Cluster of Excellence "cfAED".

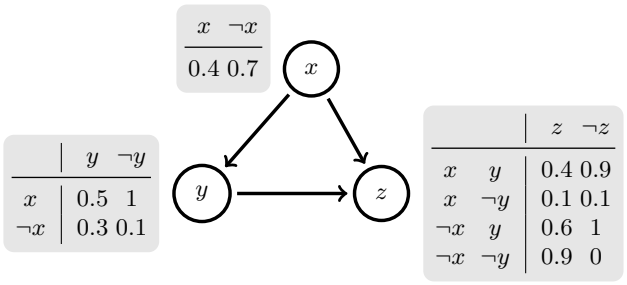


Fig. 1. A possibilistic network over $V_0 = \{x, y, z\}$

possibilistic network (PN) is a pair $\mathcal{P} = (G, \Phi)$, where $G = (V, E)$ is a DAG, and Φ contains a conditional possibility distribution $\mathsf{Pos}_{\mathcal{P}}(x \mid \mathsf{pa}(x))$ of every variable $x \in V$ given its parents $\mathsf{pa}(x)$ (see Figure 1). This PN defines the joint possibility distribution over the valuations of the variables in V

$$\mathsf{Pos}_{\mathcal{P}}(V) = \prod_{x \in V} \mathsf{Pos}_{\mathcal{P}}(x \mid \mathsf{pa}(x)).$$

Let V be a fixed but arbitrary finite set of propositional variables. A *V-context* is a propositional formula over V. A *V-GCI* is of the form $\langle C \sqsubseteq D : \varphi \rangle$ with C, D concepts and φ a V-context. A *V-TBox* is a finite set of V-GCIs. *V-assertions* are of the form $\langle C(a) : \varphi \rangle$ or $\langle r(a, b) : \varphi \rangle$ where $r \in \mathsf{N_C}$, $a, b \in \mathsf{N_I}$, C is a concept and φ is a V-context. A *V-ABox* is a finite set of V-assertions. A $\mathcal{PEL}$ KB is a tuple $\mathcal{K} = (\mathcal{P}, \mathcal{T}, \mathcal{A})$ where $\mathcal{P}$ is a PN over V, $\mathcal{T}$ is a V-TBox and $\mathcal{A}$ is a V-ABox.

The semantics of this logic is defined using multiple worlds. A *contextual interpretation* is a pair $(\mathcal{I}, \mathcal{W})$ where $\mathcal{I}$ is an $\mathcal{EL}$ interpretation and $\mathcal{W}$ is a valuation of the variables in V. $(\mathcal{I}, \mathcal{W})$ *satisfies* the axiom $\langle \lambda : \varphi \rangle$ ($(\mathcal{I}, \mathcal{W}) \models \langle \lambda : \varphi \rangle$), iff either (i) $\mathcal{W} \not\models \varphi$, or (ii) $\mathcal{I} \models \lambda$. It is a *model* of the $\mathcal{PEL}$ TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) iff it satisfies all the axioms in $\mathcal{T}$ (resp. $\mathcal{A}$). A *possibilistic interpretation* is a pair $\mathfrak{P} = (\mathfrak{J}, \mathsf{Pos})$, where $\mathfrak{J}$ is a finite set of contextual interpretations and Pos is a possibility distribution over $\mathfrak{I}$. $\mathfrak{P}$ is a *model* of the $\mathcal{PEL}$ TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) if every $(\mathcal{I}, \mathcal{W}) \in \mathfrak{J}$ is a model of $\mathcal{T}$ (resp. $\mathcal{A}$). $\mathfrak{P}$ is a *model* of the PN $\mathcal{P}$ if for every valuation $\mathcal{W}$,

$$\max_{(\mathcal{I}, \mathcal{W}) \in \mathfrak{J}} \mathsf{Pos}(\mathcal{I}, \mathcal{W}) = \mathsf{Pos}_{\mathcal{P}}(\mathcal{W}).$$

$\mathfrak{P}$ is a *model* of the $\mathcal{PEL}$ KB $\mathcal{K} = (\mathcal{P}, \mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{P}$.

Each possibilistic interpretation $\mathfrak{P} = (\mathfrak{J}, \mathsf{Pos})$ defines a possibility distribution $\mathsf{Pos}_{\mathfrak{P}}$ over all CQs given by $\mathsf{Pos}_{\mathfrak{P}}(\mathsf{q}) := \max_{(\mathcal{I}, \mathcal{W}) \in \mathfrak{J},\ \mathcal{I} \models \mathsf{q}} \{\mathsf{Pos}(\mathcal{I}, \mathcal{W})\}$. The *entailment degree* of q w.r.t. the $\mathcal{PEL}$ KB $\mathcal{K}$ is

$$\mathsf{Pos}_{\mathcal{K}}(\mathsf{q}) := \inf_{\mathfrak{P} \models \mathcal{K}} \{\mathsf{Pos}_{\mathfrak{P}}(\mathsf{q})\}.$$

These possibility distributions are extended to contexts in the obvious way, by setting $\mathsf{Pos}_{\mathfrak{P}}(\varphi) := \mathsf{Pos}_{\mathcal{P}}(\varphi) = \max_{\mathcal{W} \models \varphi} \mathsf{Pos}_{\mathcal{P}}(\mathcal{W})$. We can then define the con-

Table 1. $\mathcal{PEL}$ reasoning problems and their complexity

Problem	data	KB	network	combined
p-entailment	P	P	NP-c	NP-c
top-k answer	P	P	Δ_2^p-c	Δ_2^p-c
conditional top-k answer	P	P	Δ_2^p-c	Δ_2^p-c
k most preferred worlds	P	P	coNP-c	coNP-c

ditional possibilities of a query given a context, and of a context given a query, using the standard product rule. Formally,

$$\mathsf{Pos}_{\mathcal{K}}(\mathsf{q} \wedge \varphi) = \mathsf{Pos}_{\mathcal{K}}(\mathsf{q} \mid \varphi)\mathsf{Pos}_{\mathcal{K}}(\varphi),$$
$$\mathsf{Pos}_{\mathcal{K}}(\mathsf{q} \wedge \varphi) = \mathsf{Pos}_{\mathcal{K}}(\varphi \mid \mathsf{q})\mathsf{Pos}_{\mathcal{K}}(\mathsf{q}),$$

where

$$\mathsf{Pos}_{\mathcal{K}}(\mathsf{q} \wedge \varphi) = \inf_{(\mathfrak{I},\mathsf{Pos})\models\mathcal{K}} \left\{ \max_{\mathcal{I}\models\mathsf{q},\ \mathcal{W}\models\phi} \mathsf{Pos}(\mathcal{I},\mathcal{W})\} \right\}.$$

We consider three main reasoning problems in this setting; namely, deciding p-entailment, retrieving the top-k answers to a query, and the k most preferred worlds entailing a given query. We formally define these problems next. The problem of *p-entailment* refers to deciding whether $\mathsf{Pos}_{\mathcal{K}}(\mathsf{q}) \geq p$ for some given $p \in (0, 1]$. The *top-k answer* problem consists in deciding whether a tuple $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k)$ of different answers to q w.r.t. $\mathcal{K}$ is such that (i) for all $i, 1 \leq i < k$, $\mathsf{Pos}_{\mathcal{K}}(\mathfrak{a}_i) \geq \mathsf{Pos}_{\mathcal{K}}(\mathfrak{a}_{i+1})$, and (ii) for every other answer $\mathfrak{a}$, $\mathsf{Pos}_{\mathcal{K}}(\mathfrak{a}_k) \geq \mathsf{Pos}_{\mathcal{K}}(\mathfrak{a})$. This problem can be generalized to consider additional contextual evidence; that is, verify whether $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k)$ are the top-k answers to q *given* the context φ. Finally, the *k most preferred worlds* problem is the problem of deciding whether a tuple of k valuations of the variables V $(\mathcal{W}_1, \ldots, \mathcal{W}_k)$ is such that $\mathsf{Pos}_{\mathcal{K}}(\mathcal{W}_i \mid \mathsf{q}) \geq \mathsf{Pos}_{\mathcal{K}}(\mathcal{W}_{i+1} \mid \mathsf{q})$ holds for all $i, 1 \leq i < k$, and there exists no other valuation $\mathcal{W}$ such that $\mathsf{Pos}_{\mathcal{K}}(\mathcal{W} \mid \mathsf{q}) > \mathsf{Pos}_{\mathcal{K}}(\mathcal{W}_k \mid \mathsf{q})$.

The complexity of all these problems is summarized in Table 1, where network complexity refers to the complexity considering only the size of the PN as input, KB complexity considers the size of the ABox and TBox, while combined complexity considers the whole KB together with the PN and the query as the size of the input. As it can be seen, all the problems remain tractable w.r.t. data and KB complexity, but the complexity increases as soon as the PN or the query is considered part of the input. This corresponds to the behaviour exhibited by query answering in the classical $\mathcal{EL}$ [5]. The full details of these results can be found in the appendix.

Although all the complexity bounds are tight, they are all based on performing black-box query entailment tests on $\mathcal{EL}$ KBs. As future work we plan to adapt specific query answering techniques to produce effective algorithms that can be used in practice. We will also extend our framework to other kinds of standard and non-stardard reasoning tasks.

References

1. BenAmor, N., Dubois, D., Gouider, H., Prade, H.: Possibilistic Networks : A New Setting for Modeling Preferences. In: Proc. of SUM'14. LNCS, vol. 8720. Springer Verlag (2014)
2. Benferhat, S., Dubois, D., Garcia, L., Prade, H.: Possibilistic logic bases and possibilistic graphs. In: Proc. of UAI'99. Morgan-Kaufmann Publishers (1999)
3. Ceylan, İ.İ., Peñaloza, R.: The Bayesian Description Logic $\mathcal{BEL}$. In: Proc. of IJCAR'14. LNCS, vol. 8562. Springer Verlag (2014)
4. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
5. Rosati, R.: On conjunctive query answering in EL. In: Proc. of DL'07. CEUR Workshop Proceedings, vol. 250. CEUR-WS (2007)

Dynamic Bayesian Description Logics

İsmail İlkan Ceylan[1]⋆ and Rafael Peñaloza[2] ⋆⋆

[1] Theoretical Computer Science, TU Dresden, Germany
ceylan@tcs.inf.tu-dresden.de
[2] KRDB Research Centre, Free University of Bozen-Bolzano, Italy
rafael.penaloza@unibz.it

1 Introduction

It is well known that many artificial intelligence applications need to represent and reason with knowledge that is not fully certain. This has motivated the study of many knowledge representation formalisms that can effectively handle uncertainty, and in particular probabilistic description logics (DLs) [7–9]. Although these logics are encompassed under the same umbrella, they differ greatly in the way they interpret the probabilities (e.g. statistical vs. subjective), their probabilistic constructors (i.e., probabilistic axioms or probabilistic concepts and roles), their semantics, and even their probabilistic independence assumptions. A recent example of probabilistic DLs are the *Bayesian DLs*, which can express both logical and probabilistic dependencies between axioms [2–4].

One common feature among most of these probabilistic DLs is that they consider the uncertainty degree (i.e., the probability) of the different events to be fixed and static through time. However, this assumption is still too strong for many application scenarios. Consider for example a situation where a grid of sensors is collecting knowledge that is then fed into an ontology to reason about the situation of a large system. Since the sensors might perform an incorrect reading, this knowledge and the consequences derived from it can only be guaranteed to hold with some probability. However, the failure rate of a sensor is not static over time; as the sensor ages, its probability of failing increases. Moreover, the speed at which each sensor ages may also be influenced by other external factors like the weather at the place it is located, or the amount of use it is given.

We propose to extend the formalism of Bayesian DLs to *dynamic* Bayesian DLs, in which the probabilities of the axioms to hold are updated over discrete time steps following the principles of dynamic Bayesian networks. Using this principle, we can not only reason about the probabilistic entailments at every point in time, but also reason about future events given some evidence at different times. This work presents the first steps towards probabilistic reasoning about complex events over time.

⋆ Supported by DFG within the Research Training Group "RoSI" (GRK 1907).
⋆⋆ The work was developed while the author was still affiliated with TU Dresden and the author has been partially supported by the Cluster of Excellence "cfAED".

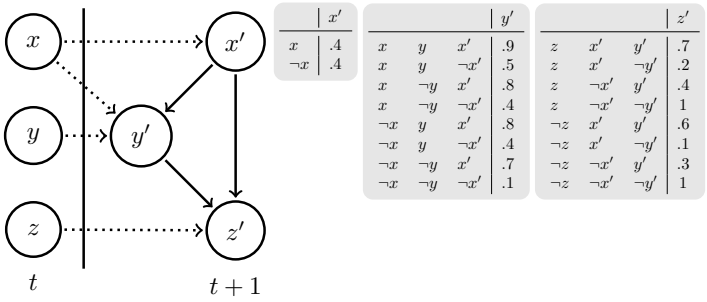


Fig. 1. The TBN $\mathcal{B}_{\rightarrow}$ over the variables $V = \{x, y, z\}$

2 Formalism

Following the ideas presented in [2], a dynamic Bayesian ontology is an ontology from an arbitrary (but fixed) DL $\mathcal{L}$, whose axioms are annotated with a context that expresses when they are considered to hold. The difference is that these contexts are now related via a dynamic Bayesian network.

A *Bayesian network* (BN) [5] is a pair $\mathcal{B} = (G, \Phi)$, where $G = (V, E)$ is a finite DAG and Φ contains, for every $x \in V$, a conditional probability distribution $P_{\mathcal{B}}(x \mid \pi(x))$ of x given its parents $\pi(x)$. Dynamic BNs (DBNs) [6,10] extend BNs to provide a compact representation of evolving joint probability distributions for a fixed set of random variables. Updates of the JPD are expressed through a two-slice BN, which expresses the probabilities at the next point in time, given the current context. A *two-slice BN* (TBN) is a pair (G, Φ), where $G = (V \cup V', E)$ is a DAG containing no edges between elements of V, $V' = \{x' \mid x \in V\}$, and Φ contains for every $x' \in V'$ a conditional probability distribution $P(x' \mid \pi(x'))$ of x' given its parents $\pi(x')$ (see Figure 1). A *dynamic Bayesian network* (DBN) is a pair $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_{\rightarrow})$ where $\mathcal{B}_1$ is a BN and $\mathcal{B}_{\rightarrow}$ is a TBN. Using the Markov property: the probability of the future state is independent from the past, given the present state, the DBN $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_{\rightarrow})$ defines, for every $t \geq 1$, a probability distribution $P_{\mathcal{B}}(V_t) = \prod_{i=2}^{t} \prod_{x \in V} P_{\mathcal{B}_{\rightarrow}}(x_i \mid \pi(x_i)_{i-1}) P_{\mathcal{B}_1}(V_1)$. This distribution is defined by unraveling the DBN starting from $\mathcal{B}_1$, using $\mathcal{B}_{\rightarrow}$ until t copies of V have been created. This produces a new BN $\mathcal{B}_{1:t}$ encoding the distribution over time of the variables. Figure 2 depicts $\mathcal{B}_{1:3}$ for the DBN $(\mathcal{B}_1, \mathcal{B}_{\rightarrow})$ where $\mathcal{B}_{\rightarrow}$ is the TBN from Figure 1. The conditional probability tables of each node given its parents (not depicted) are those of $\mathcal{B}_1$ for the nodes in V_1, and of $\mathcal{B}_{\rightarrow}$ for nodes in $V_2 \cup V_3$. Notice that $\mathcal{B}_{1:t}$ has t copies of each random variable in V.

A *V-context* is a consistent set of literals over V. A *V-axiom* is of the form $\langle \alpha : \kappa \rangle$ where $\alpha \in \mathfrak{A}$ is an axiom and κ is a V-context. A *V-ontology* is a finite set O of V-axioms, from the DL $\mathcal{L}$. A *$\mathcal{DBL}$ knowledge base* (KB) over V is a pair $\mathcal{K} = (\mathsf{O}, \mathcal{D})$ where $\mathcal{D}$ is a DBN over V and O is a V-ontology. The semantics of this logic is defined by producing for every point in time, a multiple-world

interpretation, where each world is associated to a probability that is compatible with the probability distribution defined by the DBN at that point in time.

3 Reasoning

The main reasoning task that we consider is to compute the probability of observing a consequence at different points in time. We consider three variants of this problem; namely, the probability of observing the consequence (i) *exactly* at time $t \geq 0$, (ii) *at most* at time t, or (iii) at *any* point in time. Combining methods from DBNs and context-based reasoning [1], we show that all these reasoning problems can be solved effectively.

The main idea is based on the unraveling of the DBN to time t. Using this unraveling, we readily know the probability of each context at every point in time between the present state and t. The logical element of the problem (i.e., knowing which contexts entail the consequence under consideration) is handled through the computation of a so-called *context formula*, which intuitively summarizes all the logical causes for the consequence to follow. Importantly, this context formula can be computed once for each consequence, and used for many different tests while reasoning. This unraveling and context formula can be used to solve the problems (i) and (ii) introduced above, with the help of a standard BN inference engine. Moreover, it is possible to add evidence observed at different points in time into the computation. This additional evidence does not yield any technical difficulties to our techniques, although may cause an increase in complexity, depending on the type and frequency of observations.

Clearly, the unraveling method cannot be used to compute the probability of *eventually* observing the consequence, as described by the problem (iii) above: one would potentially need to unravel the DBN to an infinite time, yielding a structure for which no effective reasoning methods exist. Instead, we identify some conditions under which this probability is easy to compute. Overall, this does not yield a full reasoning mechanism, but provides a good approximation in several meaningful cases.

As mentioned before, this work provides only the first steps towards a formalism for reasoning about events with evolving uncertainty. The following step is to be able to handle more complex time expressions and evidence.

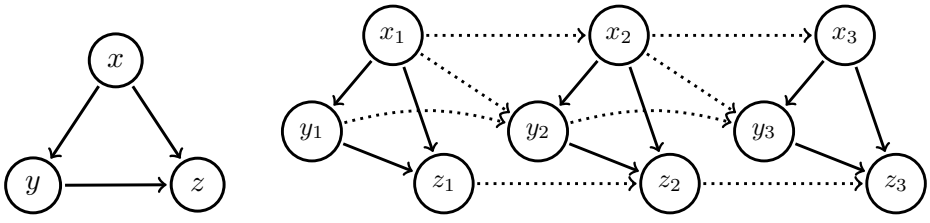


Fig. 2. $\mathcal{B}_1$ and the three step unraveling $\mathcal{B}_{1:3}$ of $(\mathcal{B}_1, \mathcal{B}_{\rightarrow})$

References

1. Baader, F., Knechtel, M., Peñaloza, R.: Context-Dependent Views to Axioms and Consequences of Semantic Web Ontologies. J. of Web Semantics 12–13, 22–40 (2012)
2. Ceylan, İ.İ., Peñaloza, R.: Bayesian Description Logics. In: Proc. of DL'14. CEUR Workshop Proceedings, vol. 1193. CEUR-WS (2014)
3. Ceylan, İ.İ., Peñaloza, R.: The Bayesian Description Logic $\mathcal{BEL}$. In: Proc. of IJCAR'14. LNCS, vol. 8562. Springer Verlag (2014)
4. Ceylan, İ.İ., Peñaloza, R.: Tight Complexity Bounds for Reasoning in the Description Logic $\mathcal{BEL}$. In: Proc. of JELIA'14. LNCS, vol. 8761. Springer Verlag (2014)
5. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
6. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Computational intelligence pp. 142–150 (1989), `http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8640.1989.tb00324.x/abstract`
7. Klinov, P., Parsia, B.: Representing sampling distributions in P-SROIQ. In: Proc. of URSW'11. Workshop Proceedings, vol. 778. CEUR (2011)
8. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 291–308 (Nov 2008)
9. Lutz, C., Schröder, L.: Probabilistic Description Logics for Subjective Uncertainty. In: Proc. of KR'10. AAAI Press (2010)
10. Murphy, K.: Dynamic bayesian networks: representation, inference and learning. Ph.D. thesis, University of California, Berkeley (2002)

ELASTIQ: Answering Similarity-threshold Instance Queries in $\mathcal{EL}$

Andreas Ecke[1⋆], Maximilian Pensel[1⋆⋆], and Anni-Yasmin Turhan[2⋆⋆]

[1] Institute for Theoretical Computer Science,
Technische Universität Dresden
[2] Department of Computer Science, University of Oxford, UK

Abstract. Recently an approach has been devised how to employ concept similarity measures (CSMs) for relaxing instance queries over $\mathcal{EL}$ ontologies in a controlled way. The approach relies on similarity measures between pointed interpretations to yield CSMs with certain properties. We report in this paper on ELASTIQ, which is a first implementation of this approach and propose initial optimizations for this novel inference. We also provide a first evaluation of ELASTIQ on the GeneOntology.

1 Introduction

Description Logics (DLs) are a family of knowledge representation whose formal semantics allow the definition of a variety of reasoning services. The most prominent ones are *subsumption* and *instance query answering*. However, many applications need to query the knowledge base in a more relaxed manner. For instance, in the application of service matching TBoxes are employed to describe types of services. Here, a user request for a service specifies several requirements for the desired service and can be represented by a complex concept. For such a concept the ABox that contains the individual services is searched for a service matching the specification by performing instance query answering. In cases where an exact match with the provided requirements is not possible, a 'feasible' alternative should be retrieved from the ABox.

To relax the notion of instance query answering one can simply employ fuzzy DLs and perform query answering on a fuzzy variant of the initial query concept. However, on the one hand reasoning in fuzzy DLs easily becomes undecidable, see [2–4] and on the other hand fuzzy concepts would always relax the initial concept in a uniform way and cannot consider user or request-specific preferences on which parts of the query are more important and should not be relaxed.

A reasoning service that allows for a given query concept the selective and gradual extension of the answer set of individuals is answering of relaxed instance queries, was proposed in [7] and further investigated in [8, 6, 9]. The selective, gradual relaxation of the answer sets returned to instance queries is achieved by the use of concept similarity measures. A *concept similarity measure* (CSM)

⋆ Supported by DFG Graduiertenkolleg 1763 (QuantLA).
⋆⋆ Partially supported by DFG in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

yields, for a pair of concepts, a value from the interval $[0, 1]$—indicating how similar the concepts are. To answer a relaxed instance query is to compute for a given concept C, a CSM $\sim$, and a threshold t between 0 and 1, a set of concepts such that each of these concepts are similar to C by a threshold of at least t, if measured by the CSM $\sim$, and then finding all their instances.

Concept similarity measures are widely used in ontology-based applications. For ontologies from the bio-medical field, such as the GeneOntology ontology [10], they are employed to discover functional similarities of genes. Furthermore, CSMs are used in ontology alignment algorithms. For DLs there exists a whole range of CSMs, which could be employed for the task of answering relaxed instance queries [1, 5, 12, 15]. In particular the CSMs generated by the framework described in [12] allow users to specify which part of the ontology's signature is to be regarded more important when it comes to the assessment of similarity of concepts. Thus, these measures naturally allow users to select important features of the query concept and which aspect of the query concept to relax.

We investigated algorithms for computing answers to relaxed instance queries for $\mathcal{EL}$. This DL has good computational properties and large, well-known biomedical ontologies such as the GeneOntology [10] are written in (polynomial extensions of) $\mathcal{EL}$. Our algorithm for computing relaxed instances w.r.t. $\mathcal{EL}$-KBs with general TBoxes relies on canonical models of the query concept and of the queried KB. The employed CSM is derived from a similarity measure for pointed interpretations, which essentially implements relaxed forms of equisimulation between interpretations. A similar idea in spirit is pursued in [16, 17] for $\mathcal{EL}$-concepts, where similarity is measured in terms of 'how much is missing' to establish a homomorphism between graph representations of $\mathcal{EL}$-concepts.

Now, for computing answers to relaxed instance queries, the similarity values for all pairs of elements between the two canonical models need to be computed in the worst case. Thus a naive implementation would hardly be efficient. We report in this paper in first optimizations for this novel inference, which we have implemented in the system ELASTIQ ($\mathcal{EL}$ answering of similarity-threshold instance queries). A first evaluation on the GeneOntology shows that the proposed optimizations are vital for this kind of inference, but that response times for a single query over large ontologies are still about a second.

The remainder of the paper in structured as follows. Next, we give an introduction to the technical terms used and the relaxed instance inference. In Section 3 we discuss the algorithm for computing relaxed instances and the similarity measures employed for it. The ELASTIQ reasoner is introduced in Section 4 together with some optimizations and the evaluation of its performance on the GeneOntology. The paper ends with conclusions and future work.

2 Preliminaries

$\mathcal{EL}$-concepts are built from two mutually disjoint sets N_C of *concept names*, and N_R of *role names* using the syntactic rule: $C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. The semantics of $\mathcal{EL}$-concepts are defined by means

of interpretations, in which concept names are interpreted as subsets of the interpretation domain and roles as binary relations. The semantics are extended to complex concepts as usual. An $\mathcal{EL}$-*TBox* consists of a finite set of *general concept inclusion axioms* (GCIs) of the form $C \sqsubseteq D$. An interpretation is a *model* for a TBox if it satisfies all its GCIs. An $\mathcal{EL}$-*ABox* describes individuals from a set N_I of *individual names* using concept assertions of the form $C(a)$ and role assertions of the form $r(a, b)$. Again, an interpretation is a model for an ABox if it satisfies all its assertions. An $\mathcal{EL}$-*knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The following commonly used reasoning tasks are implemented in most DL reasoning systems. *Concept subsumption* $C \sqsubseteq_{\mathcal{T}} D$ asks, for a TBox $\mathcal{T}$ and two concepts C and D, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. Given an individual a, a concept C, and a KB $\mathcal{K}$, a is called an *instance of* C w.r.t. $\mathcal{K}$, denoted $\mathcal{K} \models C(a)$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$. Given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a concept C, *instance retrieval* returns all individuals from $\mathcal{A}$ that are instances of C.

For the DL $\mathcal{EL}$, the polynomial-time complexity of most reasoning procedures rely on the fact that canonical models can be built, from which it is possible to read off entailments directly. These canonical models represent the most general model for a concept or the individuals of an ABox w.r.t. to a TBox. Before we can formally define these canonical models, we need to introduce some notation. If X is a concept, TBox, ABox, or KB, then:

- $\mathrm{Sig}(X)$ denotes the signature of X; that is, the set of concept, role, and individual names appearing in X, and
- $\mathrm{sub}(X)$ is the set of all sub-concepts of concepts occurring in X.

Definition 1. *(canonical models) Let C be an $\mathcal{EL}$-concept and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ an $\mathcal{EL}$-KB. The* canonical model $\mathcal{I}_{C,\mathcal{T}} = (\Delta^{\mathcal{I}_{C,\mathcal{T}}}, \cdot^{\mathcal{I}_{C,\mathcal{T}}})$ of C *w.r.t. the TBox $\mathcal{T}$ is:*

- $\Delta^{\mathcal{I}_{C,\mathcal{T}}} = \{d_C\} \cup \{d_D \mid \exists r.D \in sub(C) \cup sub(\mathcal{T})\}$
- $A^{\mathcal{I}_{C,\mathcal{T}}} = \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\}$, *for all concept names A, and*
- $r^{\mathcal{I}_{C,\mathcal{T}}} = \{(d_D, d_E) \mid D \sqsubseteq_{\mathcal{T}} \exists r.E\}$ *for all role names r.*

The canonical model $\mathcal{I}_{\mathcal{K}} = (\Delta^{\mathcal{I}_{\mathcal{K}}}, \cdot^{\mathcal{I}_{\mathcal{K}}})$ of the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *is defined as follows:*

- $\Delta^{\mathcal{I}_{\mathcal{K}}} = \{d_a \mid a \in Sig(\mathcal{A}) \cap N_I\} \cup \{d_C \mid \exists r.C \in sub(\mathcal{A}) \cup sub(\mathcal{T})\}$,
- $A^{\mathcal{I}_{\mathcal{K}}} = \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\} \cup \{d_a \mid \mathcal{K} \models A(a)\}$,
- $r^{\mathcal{I}_{\mathcal{K}}} = \{(d_D, d_E) \mid D \sqsubseteq_{\mathcal{T}} \exists r.E\} \cup \{(d_a, d_D) \mid \mathcal{K} \models \exists r.D(a)\} \cup \{(d_a, d_b) \mid r(a, b) \in \mathcal{A}\}$.

Note that canonical models for $\mathcal{EL}$ are always finite. Canonical models can be used to decide instance checking problems since a is an instance of C w.r.t. a KB $\mathcal{K}$ if and only if $a^{\mathcal{I}_{\mathcal{K}}}$ is an element of $C^{\mathcal{I}_{\mathcal{K}}}$ in the canonical model $\mathcal{I}_{\mathcal{K}}$ [13].These canonical models and their use for instance checking, are important for the algorithm for answering relaxed instance queries in Section 3.

Instance checking can be relaxed by using a concept similarity measure. Such a measure $\sim$ is a function that assigns to each pair of concepts (w.r.t. a TBox $\mathcal{T}$) a similarity value from the interval $[0, 1]$ with $C \sim C = 1$ for all concepts C. A

value $C \sim D = 0$ means that the concepts C and D are totally dissimilar, while a value of 1 indicates total similarity. A set of properties for CSMs was presented in [12]. In particular, a CSM $\sim$ is called *symmetric*, iff $C \sim D = D \sim C$; *equivalence invariant*, iff for all $C \equiv_\mathcal{T} D$ and all concepts E it holds that $C \sim E = D \sim E$; and *equivalence closed*, iff $C \equiv_\mathcal{T} D \iff C \sim D = 1$. Using those similarity measures, we define relaxed instances as follows:

Definition 2 (relaxed instance). *The individual a is a* relaxed instance *of the query concept Q w.r.t. the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the CSM $\sim_\mathcal{T}$ and the threshold $t \in [0, 1)$ iff there exists a concept X such that $Q \sim_\mathcal{T} X > t$ and $\mathcal{K} \models X(a)$.*

To compute the relaxed instances of an $\mathcal{EL}$-concept (w.r.t. an $\mathcal{EL}$-KB) it is not feasible to compute all sufficiently similar concepts and then perform instance checking for those, since (1) the number of such concepts can be infinite leading to an infinite number of queries and (2) a CSM does not necessarily provide a method how to obtain a 'sufficiently similar' concept.

3 The Algorithm for Computing Relaxed Instances

In [9], we proposed the CSM $\sim_c$ to be used to answer relaxed instance queries. This measure compares two concepts C and D w.r.t. a TBox $\mathcal{T}$ by computing their canonical models $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$ and comparing the structure of the models starting from the elements d_C and d_D, respectively. For this, we define a similarity measure $\sim_i$ on pointed interpretations. Given a pointed interpretation $p = (\mathcal{I}, d)$, we denote with

- $\mathrm{CN}(p) = \{A \in N_C \mid d \in A^\mathcal{I}\}$ the set of concept names that d is an instance of in $\mathcal{I}$, and
- $\mathrm{SC}(p) = \{(r, (\mathcal{I}, e)) \mid (d, e) \in r^\mathcal{I}\}$ the set of direct successors of d in $\mathcal{I}$.

The interpretation similarity $\sim_i$ to be defined depends on three parameters:

1. A *primitive measure* $\sim_p : N_C \times N_C \cup N_R \times N_R \to [0, 1]$ assigns a similarity value to each pair of concept names and each pair of role names. Any primitive measure has to satisfy that $x \sim_p x = 1$ for any concept or role name x. Additionally, for the similarity measure $\sim_i$ to be symmetric, $\sim_p$ needs to be symmetric as well. We give a default primitive measure, that simply assigns similarity 0 to pairs of different concept or role names x and y:

$$x \sim_{\text{default}} y = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

   However, other primitive measures are imaginable and useful. For example, one might want to express that two amounts Medium and High are more similar than Low and High, which can be achieved by using a primitive measure with $\mathsf{Medium} \sim_p \mathsf{High} = 0.5$ and $\mathsf{Low} \sim_p \mathsf{High} = 0$.

2. A *weighting function* $g : N_C \cup N_R \to \mathbb{R}_{>0}$ to prioritize different features in the similarity measure. We give a default weighting function g_{default}, that assigns 1 to every concept and role name, but again, other weighting functions can be useful for certain cases.
3. A *discounting factor* w is a constant that allows for discounting of successors, and should have a value $0 < w < 1$.

Definition 3. *Given a primitive measure $\sim_p$, a weighting function g and the discounting factor w, the* interpretation similarity measure $\sim_i$ *is defined as:*

$$p \sim_i q = \frac{sim_{CN}(p,q) + sim_{CN}(q,p) + sim_{SC}(p,q) + sim_{SC}(q,p)}{\sum_{C \in CN(p)} g(C) + \sum_{D \in CN(q)} g(D) + \sum_{(r,p') \in SC(p)} g(r) + \sum_{(s,q') \in SC(q)} g(s)},$$

where

$$sim_{CN}(p,q) = \sum_{A \in CN(p)} \max_{B \in CN(q)} g(A)(A \sim_p B), \text{ and}$$

$$sim_{SC}(p,q) = \sum_{(r,p') \in SC(p)} \max_{(s,q') \in SC(q)} g(r)(r \sim_p s)\big(w + (1-w)(p' \sim_i q')\big).$$

If all of the sets $CN(p)$, $CN(q)$, $SC(p)$, and $SC(q)$ are empty for pointed interpretations p, q, we define $p \sim_i q = 1$.

Note that $\sim_i$ does not necessarily yield an equivalence closed or equivalence invariant CSM. To regain these properties, one can first normalize the interpretations before applying the $\sim_i$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is in *interpretation normal form* if there are no elements $a, b, c \in \Delta^{\mathcal{I}}$, $b \neq c$, such that $\{(a,b),(a,c)\} \subseteq r^{\mathcal{I}}$ and for all concepts C with $b \in C^{\mathcal{I}}$ also $c \in C^{\mathcal{I}}$ holds; i.e., no node has two successors for the same role name whose instantiators are in a subset relation[3]. Any interpretation $\mathcal{I}$ can be transformed into normal form in polynomial time by removing all redundant successors.

Let $\mathcal{I}'_{C,\mathcal{T}}$ and $\mathcal{I}'_{D,\mathcal{T}}$ denote canonical models in interpretation normal form, then the CSM $\sim_c$ is defined as follows:

$$C \sim_c D = (\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D).$$

We showed that $\sim_c$ is indeed symmetric, equivalence invariant, and equivalence closed [9].

Example 1. Consider the concepts:
$C_{\mathsf{ex}} = \mathsf{Server} \sqcap \exists \mathsf{hasLoad}.\mathsf{Medium} \sqcap \exists \mathsf{provides}.(\mathsf{VideoStreamService} \sqcap \mathsf{Service})$ and
$D_{\mathsf{ex}} = \mathsf{Server} \sqcap \exists \mathsf{hasLoad}.\mathsf{Low} \sqcap \exists \mathsf{provides}.(\mathsf{DBService} \sqcap \mathsf{Service} \sqcap \exists \mathsf{queryLang}.\mathsf{SQL})$.
We use the primitive measure $\sim_p$, which is almost the default primitive measure, except for $\mathsf{Low} \sim_p \mathsf{Medium} = \mathsf{Medium} \sim_p \mathsf{High} = 0.5$ instead of 0. We also

[3] Formally, one describes this using the notion of a *simulation* between b and c in $\mathcal{I}$, see [13] for more details.

use the default weighting function g_{default} and the discounting factor $w = 0.8$. To compute the similarity between C_{ex} and D_{ex}, we have to compute their normalized canonical models (w.r.t. to the empty TBox). Based on these, we obtain: The $\mathsf{hasLoad}$-successors of C_{ex} and D_{ex} have a similarity of 0.5, since $\mathsf{Medium} \sim_p \mathsf{Low} = 0.5$. Both $\mathsf{provides}$-successors of C_{ex} and D_{ex}, are instances of $\mathsf{Service}$, while the concept names $\mathsf{VideoStreamService}$ and $\mathsf{DBService}$ have no correspondence. Similarly, only the $\mathsf{provides}$-successor of D_{ex} has an existential restriction, resulting in a value of 0 for sim_{SC} in both directions. Overall, this yields a similarity of $\frac{(1+0)+(1+0)+0+0}{2+2+0+1} = 0.4$ for the two services. Using this, we can finally compute the similarity between C_{ex} and D_{ex}:

$$\begin{aligned}
&\text{sim}_{\text{CN}}(C_{\mathsf{ex}}, D_{\mathsf{ex}}) = \text{sim}_{\text{CN}}(D_{\mathsf{ex}}, C_{\mathsf{ex}}) = 1\\
&\text{sim}_{\text{SC}}(C_{\mathsf{ex}}, D_{\mathsf{ex}}) = \text{sim}_{\text{SC}}(D_{\mathsf{ex}}, C_{\mathsf{ex}}) = (0.2 + 0.8 \cdot 0.5) + (0.2 + 0.8 \cdot 0.4) = 1.12\\
&C_{\mathsf{ex}} \sim_c D_{\mathsf{ex}} = \frac{1 + 1 + 1.12 + 1.12}{1 + 1 + 2 + 2} = 0.707.
\end{aligned}$$

The procedure to compute relaxed instances of a query concept Q w.r.t. a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a threshold t, and the CSM $\sim_c$ has the following steps [9]:

1. Compute the canonical models $\mathcal{I}_{Q,\mathcal{T}}$ and $\mathcal{I}_{\mathcal{K}}$ of the query concept Q and the ABox $\mathcal{A}$ w.r.t. the TBox $\mathcal{T}$.
2. Transform these models into $\mathcal{I}'_{Q,\mathcal{T}}$ and $\mathcal{I}'_{\mathcal{K}}$.
3. Define a *maximal interpretation similarity* $\sim_i^{\max}$ between the normalized canonical models. The measure $\sim_i^{\max}$ is behaves like $\sim_i$, but chooses a subset of the concept names and successors for elements in the canonical model $\mathcal{I}_{\mathcal{K}}$ in a way to maximize the similarity value.
4. For each individual a occurring in $\mathcal{K}$, check if the maximal interpretation similarity between the element d_Q in $\mathcal{I}'_{Q,\mathcal{T}}$ and the element d_a in $\mathcal{I}'_{\mathcal{K}}$ is larger than the given threshold t. If so, a is a relaxed instance and is returned.

Formally, $\sim_i^{\max}$ is defined as the unique solution of the following equation system:

$$p \sim_i^{\max} q = \max_{\substack{C_q \subseteq \text{CN}(q)\\ S_q \subseteq \text{SC}(q)}} \frac{\left(\begin{array}{c}\text{sim}_{\text{CN}}\big(\text{CN}(p), C_q\big) + \text{sim}_{\text{CN}}\big(C_q, \text{CN}(p)\big)\\ + \text{sim}_{\text{SC}}\big(\text{SC}(p), S_q\big) + \text{sim}_{\text{SC}}\big(S_q, \text{SC}(p)\big)\end{array}\right)}{\sum_{C \in \text{CN}(p)} g(C) + \sum_{D \in C_q} g(D) + \sum_{(r,p') \in \text{SC}(p)} g(r) + \sum_{(s,q') \in S_q} g(s)}, \quad (1)$$

where

$$\begin{aligned}
&\text{sim}_{\text{CN}}(C_1, C_2) = \sum_{A \in C_1} \max_{B \in C_2} g(A)(A \sim_p B), \text{ and}\\
&\text{sim}_{\text{SC}}(S_1, S_2) = \sum_{(r,p') \in S_1} \max_{(s,q') \in S_2} g(r)(r \sim_p s)(w + (1-w)(p' \sim_i^{\max} q')).
\end{aligned}$$

We showed that using the maximal interpretation similarity indeed solves the problem of answering relaxed instance queries correctly, see [9].

Theorem 1 ([9]). *Individual a is a relaxed instance of Q w.r.t. $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$), t, and $\sim_c$ iff $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{max} (\mathcal{I}'_{\mathcal{K}}, d_a) > t$.*

We showed an upper complexity bound of NP by (non-deterministically) translating the equation system that defines $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a)$ into a linear programming problem of polynomial size and solving it. However, this approach is not practical. Instead, ELASTIQ implements an iterative approach, that refines the similarity values and converges to $\sim_i^{\max}$ in the limit. This approach which we present next is sound, as it converges from below, but not necessarily complete.

4 The ELASTIQ Reasoner

ELASTIQ is the first implementation for answering instance queries relaxed by a similarity measure. Given an $\mathcal{EL}$-ontology, an $\mathcal{EL}$-query concept, an instantiation of $\sim_c$ and a value for a threshold, ELASTIQ computes a result set of ABox individuals, where each of these individuals is relaxed instance. The CSM employed here is fixed to $\sim_c$ as defined in Section 3, but it and can be adjusted by a custom weighting function, primitive similarity measure and the discounting factor. Computing an answer to a relaxed instance query by ELASTIQ consists of four main steps.

Step 1: Global preprocessing. The canonical model $\mathcal{I}_{\mathcal{K}}$ of the ABox and the TBox is generated by the use of a standard DL reasoner, currently ELASTIQ uses the ELK system [11].

Step 2: Local preprocessing. The canonical model of the query concept w.r.t. the TBox $\mathcal{I}_{Q,\mathcal{T}}$ is generated—as in Step 1 by the use of ELK.

ELASTIQ distinguishes the two preprocessing steps for the sake of computing several relaxed instance queries against the same KB faster. Obviously, $\mathcal{I}_{\mathcal{K}}$ does not depend on the query concept and can therefore be reused for every subsequent queries. The model $\mathcal{I}_{Q,\mathcal{T}}$, however, needs to be recreated for every different query concept Q. In both steps we use the ELK reasoner [11] to compute classification and realization of the ontology, and then retrieve subsumption and instance relationships from the results. ELASTIQ only needs to consider those domain elements that are reachable from elements representing ABox individuals and thus can be used by the main algorithm. Similarly, for $\mathcal{I}_{Q,\mathcal{T}}$ ELASTIQ creates only those domain elements that are reachable through successor relations from d_Q. The normal forms of the canonical models are computed on demand. Finally, Step 2 also initializes the data structure for the main computation in Step 3.

Step 3: Computing the maximal interpretation similarity $\sim_i^{max}$. Recall, that ELASTIQ implements an iterative approach, that refines the similarity values and converges to $\sim_i^{\max}$ in the limit. Thus the main computation yields a sequence of matrices, each representing an iteration of the computation. The rows of such a matrix M_j represent domain elements from $\mathcal{I}_{Q,\mathcal{T}}$ and the columns domain elements from $\mathcal{I}_{\mathcal{K}}$. The values inside each cell of M_j, are identified by two domain elements $d \in \Delta^{\mathcal{I}_{Q,\mathcal{T}}}$ and $e \in \Delta^{\mathcal{I}_{\mathcal{K}}}$, and converge towards $(\mathcal{I}_{Q,\mathcal{T}}, d) \sim_i^{\max} (\mathcal{I}_{\mathcal{K}}, e)$

for $j \to \infty$ [9]. Instead of computing the similarity values for all pairs of elements from the canonical models in each iteration, ELASTIQ restricts the entries in M_j to those elements that are reachable from (elements representing) ABox individuals by paths in $\mathcal{I}_\mathcal{K}$. To this end, M_0 is initialized with one row (for d_Q) and as many columns as there are individuals in the ABox. The set of columns is extended with new elements (d', e') if there exists an element (d, e) in M_0 such that d and d' are connected in $\mathcal{I}_{Q,\mathcal{T}}$ via some role r, and e and e' are connected in $\mathcal{I}_\mathcal{K}$ via some role s. Since the canonical models $\mathcal{I}_{Q\mathcal{T}}$ and $\mathcal{I}_\mathcal{K}$ are finite, the size of M_0 is bounded by $|\Delta^{\mathcal{I}_{Q,\mathcal{T}}}| \cdot |\Delta^{\mathcal{I}_\mathcal{K}}|$. Once all reachable pairs have been added to M_0, it contains values exactly for those pairs that are necessary for computing similarities between the domain elements that we are interested in—namely similarities of the query concept and each ABox individual (d_Q, d_{a_i}).

Each iteration $j + 1$ creates a new matrix M_{j+1}, and computes the values by applying Equation (1) to the values in M_j. ELASTIQ needs only to keep the current matrix M_{j+1} and the last one M_j ($j \geq 0$) in memory. The iterations for the refinement of similarity values proceeds until one of the following termination criteria is met:

- the maximal amount of iterations i_{max} specified by the user is reached; or
- no interesting similarity value (d_Q, d_a) has changed during the last iteration by more than a relative factor specified by the user.

Step 4: Comparison with t. After the iteration stopped, the interesting similarity values $M_j(d_Q, d_a)$ are compared to the input threshold t and the answer set of individuals is compiled. This set is then listed in descending order of similarity.

4.1 Optimizations for Computing Relaxed Instances

A naive implementation of the algorithm can hardly compute relaxed instances for reasonably large ontologies in acceptable time. As mentioned before, a highly effective optimization is the reuse of $\mathcal{I}_\mathcal{K}$ for multiple queries. Since ABoxes are usually much larger than query concepts, the model $\mathcal{I}_\mathcal{K}$ is also be much more costly to create than the models $\mathcal{I}_{Q,\mathcal{T}}$. Additionally, the normalization of canonical models can be done more efficiently than by computing simulations to determine unnecessary role-successors. Before adding a domain element d_C as an r-successor to some element d_D ELASTIQ checks whether there already exists an r-successor d_E for d_D such that $E \sqsubseteq C$. In this case normalization would eliminate d_C, thus avoiding the introduction of d_C (and its role successors) improves the runtime of canonical model generation further. Similarly, when adding d_C as an r-successor to d_D, ELASTIQ eliminates all r-successors d_E of d_D if $C \sqsubseteq E$.

During the generation of the canonical models ELASTIQ performs many subsumption checks. Although ELK is currently one of the fastest reasoners for $\mathcal{EL}$, caching of sub- and superclass relations yielded a great performance boost, since ELASTIQ needs to access sub- and superclass relationships for the same class several times.

The algorithm from Section 3 suggests to iterate over all subsets of CN and SC successors in order to find the maximal similarity. This exponential procedure

can be improved by looking at the primitive similarities between elements. Let $d \in \mathcal{I}_{Q,\mathcal{T}}$ and $e \in \mathcal{I}_{\mathcal{K}}$. By definition of $\sim_i^{\max}$ we are looking for those subsets of the concept names and successors of e that maximize the similarity. Instead of iterating over all subsets of $\mathrm{CN}(e)$ to find the best pairing, we showed that if $B \in \mathrm{CN}(e)$ such that $\exists A \in \mathrm{CN}(d)$ with $A \sim_p B = 1$, we can always keep B in the subset of $\mathrm{CN}(e)$, because it will always increase the similarity. Conversely, if $B' \in \mathrm{CN}(e)$ such that $\forall A \in \mathrm{CN}(d)$, then $A \sim_p B' = 0$, and B' can be left out of the subset of $\mathrm{CN}(e)$, since it cannot increase the similarity. Analogously, we can remove (s, q) from $\mathrm{SC}(e)$ if for all $(r, p) \in \mathrm{SC}(d)$ we have $r \sim_p s = 0$. This can dramatically reduce the number of subsets to be checked. In fact, for the default primitive measure, this means that the best subset of concept names can always be computed in linear time by checking each concept name in $\mathrm{CN}(e)$ separately.

4.2 Evaluation

Our preliminary performance evaluation of ELASTIQ used different versions of the GeneOntology that describe *schizosaccharomyces pombe*—some species of yeast. These ontologies ranged from 9,157 concept names and 34,875 individuals in the first version to 51,949 concept names and 289,206 individuals for the 15th version. The sizes of canonical models ranged from 77,941 to 602,548 elements.

We obtained our test ontologies by custom dataset generation provided by the Manchester OWL Corpus [14]. These GeneOntology versions are anonymised and therefore any contentual interpretation of our results is virtually impossible. We restricted our investigations solely to the performance of ELASTIQ and leave the intricate task of a quality assessment for future work. We discovered that for each individual e there exists a very fragmented concept assertion in the ABox of the form $\exists is\_a.C_e(e)$, where the qualification C_e is rarely larger than 3 conjuncts with a role-depth of at most 2.

Our test suite contains 10 randomly generated query concepts with increasingly complex structures. These queries were built over the common signature of versions 1–15 of the GeneOntology (approximately 1,000 concept names and 4 roles). The smallest query (Query 1) only contained 6 concept and role names and had a role-depth of 2, while the Query 10 had a size of 670 and a role-depth of 5. Due to the plain structure of concept assertions we wrapped each query concept Q_i with $\exists is\_a.Q_i$ in order to provoke a more complex computation. For these queries, the sizes of the canonical models $\mathcal{I}_{Q,\mathcal{T}}$ ranged from 2 to 236 elements. We evaluated the queries for the default primitive measure and weighting function, and counted the number of relaxed instances for a threshold of $t = 0.333$. The test system had a 1800 MHz dual core processor AMD Turion II Neo and 6 GB of RAM. Figure 1 shows the runtime of ELASTIQ for answering all 10 relaxed instance queries w.r.t. each ontology version. The high runtimes for ontology versions 11, 12, and especially 13–15 is mainly due to the increase of the size of the canonical model $\mathcal{I}_{\mathcal{K}}$. Most queries returned a lot more relaxed instances for the ontologies 11–15 than for ontologies 1–10. Queries 8 and 9 returned the largest number of relaxed instances, up to over 200,000 for Query 8 evaluated on nnotations15.owl.

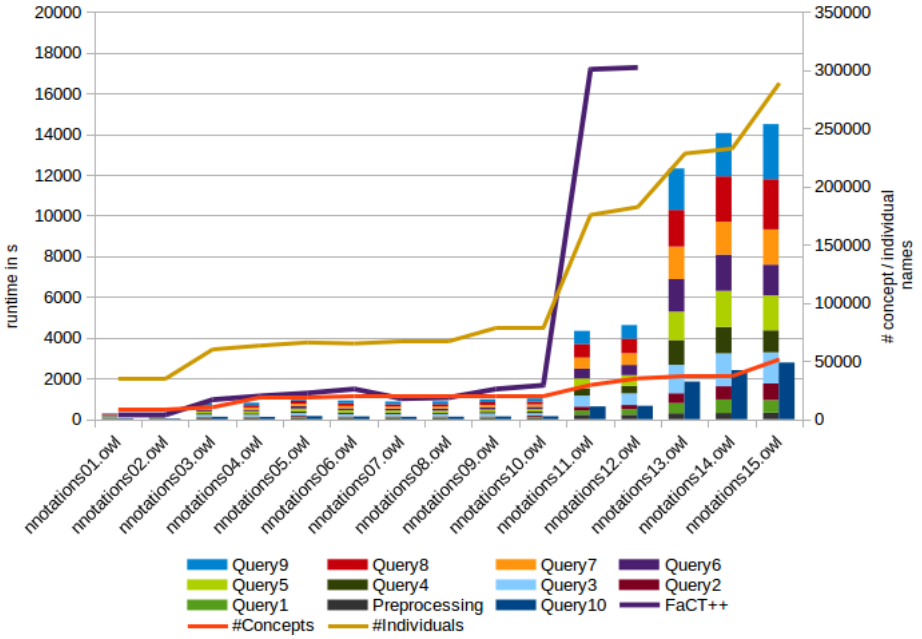


Fig. 1. ELASTIQ's runtime for answering relaxed instance queries in different versions of the GeneOntology

When breaking down the times for preprocessing and the query answering further, it shows that the preprocessing time is dominated by the flattening of the ontology, the reasoning done by ELK, and the construction of the canonical model $\mathcal{I}_{\mathcal{K}}$, while the time to construct the canonical models $\mathcal{I}_{Q,\mathcal{T}}$ is negligible. However, the overall query answering time is largely spend on Step 3, i.e., the iterations to compute the maximal similarity.

ELASTIQ performs ABox realization to obtain $\mathcal{I}_{\mathcal{K}}$ and in addition a kind of relaxed ABox realization for the query concepts in the test suite. Now, while it is clear that ELASTIQ is slower than ELK for ABox realization, it showed, suprisingly, that this does not need to be the case for other optimized DL reasoners. We compared ELASTIQ's overall reasoning times with the ABox realization times of the commonly used FaCT++ reasoner [18]. Figure 2 shows that ELASTIQ mostly performed better than FaCT++, although solving a more complex task.[4] However, computation times of more than a minute for 10 relaxed queries over ABoxes with 1,000 individuals still calls for further improvement.

5 Conclusions and Future Work

In this paper we investigated the novel inference of answering relaxed instance queries. These queries can be gradually relaxed by varying the threshold, while

[4] Note that FaCT++ classification resulted in an error for ontologies 13–15.

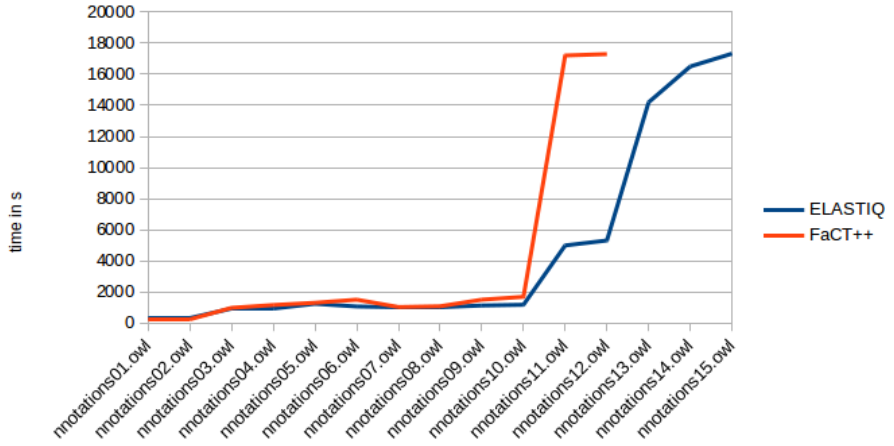


Fig. 2. Runtime of ELASTIQ compared to FaCT++ ABox Realization times.

the similarity measure allows to specify which parts of the query can be relaxed and which should be kept. We devised a concept similarity measure, $\sim_c$, that works for general $\mathcal{EL}$-TBoxes, and showed how to relax instance queries using this measure. We presented ELASTIQ, a prototype sytem for relaxed instance query answering, some straight-forward, but highly effective optimization techniques, and gave a first performance evaluation using different queries on increasingly complex versions of the GeneOntology.

It turned out that for ontologies with large ABoxes, ELASTIQ is still not fast enough. We want to explore further optimizations for the computation of $\sim_i^{max}$. Currently, the matrices converge to $\sim_i^{max}$ from below. With upper bounds on the maximal similarity, it would be possible to prune computation early on individuals that are certainly not relaxed instances. While currently the algorithm decides which individuals are certainly relaxed answers, an upper bound could also be used to determine which individuals are certainly not relaxed answers, therefore making the approach not just sound, but complete. We also need to perform further evaluations for the performance on ontologies and query concepts from other domains, but also to evaluate the quality of the answers returned by ELASTIQ in regard of the different CSMs in use.

Currently, one needs to specify a threshold, above which individuals are considered as relaxed answers. This threshold approach guarantees a minimal similarity and hence quality of the result, but it is hard to predict how many relaxed answers a query would return for a certain threshold. In cases where just the first few most similar relaxed answers are of interest, top-k answering would be more useful. We are investigating to efficiently implement this type of answering mechanism. Finally, it would be useful to not just consider instance queries, but more expressive query types as well. We are currently working on the theoretical basis for relaxing conjunctive queries.

References

1. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147 of *CEUR Workshop Proceedings*, 2005.
2. S. Borgwardt, F. Distel, and R. Peñaloza. How fuzzy is my fuzzy description logic? In B. Gramlich, D. Miller, and U. Sattler, editors, *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*, volume 7364 of *Lecture Notes In Artificial Intelligence*, pages 82–96. Springer-Verlag, 2012.
3. S. Borgwardt and R. Peñaloza. Undecidability of fuzzy description logics. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-12)*, pages 232–242. AAAI Press, 2012.
4. M. Cerami and U. Straccia. On the (un)decidability of fuzzy description logics under lukasiewicz t-norm. *Inf. Sci.*, 227:1–21, 2013.
5. C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Proc. of Convegno Italiano di Logica Computazionale, CILC05*, 2005.
6. A. Ecke. Similarity-based relaxed instance queries in $\mathcal{EL}^{++}$. In *Proceedings of the First Workshop on Logics for Reasoning about Preferences, Uncertainty, and Vagueness*, CEUR-WS. CEUR, 2014.
7. A. Ecke, R. Peñaloza, and A.-Y. Turhan. Towards instance query answering for concepts relaxed by similarity measures. In *Workshop on Weighted Logics for AI (in conjunction with IJCAI'13)*, Beijing, China, 2013.
8. A. Ecke, R. Peñaloza, and A.-Y. Turhan. Answering instance queries relaxed by concept similarity. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*, pages 248–257. AAAI Press, 2014.
9. A. Ecke, R. Peñaloza, and A.-Y. Turhan. Similarity-based relaxed instance queries. *Journal of Applied Logic*, 2015. In press.
10. T. Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
11. Y. Kazakov, M. Krötzsch, and F. Simančík. The incredible ELK: from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reasoning*, 53(1):1–61, 2014.
12. K. Lehmann and A.-Y. Turhan. A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts. In L. F. del Cerro, A. Herzig, and J. Mengin, editors, *Proc. of the 13th European Conf. on Logics in A.I. (JELIA 2012)*, Lecture Notes In Artificial Intelligence, pages 307–319. Springer, 2012.
13. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194–228, 2010.
14. N. Matentzoglu, S. Bail, and B. Parsia. A snapshot of the OWL web. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 331–346, 2013.
15. B. Suntisrivaraporn. A similarity measure for the description logic $\mathcal{EL}$ with unfoldable terminologies. In *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pages 408–413, 2013.
16. S. Tongphu and B. Suntisrivaraporn. A non-standard instance checking for the description logic $\mathcal{ELH}$. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2014)*, Rome, Italy, 2014.

17. S. Tongphu and B. Suntisrivaraporn. On desirable properties of the structural subsumption-based similarity measure. In T. Supnithi, T. Yamaguchi, J. Z. Pan, V. Wuwongse, and M. Buranarach, editors, *Proceedings of the 4th Joint International Conference on Semantic Technology, (JIST 2014)*, volume 8943 of *LNCS*, pages 19–32. Springer, 2014.
18. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *Proceedings of the Third International Joint Conference on Automated Reasoning*, IJCAR'06, pages 292–297, Berlin, Heidelberg, 2006. Springer-Verlag.

Polynomial encoding of ORM conceptual models in $\mathcal{CFDI}_{nc}^{\forall-}$

Pablo Rubén Fillottrani[1,2], C. Maria Keet[3], David Toman[4]

[1] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, prf@cs.uns.edu.ar
[2] Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina
[3] Department of Computer Science, University of Cape Town, South Africa
mkeet@cs.uct.ac.za
[4] Cheriton School of Computer Science, University of Waterloo, Canada
david@cs.uwaterloo.ca

Abstract. The use of conceptual models has long been confined to the data analysis stage of software development. In recent years, this has been extended to use them at run-time as well, for, among others, querying large amounts of data. This brings afore the need to have tractable logic-based reconstructions of the conceptual models, i.e., in at most PTIME. We provide such a logic-based reconstruction for most of ORM using the Description Logic language $\mathcal{CFDI}_{nc}^{\forall-}$, which has several features important for conceptual models, notably n-ary relationships, complex identification constraints, and role subsumption. The encoding captures over 96% of the constructs used in practice in the set of 33 ORM diagrams analysed. The results are easily transferable to EER and UML Class diagrams, with an even greater coverage.

1 Introduction

While for many years conceptual models were developed and then shelved upon implementation or used 'offline' for documentation purposes, recent years has seen an increase in using such precious resources computationally. One strand of investigation focuses on expressiveness and a logic foundation to compute satisfiability and detect inconsistencies over the TBox only, among others [2, 5, 8, 16, 24], and several implementations exists using various technologies that are more or less scalable; e.g., using OCL [16], Alloy [8, 23], or a DL reasoner [19]. Another looks at usage during runtime for a range of purposes, such as using a conceptual model for scalable test data generation [35] and for designing [6] and executing [13] queries. Approximations of conceptual models are used also deeper into the machinery of querying databases, in particular during various stages of query compilation, e.g., when reasoning about duplicates [39]. The former purpose requires a logic foundation using a (very) expressive logic, whereas the latter requires computationally better behaved logics to keep the whole process feasible. What such a tractable logic language looks like that captures most, or all, important conceptual data modelling language features, has received little

attention, however. To the best of our knowledge, there are only four efforts to capture a fragment of which concept or full satisfiability checking is (or is claimed to be) in polynomial time (or less): one for ER [2], two for UML [1, 25], and one for ORM [35], with the first one logic-driven and the last one implementation-driven, and they all differ substantially in coverage of features. The main general question, thus, is: what is a useful fragment for tractable runtime usage of a conceptual data model? Then, how to formalise it.

To answer this question, we choose to focus on ORM first, for it is strictly more expressive than ER and UML class diagrams, and then facilitating transferability of the results. Given that there is a lot of insight in computational complexity of Description Logic (DL) languages and which ones are in P, we zoom in on those for a logic-based reconstruction. It appears that the DL $\mathcal{CFDI}_{nc}^{\forall-}$ [40] is a good fit and can capture 96.5% of the ORM models in a dataset of 33 ORM models that are part of the dataset of 101 conceptual models (dataset available from [28]). This is chiefly thanks to 'trading' costly but lesser used covering constraints for the more often used arbitrary identifiers and n-ary relationships.

In the remainder of the paper, we first provide background definitions in Section 2. The main results are presented in Section 3, and are discussed and compared with related research in Section 4. We summarize our results and briefly outline future work in Section 5.

2 Preliminaries

The two preliminaries are the DL $\mathcal{CFDI}_{nc}^{\forall-}$ and the ORM language, so as to keep the paper self-contained.

2.1 The Description Logic $\mathcal{CFDI}_{nc}^{\forall-}$

All members of the $\mathcal{CFD}$ family of DLs are fragments of FOL with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *features*.

Definition 1 ($\mathcal{CFDI}_{nc}^{\forall-}$ Knowledge Bases) Let F, PC and IN be disjoint sets of (names of) functional features, primitive concepts and individuals, respectively. A *path function* Pf is a word in F^*, and we denote the empty word by *id* and concatenation by ".". Metadata and data in a $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base $\mathcal{K}$ are respectively defined by a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. A TBox $\mathcal{T}$ consists of a finite set of *inclusion dependencies* of the form

$$\mathrm{A} \sqsubseteq \mathrm{B}, \quad \mathrm{A} \sqsubseteq \neg \mathrm{B}, \quad \mathrm{A} \sqsubseteq \forall f.\mathrm{B}, \quad \forall f.\mathrm{A} \sqsubseteq \mathrm{B}, \quad \mathrm{A} \sqsubseteq \exists f^{-1}, \quad \text{or} \quad \mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$$

where $\mathrm{A}, \mathrm{B} \in \mathsf{PC}$, $f \in \mathsf{F}$, and $\mathsf{Pf}_i \in \mathsf{F}^*$. A concept "$\mathrm{B} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$" that participates in the last dependency is called a *path functional dependency* (PFD). An ABox $\mathcal{A}$ consists of a finite set of facts in the form of *concept assertions* $\mathrm{A}(a)$, and *function assertions* $f(a) = b$ where $\mathrm{A} \in \mathsf{PC}$, $f \in \mathsf{F}$, and $a, b \in \mathsf{IN}$. Any PFD

occurring in $\mathcal{T}$ must also satisfy a *regularity* condition by adhering to one of the following two forms:

$$\mathrm{C} : \mathsf{Pf} \,.\, \mathsf{Pf}_1, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \quad \text{or} \quad \mathrm{C} : \mathsf{Pf} \,.\, \mathsf{Pf}_1, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.g. \tag{1}$$

A PFD is a *key* if it adheres to the first of these forms.

The semantics is defined in the standard way with respect to an interpretation $\mathcal{I} = (\triangle, (\cdot)^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretation of primitive concepts A to be subsets of $\triangle$, features f to be total functions on $\triangle$, and individuals a to be elements of $\triangle$. The interpretation function is extended to path expressions by interpreting id, the empty word, as the identity function $\lambda x.x$, concatenation as function composition, and to derived concept descriptions as follows:

$$\begin{aligned}
(\neg \mathrm{A})^{\mathcal{I}} &= \triangle \setminus \mathrm{A}^{\mathcal{I}} \\
(\forall f.\mathrm{A})^{\mathcal{I}} &= \{x \mid f^{\mathcal{I}}(x) \in \mathrm{A}^{\mathcal{I}}\} \\
(\exists f^{-1})^{\mathcal{I}} &= \{x \mid \exists y \in \triangle : f^{\mathcal{I}}(y) = x\} \\
(\mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf})^{\mathcal{I}} &= \{x \mid \forall y \in \mathrm{C}^{\mathcal{I}} : (\textstyle\bigwedge_{i=1}^{k} \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y)) \\
&\qquad \Rightarrow \mathsf{Pf}^{\mathcal{I}}(x) = \mathsf{Pf}^{\mathcal{I}}(y)\}
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies an inclusion dependency $\mathrm{C} \sqsubseteq \mathrm{D}$ if $\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$, a concept assertion $\mathrm{A}(a)$ if $a^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$, and a function assertion $f(a) = b$ if $f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$. $\mathcal{I}$ satisfies a knowledge base $\mathcal{K}$ if it satisfies each inclusion dependency and assertion in $\mathcal{K}$. In addition, every $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base must satisfy the following two conditions.

1. *(inverse feature and value restriction interaction)* If $\{\mathrm{A} \sqsubseteq \exists f^{-1}, \forall f.\mathrm{A}' \sqsubseteq \mathrm{B}\} \subseteq \mathcal{T}$ then (a) $\mathrm{A} \sqsubseteq \mathrm{A}' \in \mathcal{T}$, (b) $\mathrm{A}' \sqsubseteq \mathrm{A} \in \mathcal{T}$ or (c) $\mathrm{A} \sqsubseteq \neg \mathrm{A}' \in \mathcal{T}$.

2. *(inverse feature and PFD interaction)* Any non-key PFD occurring in $\mathcal{T}$ that involves features used in the $\exists f^{-1}$ concept in $\mathcal{T}$ must also satisfy a *stronger* regularity condition by adhering to the following form:

$$\mathrm{C} : \mathsf{Pf} \,.f, \mathsf{Pf}_2, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.g. \tag{2}$$

Proposition 1 ([40]). *Consistency of $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases is complete for PTIME.*

Other reasoning tasks, such as logical implication and concept consistency reduce (linearly) to knowledge base consistency. Relaxing either of the aforementioned conditions leads to EXPTIME and PSPACE completeness, respectively [40]. Note also, that condition (2) imposed on PFDs applies only to non-key PFDs. Overall, however, the restrictions do not seem to impact the modeling utility of $\mathcal{CFDI}_{nc}^{\forall-}$ in relation to keys and functional constraints. Indeed, arbitrary functional dependencies in relational schema are easily captured. Finally and for convenience $\mathcal{CFDI}_{nc}^{\forall-}$ supports additional syntax, e.g., subsumptions of the form $\exists f^{-1} \sqsubseteq \mathrm{A}$. This additional syntax is mere syntactic sugar and can be equivalently expressed in $\mathcal{CFDI}_{nc}^{\forall-}$ as defined above [40].

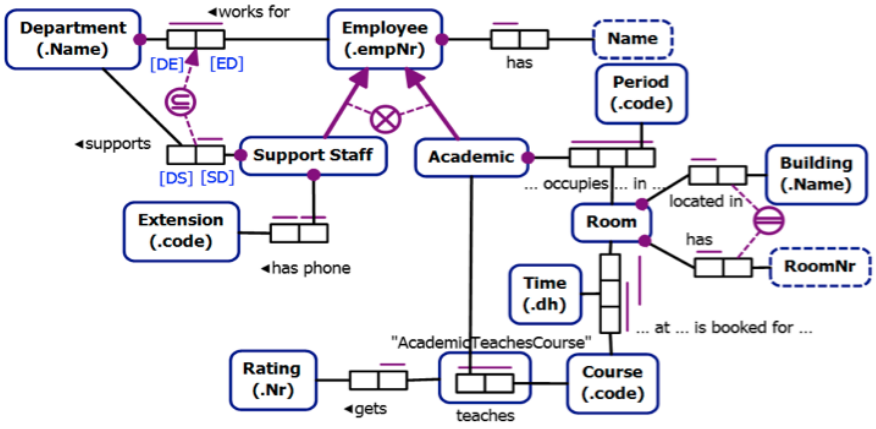


Fig. 1. Sample ORM2 diagram for a particular university where, among others, each Support Staff must have exactly one Extension that is identified by a code, Support Staff and Academics are subtypes of Employee and are disjoint, each Room is identified by the Building and its RoomNr, each Employee has exactly one Name, and a support staff that supports a department also works for that department.

2.2 Brief overview of ORM2

Object-Role Modelling as a general term is also known as Fact-Based Modeling and comprises several notational variants with varying language features, ranging from its predecessor NIAM and its recent notation in the proprietary CogNIAM tool of PNA, to the FCO-IM flavour [4], to ORM as in [21] popularised with VisioModeler 3.1 and ORM2 [22] popularised with the NORMA tool [15] as plugin for MS Visio, among others. We use Halpin's notation [22] in the remainder of the paper for its compactness. As we are interested in the static, structural components, we ignore deontic constraints[5]—they did not occur in any of the evaluated ORM models anyway [28]—and derived constraints (ORM's version of methods). ORM2 has four different kinds of named entities:

- *entity type*, which is like an EER entity type and UML class and may be objectified, and is denoted with a blue rectangle with round corners;
- *value type*, which is an entity type that has a binary fact type (relationship) to a data type, denoted with a blue dashed rectangle with round corners;
- *role* that each entity/value type plays in a fact type, denoted with a small rectangle and connected to the object type or value type;
- *n-ary fact type* ($n \geq 1$) that relates entity types to each other or entity types to value types and they have to be elementary (uniqueness constraint spanning n or $n-1$ roles), denoted with a rectangle composed of the roles.

Typically, roles and fact types are named automatically, but this can be added, as indicated with the user-named role [DE] in Fig. 1; note that the text next to the fact types are 'readings' for model verbalisations, not fact type names.

[5] Refer to [36] for a treatment of deontic constraints in the context of DLs and SBVR

ORM has many constraint types (some of which are used rarely [28]); a non-exhaustive example is shown in Fig. 1. The ones used in the figure are mandatory (small blob), uniqueness (line over rectangle), reference schemes (simple identifiers, e.g., .empNr), and external identifiers (circle with double horizontal line), subtype (arrow) and subset (over roles, fact types, paths), disjointness (encircled cross); others include coverage, constraints on values, and 11 relationship constraints (a.o., transitive, irreflexive).

As transformations exist from one ORM model to UML and to ER [22], ORM is 'more conceptual' than the other models for one does not have to commit to an implementation paradigm upfront. Moreover, the notation is also used for business rules specifications [32].

3 ORM2$_{cfd}$ fragment into $\mathcal{CFDI}_{nc}^{\forall -}$

While we primarily focus on ORM to $\mathcal{CFDI}_{nc}^{\forall -}$ mapping here, we also aim for an easy extension to EER and UML Class diagrams, the ability to use it for inter-model assertions across models represented in different languages [17], and for unification of the conceptual modelling language families for a widely used subset of features. To this end, we first define the syntax and semantics of a 'generic' conceptual data modelling language, which we name $\mathcal{CM}_{com}^{-}$, as it is, essentially, a proper fragment of $\mathcal{CM}_{com}$—used as common conceptual data modelling language for EER, UML, and ORM [30]—without covering and disjunctive mandatory constraints, and with limited cardinalities and a more precise definition of relationship subsumption and disjointness. $\mathcal{CM}_{com}^{-}$ contains those features mappable into $\mathcal{CFDI}_{nc}^{\forall -}$ (as described in Section 3.3) and captures a subset of features of ORM, named ORM2$_{cfd}$. Thus, within the scope of this paper, one can equate $\mathcal{CM}_{com}^{-}$ and ORM2$_{cfd}$.

3.1 Syntax

The syntax is introduced first, and subsequently illustrated with an example. We assume a transformation where an ORM value type is encoded in $\mathcal{CM}_{com}^{-}$ as an attribute, and note that recursive relationships are allowed such that a class can participate more than once in a relationship.

Definition 2 (Conceptual Data Model $\mathcal{CM}_{com}^{-}$ syntax) A $\mathcal{CM}_{com}^{-}$ conceptual data model is a tuple $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}_R, \text{CARD}_A, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{DISJ}_C, \text{DISJ}_R, \text{ID}, \text{EXTID}, \text{FD}, \text{OBJ}, \text{REX})$ such that:

1. $\mathcal{L}$ is a finite alphabet partitioned into the sets: $\mathcal{C}$ (*object type* symbols), $\mathcal{A}$ (*attribute* symbols), $\mathcal{R}$ (*relationship* symbols), $\mathcal{U}$ (*role* symbols), and $\mathcal{D}$ (*domain* symbols); the tuple $(\mathcal{C}, \mathcal{A}, \mathcal{R}, \mathcal{U}, \mathcal{D})$ is the *signature* of Σ.
2. ATT is a function that maps a class symbol in $\mathcal{C}$ to an $\mathcal{A}$-labeled tuple over $\mathcal{D}$, so that $\text{ATT}(C) = \{A_1 : D_1, \ldots, A_h : D_h\}$ where h is a non-negative integer.
3. REL is a function that maps a relationship symbol in $\mathcal{R}$ to an $\mathcal{U}$-labeled tuple over $\mathcal{C}$, $\text{REL}(R) = \{U_i : C_i\}_{i=1}^{h}, h \geq 1$, $U_i \neq U_j$ if $i \neq j$, h is called the *arity* of R, and if $(U_i, C_i) \in \text{REL}(R)$ then $\text{PLAYER}(R, U_i) = C_i$ and $\text{ROLE}(R, C_i) = U_i$. The signature of the relation is $\sigma_R = \{U_i\}_{i=1}^{n}$.

4. CARD_R is a partial function $\text{CARD}_R : \mathcal{R} \times \mathcal{U} \to \{0,1\} \times \{1,\infty\}$ denoting cardinality constraints. We denote with $\text{CMIN}(R,U)$ and $\text{CMAX}(R,U)$ the first and second component of CARD_R.
5. CARD_A is a partial function $\text{CARD}_A : \mathcal{C} \times \mathcal{A} \to \{0,1\} \times \{1,\infty\}$ denoting multiplicity constraints for attributes, such that $\text{CARD}_A(C,A)$ is defined iff $(A,D) \in \text{ATT}(C)$ for some $D \in \mathcal{D}$. We denote with $\text{CMIN}(C,A)$ and $\text{CMAX}(C,A)$ the first and second component of CARD_A.
6. ISA_C is a binary relationship $\text{ISA}_C \subseteq \mathcal{C} \times \mathcal{C}$.
7. ISA_R is a binary relationship $\text{ISA}_R \subseteq \mathcal{R} \times \mathcal{R}$. ISA_R between relationships is restricted to relationships with compatible signatures.
8. ISA_U is a binary relationship $\text{ISA}_U \subseteq \mathcal{U} \times \mathcal{U}$.
9. DISJ_C is a relationship over $2^{\mathcal{C}} \times \mathcal{C}$, describing disjointness partitions over groups of ISA that share the same superclass.
10. DISJ_R is a subset of $2^{\mathcal{R}}$, describing disjointness over a group of relations (with compatible signatures).
11. ID is a partial function, $\text{ID} : \mathcal{C} \to 2^{\mathcal{A}}$, that maps a class symbol in $\mathcal{C}$ to its identifier (key) attributes; $\text{CARD}_A(C,A) = (1,1)$ holds for each $A \in \text{ID}(C)$.
12. EXTID is called an external uniqueness, which is a partial function that maps a class to a set of sets of relation-role pairs or attributes, $\text{EXTID} : \mathcal{C} \to 2^{2^{(\mathcal{R}\times\mathcal{U})\cup\mathcal{A}}}$, where for any $S \in \text{EXTID}(C)$ it holds that at least one pair $(R,U) \in S$, and whenever $(R,U) \in S$ then $\text{PLAYER}(R,U) = C$, $\text{CARD}_R(R,U) = (1,1)$, and when $A \in S$ then $\text{CARD}_A(C,A) = (1,1)$.
13. FD is a functional dependency assertion, a partial function $\text{FD} : \mathcal{R} \to 2^{2^{\mathcal{U}}} \times \mathcal{U}$ such that $\text{FD}(R)$ may be defined only if its arity is ≥ 2 and for all $U_i, U \in \mathcal{U}$ such that $(\{U_i\}, U) \in \text{FD}(R)$ then exist C_i, C such that $U_i : C_i \in \text{REL}(R)$ and $U : C \in \text{REL}(R)$.
14. OBJ is an objectification partial function that maps an n-ary relationship symbol $R \in \mathcal{R}$ to a $C \in \mathcal{C}$, i.e., $\text{OBJ} : \mathcal{R} \to \mathcal{C}$. Whenever $\text{OBJ}(R) = C$ and $U_i : C_i \in \text{REL}(R), 1 \leq i \leq n$ then there exist n new binary relationships $R_i \in \mathcal{R}$ such that $\text{REL}(R_i) = \{U_i^1 : C, U_i^2 : C_i\}$, with $U_i^1, U_i^2 \in \mathcal{U}$; $\text{EXTID}(C) = \{(U_i, C_i) \mid 0 < i \leq n\}$; $\text{CARD}_R(R_i, U_i^1) = (1,1)$ and $\text{CARD}_R(R_i, U_i^2) = (0,1)$.
15. REX is a subset of $2^{\mathcal{U}}$ describing disjointness partitions over a group of roles, i.e, if $\{U_i\}_{i=1}^h \in \text{REX}$ then exist $R_i \in \mathcal{R}, C_i \in \mathcal{C}$ such $U_i : C_i \in \text{REL}(R_i)$, and all R_i have the same arity.

To link this syntax to ORM's icons, value types are transformed into attributes, any unary relationship is translated into a class and a binary relationship with a Boolean datatype, and a suitable naming scheme for the roles and fact types is in place.

Example 1 Let us consider a mapping between this $\mathcal{CM}^-_{com}$ syntax and some of the mappable ORM icons of the ORM2$_{cfd}$ fragment, using the diagram in Fig. 1.
The fact type verbalised with works for can be represented with the relationship
$\text{REL}(\texttt{works}) = \{\texttt{DE} : \texttt{Department}, \texttt{ED} : \texttt{Employee}\}$
Identification (single attribute, resp. external) of the Employee and Room with
$\text{ID}(\texttt{Employee}) = \{\texttt{empNr}\}$ and $\text{EXTID}(\texttt{Room}) = \{\{\texttt{locBuildingName}, \texttt{roomNr}\}\}$.
Subsumption of ORM entity types and fact types is straightforward as:

$\texttt{Academic}\ \text{ISA}_{\texttt{C}}\ \texttt{Employee}$ and $\texttt{supports}\ \text{ISA}_{\texttt{R}}\ \texttt{works}$
and mandatory participation of Department in works as
$\text{CARD}_{\texttt{R}}(\texttt{works}, \texttt{DE}) = (1, \infty)$. ♢

3.2 Semantics

The model-theoretic semantics of $\mathcal{CM}^{-}_{com}$ in the light of ORM2$_{cfd}$ is as follows.

Definition 3 ($\mathcal{CM}^{-}_{com}$ Semantics) Let Σ be a $\mathcal{CM}^{-}_{com}$ conceptual data model. An *interpretation* for the conceptual model Σ is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{I}}_{\mathcal{D}}, \cdot^{\mathcal{I}})$, such that:

- $\Delta^{\mathcal{I}}$ is a nonempty set of abstract objects disjoint from $\Delta^{\mathcal{I}}_{\mathcal{D}}$;
- $\Delta^{\mathcal{I}}_{\mathcal{D}} = \bigcup_{D_i \in \mathcal{D}} \Delta_{D_i}$ is the set of basic domain values used in Σ; and
- $\cdot^{\mathcal{I}}$ is a function that maps:
  - Every basic domain symbol $D \in \mathcal{D}$ into a set $D^{\mathcal{I}} = \Delta_D$.
  - Every class $C \in \mathcal{C}$ to a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.
  - Every attribute $A \in \mathcal{A}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}_{\mathcal{D}}$, such that for each $A \in \mathcal{A}$ and $C \in \mathcal{C}$ with $A : D \in \text{ATT}(C)$ for some $D \in \mathcal{D}$, then for all $o \in C^{\mathcal{I}}$ there exists $d \in D^{\mathcal{I}}$ such that $(o, d) \in A^{\mathcal{I}}$.
  - Every relationship $R \in \mathcal{R}$ to a set $R^{\mathcal{I}}$ of tuples of $\mathcal{U}$-labeled elements of $\Delta^{\mathcal{I}}$: for R an n-ary relationship connecting the classes $C_1, \ldots, C_n$, $\text{REL}(R) = \{U_i : C_i\}_{i=1}^{n}$, i.e., $\{U_i : o_i\}_{i=1}^{n} \in R^{\mathcal{I}}$ implies $o_i \in C_i^{\mathcal{I}}$.

$\mathcal{I}$ is called a *legal database state* or *legal application software state* if it satisfies all of the constraints expressed in the conceptual data model:

- for each $C_1, C_2 \in \mathcal{C}$: if $C_1\ \text{ISA}_C\ C_2$, then $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- for each $R_1, R_2 \in \mathcal{R}$ (with the same signature): if $R_1\ \text{ISA}_R\ R_2$, then $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$. In addition, ORM stipulates that the participating role sequences (which in this case are all roles of R_i) of every relationship participating in the $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$ to be *role-sequence compatible*.[6]
- for each $U_1, U_2 \in \mathcal{U}$: if $U_1\ \text{ISA}_U\ U_2$, then there exist $R_1, R_2 \in \mathcal{R}$ with $\text{PLAYER}(R_1, U_1) = C_1$ and $\text{PLAYER}(R_2, U_2) = C_1$, and $C_1, C_2 \in \mathcal{C}$, and $\{o \in C_1^{\mathcal{I}}$ such that $U_1 : o \in R_1^{\mathcal{I}}\} \subseteq \{o \in C_2^{\mathcal{I}}$ such that $U_2 : o \in R_2^{\mathcal{I}}\}$.
- for each $C \in \mathcal{C}$, $R \in \mathcal{R}$, $U \in \mathcal{U}$: if $\text{CARD}_R(R, U) = (x, y)$ and $\text{PLAYER}(R, U) = C$, then $x \leq \#\{o \in C^{\mathcal{I}}$ such that $U : o \in R^{\mathcal{I}}\} \leq y$.
- for each $C \in \mathcal{C}$, $A \in \mathcal{A}$: if $\text{CARD}_A(C, A) = (x, y)$, then $A : D \in \text{ATT}(C)$ for some $D \in \mathcal{D}$, and $x \leq \#\{d \in D^{\mathcal{I}}$ such that $(o, d) \in A^{\mathcal{I}}, o \in C^{\mathcal{I}}\} \leq y$.
- if $\{C_i\}_{i=1}^{n} \in \text{DISJ}_C$ then $C_i^{\mathcal{I}} \neq C_j^{\mathcal{I}}$ for all $i, j, j \neq i, 1 \leq i, j \leq n$.
- if $\{R_i\}_{i=1}^{n} \in \text{DISJ}_R$ then R_i are role-sequence compatible and $R_j^{\mathcal{I}} \cap R_k^{\mathcal{I}} = \emptyset$.
- for each $C \in \mathcal{C}, A_i \in \mathcal{A}, 1 \leq i \leq n$: if $\text{ID}(C) = \{A_i\}$, then exists $D_i \in \mathcal{D}$ such that $A_i : D_i \in \text{ATT}(C)$, and $\#\{o \in C^{\mathcal{I}}$ such that $\bigwedge_{i=1}^{n}(o, d_i) \in A_i^{\mathcal{I}}\} \leq 1$ for any $d_i \in D_i^{\mathcal{I}}, 1 \leq i \leq n$.
- for each $C \in \mathcal{C}$, $R_i \in \mathcal{R}$, $U_i \in \mathcal{U}, 1 \leq i \leq h$, $A_j, 1 \leq j \leq l$: if $\{(R_i, U_i)\}_{i=1}^{h} \cup \{A_j\}_{j=1}^{l} \in \text{EXTID}(C)$, then exist $D_j \in \mathcal{D}$ with $A_j : D_j \in \text{ATT}(C)$, and $\#\{o \in C^{\mathcal{I}}$ such that $(U_i : o) \in R_i^{\mathcal{I}}$ and $(o, d_j) \in A_j^{\mathcal{I}}\} \leq 1$ for any $d_j \in D_j^{\mathcal{I}}, 1 \leq j \leq l$.

[6] We assume that the compatibility is enforced *explicitly* by additional ISA_C pairs of the classes linked to the matching roles in the relationships, e.g., for $U : C \in \text{REL}(R_1)$ and $U : D \in \text{REL}(R_2)$ we have $C\ \text{ISA}_C\ D$.

– for each $U, U_i \in \mathcal{U}, 1 \leq i \leq h$, $R \in \mathcal{R}$: if $\text{FD}(R) = (\{U_i\}_{i=1}^h, U)$, then exists $C, C_i \in \mathcal{C}$ such that $C : U \in \text{REL}(R)$, $C_i : U_i \in \text{REL}(R)$ and $\#\{o \in C^{\mathcal{I}}$ such that $\{U : o\} \cup \{U_i : o_i, \} \in R^{\mathcal{I}}\} \leq 1$ for any $o_i \in C_i^{\mathcal{I}}, 1 \leq i \leq h$.
– for each $R \in \mathcal{R}$, $C \in \mathcal{C}$: if $\text{OBJ}(R) = C$, then exist $R_i \in \mathcal{R}$ with $\text{REL}(R_i) = \{U_i^1 : C, U_i^2 : C_i\}$, for all $1 \leq i \leq n$, and the following statements are also satisfied: $\text{EXTID}(C) = \{(R_i, U_i^2) \mid 0 < i \leq n\}$, $\text{CARD}_R(R_i, U_i^1) = (1, 1)$, and $\text{CARD}_R(R_i, U_i^2) = (0, 1)$.
– for each $U_i \in \mathcal{U}, 1 \leq i \leq h$: if $\{U_i\}_{i=1}^h \in \text{REX}$, then exists $R_i \in \mathcal{R}$ with arity m, such that $U_i : C_i \in \text{REL}(R_i)$. Then for each $o \in C_i^{\mathcal{I}}$ the following condition is true $\#\{k$ such that $\{U_k : o\} \in R_k^{\mathcal{I}}, 1 \leq k \leq h\} \leq 1$.

Σ is said to be *globally consistent* if it admits at least one legal state.

3.3 Mapping of ORM2$_{cfd}$ to $\mathcal{CFDI}_{nc}^{\forall -}$

We formalize how $\mathcal{CFDI}_{nc}^{\forall -}$ can capture ORM2$_{cfd}$ conceptual models. Let $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}_R, \text{CARD}_A, \text{ISA}_C, \text{ISA}_R, \text{ISA}_U, \text{DISJ}_C, \text{DISJ}_R, \text{ID}, \text{EXTID}, \text{FD}, \text{OBJ}, \text{REX})$ be a $\mathcal{CM}_{com}^-$ conceptual data model corresponding to ORM2$_{cfd}$. We map Σ to a $\mathcal{CFDI}_{nc}^{\forall -}$ TBox $\mathcal{T}_\Sigma$ in the vocabulary PC, F using the following rules:

– Include in the vocabulary one concept name for each ORM2$_{cfd}$ object type and datatype, i.e., for each $C \in \mathcal{C}$, $D \in \mathcal{D}$ we have $\mathrm{C} \in \mathsf{PC}$, $\mathrm{D} \in \mathsf{PC}$.
– To map attributes, there are two cases to consider: if the attribute is functional then it is mapped as a function symbol and two concepts that reify each of the roles; otherwise it is reified as a new concept, with the corresponding cardinality constraints. For each $C \in \mathcal{C}$ such that $\text{ATT}(C) = \{A_i : D_i\}_{i=1}^n$, and for each i:
  • if $\text{CARD}_A(C, A_i) = (1, 1)$, then we introduce two new concepts $\mathrm{U}_1^{\mathrm{A_i}}, \mathrm{U}_2^{\mathrm{A_i}} \in \mathsf{PC}$, a new function symbol $\mathrm{a_i} \in \mathsf{F}$, and $\{\mathrm{C} \sqsubseteq \forall \mathrm{a_i}.\mathrm{D_i}, \mathrm{C} \sqsubseteq \mathrm{U}_1^{\mathrm{A_i}}, \mathrm{U}_1^{\mathrm{A_i}} \sqsubseteq \mathrm{C}, \mathrm{U}_2^{\mathrm{A_i}} \sqsubseteq \exists \mathrm{a_i}^{-1}, \exists \mathrm{a_i}^{-1} \sqsubseteq \mathrm{U}_2^{\mathrm{A_i}}\} \in \mathcal{T}_\Sigma$.
  • otherwise we introduce new concept symbols $\mathrm{A_i}, \mathrm{U}_{\mathrm{i},1}^{\mathrm{A}}, \mathrm{U}_{\mathrm{i},2}^{\mathrm{A}} \in \mathsf{PC}$, and two new function symbols $\mathrm{a}_{\mathrm{i},1}, \mathrm{a}_{\mathrm{i},2} \in \mathsf{F}$, with $\{\mathrm{A_i} \sqsubseteq \forall \mathrm{a}_{\mathrm{i},1}.\mathrm{U}_{\mathrm{i},1}^{\mathrm{A}}, \mathrm{A_i} \sqsubseteq \forall \mathrm{a}_{\mathrm{i},2}.\mathrm{U}_{\mathrm{i},2}^{\mathrm{A}}, \mathrm{U}_{\mathrm{i},1}^{\mathrm{A}} \sqsubseteq \mathrm{C}, \mathrm{U}_{\mathrm{i},2}^{\mathrm{A}} \sqsubseteq \mathrm{D_i}, \mathrm{U}_{\mathrm{i},1}^{\mathrm{A}} \sqsubseteq \exists \mathrm{a}_{\mathrm{i},1}{}^{-1}, \mathrm{U}_{\mathrm{i},2}^{\mathrm{A}} \sqsubseteq \exists \mathrm{a}_{\mathrm{i},2}{}^{-1}, \forall \mathrm{a}_{\mathrm{i},1}.\mathrm{U}_{\mathrm{i},1}^{\mathrm{A}} \sqsubseteq \mathrm{A_i}, \forall \mathrm{a}_{\mathrm{i},2}.\mathrm{U}_{\mathrm{i},2}^{\mathrm{A}} \sqsubseteq \mathrm{A_i}, \mathrm{A_i} \sqsubseteq \mathrm{A_i} : \mathrm{a}_{\mathrm{i},1}, \mathrm{a}_{\mathrm{i},2} \rightarrow id\} \subseteq \mathcal{T}_\Sigma$. If $\text{CMIN}(C, A_i) = 1$, then also $\mathrm{C} \sqsubseteq \mathrm{U}_{\mathrm{i},1}^{\mathrm{A}} \in \mathcal{T}_\Sigma$; if if $\text{CMAX}(C, A_i) = 1$, then $\mathrm{A_i} \sqsubseteq \mathrm{A_i} : \mathrm{a}_{\mathrm{i},1} \rightarrow id \in \mathcal{T}_\Sigma$.
– The mapping of relationships (ORM2$_{cfd}$ fact types) is similar as the mapping of attributes. If the relationship is binary and one of its roles has a $(1, 1)$ constraint, then it is mapped as an attribute; in the other cases the relationship and its roles are reified as a new concepts, with new attributes from the reified relationship to the reified roles. The reified roles are then subconcepts of the concepts originally participating in the relationship. For each $R \in \mathcal{R}$ such that $\text{REL}(R) = \{U_i : C_i\}_{i=1}^n$ then
  • if $n = 2$ and $\text{CARD}_R(R, U_1) = (1, 1)$, then we introduce a new function symbol $\mathrm{u_1} \in \mathsf{F}$, and $\{\mathrm{C_1} \sqsubseteq \forall \mathrm{u_1}.\mathrm{C_2}, \mathrm{C_1} \sqsubseteq \mathrm{C_1} : \mathrm{u_1} \rightarrow id\} \in \mathcal{T}_\Sigma$. Similarly if $\text{CARD}_R(R, U_2) = (1, 1)$ then we introduce function symbol $\mathrm{u_2} \in \mathsf{F}$ and $\{\mathrm{C_2} \sqsubseteq \forall \mathrm{u_2}.\mathrm{C_1}, \mathrm{C_2} \sqsubseteq \mathrm{C_2} : \mathrm{u_2} \rightarrow id\} \in \mathcal{T}_\Sigma$.

- otherwise we add new concept symbols $\mathrm{R}, \mathrm{U_i^R} \in \mathsf{PC}, 1 \leq i \leq n$, new function symbols $\mathrm{u_i^R} \in \mathsf{F}, 1 \leq i \leq n$, and then $\{\mathrm{R} \sqsubseteq \forall \mathrm{u_i^R}.\mathrm{U_i^R}, \mathrm{U_i^R} \sqsubseteq \mathrm{C_i}, \mathrm{U_i^R} \sqsubseteq \exists \mathrm{u_i^R}^{-1}, \forall \mathrm{u_i^R}.\mathrm{U_i^R} \sqsubseteq \mathrm{R}\}_{i=1}^{n} \cup \{\mathrm{R} : \mathrm{u_1^R}, \ldots, \mathrm{u_n^R} \rightarrow id\} \subseteq \mathcal{T}_\Sigma$. Additionally for each $i, 1 \leq i \leq n$ if $\textsc{cmin}(R, U_i) = 1$ then also $\mathrm{C_i} \sqsubseteq \mathrm{U_i^R} \in \mathcal{T}_\Sigma$; and if $\textsc{cmax}(R, U_i) = 1$, then $\mathrm{R} \sqsubseteq \mathrm{R} : \mathrm{u_i^R} \rightarrow id \in \mathcal{T}_\Sigma$.

– for each $C_1, C_2 \in \mathcal{C}$ such that C_1 $\textsc{isa}_C$ C_2, then $\mathrm{C_1} \sqsubseteq \mathrm{C_2} \in \mathcal{T}_\Sigma$.
– for each $R_1, R_2 \in \mathcal{R}$ such that R_1 $\textsc{isa}_R$ R_2, then $\mathrm{R_1} \sqsubseteq \mathrm{R_2} \in \mathcal{T}_\Sigma$. In order to define relationship inheritance in ORM2$_{cfd}$, the types of the participating concepts must be compatible, therewith adhering to the syntax restriction of ORM (as aside: without this condition, the reconstruction in a PTIME language is not possible).
– for each $U_1, U_2 \in \mathcal{U}$ such that U_1 $\textsc{isa}_U$ U_2 then $\mathrm{U_1} \sqsubseteq \mathrm{U_2} \in \mathcal{T}_\Sigma$.
– for each $C_i, C \in \mathcal{C}$ such that $\{C_i\}_{i=1}^{n}$ $\textsc{disj}_C$ C, then $\{\mathrm{C_i} \sqsubseteq \mathrm{C}\}_{i=1}^{n} \cup \{\mathrm{C_i} \sqsubseteq \neg \mathrm{C_j}\}_{i \neq j, i, j=1}^{n} \subseteq \mathcal{T}_\Sigma$, stating the concepts are pairwise disjoint.
– for each $R_i \in \mathcal{R}$ such that $\{R_i\}_{i=1}^{n} \in \textsc{disj}_R$, then $\{\mathrm{R_i} \sqsubseteq \neg \mathrm{R_j}, \mathrm{R_i} \sqsubseteq \mathrm{R_j} : \sigma_{R_j} \rightarrow id\}_{i \neq j, i, j=1}^{n} \subseteq \mathcal{T}_\Sigma$. Again, ORM has the condition that relationship exclusion must be defined over compatible types for the participating concepts (which also happens to be a necessary condition for the efficiency of the translation).
– for each $C \in \mathcal{C}$, $A_i \in \mathcal{A}$ such that $\textsc{id}(C) = \{A_i\}_{i=1}^{n}$, then $\mathrm{C} \sqsubseteq \mathrm{C} : \mathrm{a_1}, \ldots, \mathrm{a_n} \rightarrow id \in \mathcal{T}_\Sigma$. In this case since the attributes are keys then they must be in $(1,1)$ constraint with C, so they are mapped as features in $\mathcal{T}_\Sigma$ by the first point of the respective rule, described above.
– for each $C \in \mathcal{C}$, $R_i \in \mathcal{R}$, $U_i \in \mathcal{U}$, $A_j \in \mathcal{A}$ such that $\{(R_i, U_i)\}_{i=1}^{h} \cup \{A_j\}_{j=1}^{k} \in \textsc{extid}(C)$, then $\mathrm{C} \sqsubseteq \mathrm{C} : \mathrm{u_1^R}, \ldots, \mathrm{u_h^R}, \mathrm{a_1}, \ldots, \mathrm{a_k} \rightarrow id \in \mathcal{T}_\Sigma$. Here the only possibility is that U_i and A_j belonging to the external identifier have a $(1,1)$ constraint, so they are mapped with features.
– for each $U, U_i \in \mathcal{U}$, $R \in \mathcal{R}$: if $\textsc{fd}(R) = (\{U_i\}_{i=1}^{h}, U)$, then $\mathrm{C_R} \sqsubseteq \mathrm{C_R} : \mathrm{u_1^R}, \ldots, \mathrm{u_h^R} \rightarrow \mathrm{u^R}$. This case is similar as the previous one: we ensure the roles and attributes belonging to the external identifier are mapped as features in $\mathcal{T}_\Sigma$ because the are $(1,1)$ to C.
– for each $R \in \mathcal{R}$, $C \in \mathcal{C}$: if $\textsc{obj}(R) = C$, then we have the mappings for $\textsc{extid}(C)$, $\textsc{card}_R(R_i, U_i^1) = (1,1)$, and $\textsc{card}_R(R_i, U_i^2) = (0,1)$.
– for each $U_i \in \mathcal{U}$: if $\{U_i\}_{i=1}^{h} \in \textsc{rex}$, then exists $R_i \in \mathcal{R}, C_i \in \mathcal{C}$ with arity m, such that $(U_i : C_i) \in \textsc{rel}(R_i), 1 \leq i \leq h$. Then $\{\mathrm{U_i^R} \sqsubseteq \neg \mathrm{U_j^R}\}_{i \neq j, i, j=1}^{h} \subseteq \mathcal{T}_\Sigma$. Note that REX requires optional participation in the role, and therewith uses the second case of the relationship mapping above.

Case analysis of the translation combined with Proposition 1 yields the following:

Theorem 1. *Let Σ be an* ORM2$_{cfd}$ *conceptual data model. A class C is consistent in Σ if and only if the knowledge base $(\mathcal{T}_\Sigma, \{C(a_C)\})$ is consistent. Σ is globally consistent if and only if $\mathcal{T}_\Sigma$ is consistent with $\mathcal{A} = \{C(a_C) \mid C \in \mathcal{C}\}$.*

Example 2 Consider again the running example with Fig. 1 and the illustration of the syntax (Example 1), the corresponding $\mathcal{CFDI}_{nc}^{\forall -}$ knowledge base contains, among others: the translation where `empNr` is an attribute of `Employee` with

$\text{CARD}_{\texttt{A}}(\texttt{empNr}, \texttt{Employee}) = (1, 1)$, the following $\mathcal{CFDI}^{\forall-}_{nc}$ axioms:

$$\{\text{Employee} \sqsubseteq \forall \text{empNr}.\text{Integer}, \text{Employee} \sqsubseteq \text{U}_1^{\text{empNr}}, \text{U}_1^{\text{empNr}} \sqsubseteq \text{Employee}, \\ \text{U}_2^{\text{empNr}} \sqsubseteq \exists \text{empNr}^{-1}, \exists \text{empNr}^{-1} \sqsubseteq \text{U}_2^{\text{empNr}}\}.$$

Then, to represent the identifier, we add $\text{Employee} \sqsubseteq \text{Employee} : \text{empNr} \rightarrow id$. The mapping of $\text{REL}(\texttt{works})$ with cardinalities $\text{CARD}_{\texttt{R}}(\texttt{works}, \texttt{DE}) = (1, \infty)$ and $\text{CARD}_{\texttt{R}}(\texttt{works}, \texttt{ED}) = (0, \infty)$ is the set of axioms

$$\{\ \text{Works} \sqsubseteq \forall \text{de}^{\text{works}}.\text{DE}^{\text{works}}, \text{DE}^{\text{works}} \sqsubseteq \text{Department}, \text{DE}^{\text{works}} \sqsubseteq \exists \text{de}^{\text{works}\,-1}, \\ \forall \text{de}^{\text{works}}.\text{DE}^{\text{works}} \sqsubseteq \text{Works}, \text{Works} \sqsubseteq \text{Works} : \text{de}^{\text{works}}, \text{ed}^{\text{works}} \rightarrow id, \\ \text{Department} \sqsubseteq \text{DE}^{\text{works}}, \text{Works} \sqsubseteq \forall \text{ed}^{\text{works}}.\text{ED}^{\text{works}}, \text{ED}^{\text{works}} \sqsubseteq \text{Employee}, \\ \text{ED}^{\text{works}} \sqsubseteq \exists \text{ed}^{\text{works}\,-1}, \forall \text{ed}^{\text{works}}.\text{ED}^{\text{works}} \sqsubseteq \text{Works} \qquad \},$$

and likewise for the remainder of the diagram in Figure 1. ♢

4 Discussion

There are many papers with logic-based reconstructions of ORM, EER, and UML; we discuss a subset relevant to the scope of this paper.

Comparison with other ORM2 Encodings. Fairly expressive logic-based reconstructions of ORM fragments exist, including $ORM2^{zero}$ in the EXPTIME-complete $\mathcal{ALCQI}$ [20], ORM2$^-$ [30] in the EXPTIME-complete $\mathcal{DLR}_{\mathit{ifd}}$ [11], an ORM2 fragment (e.g., [41]) in $\mathcal{SROIQ}$ that is N2EXPTIME-Complete [27], and ORM in the undecidable FOL [21]. An Alloy encoding and a numeric model as encoding for ORM are proposed in [23], which are experimentally compared to unsatisfiability pattern checks, showing that the latter two far outperform the Alloy approach (seconds vs. hours and timeouts), but complexity results are not provided. Their ORM fragment of the number encoding does include 'costly' features, such as covering constraints, disjunctive mandatory, arbitrary frequency (with uniqueness check), external identifiers, and value constraints, but it is unclear what was used in the test ORM diagrams. The only ORM fragment claimed in PTIME is Smaragdakis et al.'s ORM$^-$ [35] that also uses a number model. It includes non-overlapping uniqueness constraints over n-ary relationships, simple mandatory, non-overlapping frequency constraints (cardinalities > 1), value constraints, and subtype constraints. Arbitrary frequency constraints (like arbitrary projections in a relational table) cause undecidability, but, though not specified in [35], one could assume it always occurs in conjunction with a suitable uniqueness constraint in order to regain decidability, as discussed in [23, 29]. Their value constraints are not constrained either, i.e., value ranges of integers, floats, and enumerations are allowed, and have no constraints, such as so-called "safe" data types [3]. Most problematic, however, is the integer bound propagation in step 2 of Algorithm 2 in [35], which has recently shown to be NP-Complete [7], hence, Smaragdakis et al.'s solution seems to be at least NP-Hard. To the best of our knowledge, the here provided logic-based reconstruction of the ORM2$_{\mathit{cfd}}$ ORM fragment in the PTIME $\mathcal{CFDI}^{\forall-}_{nc}$ is the first tractable encoding of ORM, yet still capturing most of the entities encountered in extant ORM models.

Extensibility to EER and UML Class Diagrams. As ORM is more expressive than UML Class Diagrams and EER, and an ORM diagram can be translated into UML and into ER [22], the results obtained should be at least as good for those. A quick matching thanks to availing of the unifying metamodel [31, 18] reveals that that is the case, where $\mathcal{CFDI}_{nc}^{\forall-}$ can encode over 97% of the 34 UML models and over 99% of the 34 (E)ER models in our dataset. This can be of use here as well, as also most UML Class Diagram encodings focus on expressiveness, using, among others, $\mathcal{DLR}_{\mathit{ifd}}$ [5] as well, or an OCL-lite encoding matching $\mathcal{ALCI}$ [34], which is still EXPTIME-complete. Kaneiwa and Satoh claim to have some fragments of UML Class Diagrams in P and PSpace for full satisfiability checking (all classes must be satisfiable) [25, 26], but this has been proven otherwise by Artale et al. [1]. Focusing on coverage of features, the smallest restricted fragment is shown to be NLogSpace [1] when disallowing ISA on associations and completeness on subclasses, using approximations of reified binaries (i.e., missing EXTID, and thus also no qualified associations). An initial analysis shows that this might still capture almost 96% of the UML models in our dataset, and might thus also be a worthwhile fragment of UML. Such high coverage can be obtained partially due to the changes to UML v2.4.1 [33] where relational properties (asymmetry and transitivity) for aggregation have been dropped, with aggregation taking up about a quarter of all associations, and not all UML features in the standard are implemented in modelling tools.

The story is similar for (E)ER. Various encodings exist [14, 37, 38], (partially due to the absence of a standard), which either use a language in the EXPTIME-complete $\mathcal{DLR}$ family [9–11] or *DL-Lite* family [12] with different computational complexities for different EER fragments [2]. Trading functionality for gaining a little in computational complexity [2], however, is certainly not an option if coverage is also an aim, especially due to all the identifiers in EER and ORM (which can be represented in $\mathcal{CFDI}_{nc}^{\forall-}$).

5 Conclusions

A logic-based reconstruction for most of ORM using the PTIME Description Logic language $\mathcal{CFDI}_{nc}^{\forall-}$ has been presented, covering 96.5% of a set of extant ORM models. This is the first tractable encoding of ORM, which includes features important for conceptual models, notably n-ary relationships and complex identification constraints. Future work includes working toward implementations of the scenarios alluded to in the introduction, and we also expect to apply this encoding to facilitate inter-model interoperability [17] as the results are easily transferable to EER and UML Class diagrams.

Acknowledgments. This work is based upon research supported by the National Research Foundation of South Africa (Project UID: 90041), the Argentinian Ministry of Science and Technology (PRF and CMK), and NSERC (DT).

References

1. Artale, A., Calvanese, D., Ibáñez-García, Y.: Full satisfiability of UML class diagrams. In: Parsons, J., et al. (eds.) Proceedings of the 29th International Conference on Conceptual Modeling (ER'10). LNCS, vol. 6412, pp. 317–331. Springer (2010)
2. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Reasoning over extended ER models. In: Parent, C., Schewe, K.D., Storey, V.C., Thalheim, B. (eds.) Proceedings of the 26th International Conference on Conceptual Modeling (ER'07). LNCS, vol. 4801, pp. 277–292. Springer (2007), auckland, New Zealand, November 5-9, 2007
3. Artale, A., Ryzhikov, V., Kontchakov, R.: DL-Lite with attributes and datatypes. In: Proceeding of the 20th European Conference on Artificial Intelligence (ECAI'12). pp. 61–66. IOS Press (2012)
4. Bakema, G., Zwart, J.P., van der Lek, H.: Volledig Communicatiegeoriënteerde Informatiemodellering FCO-IM. Academic Service (2005)
5. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
6. Bloesch, A.C., Halpin, T.A.: Conceptual Queries using ConQuer-II. In: Proceedings of ER'97: 16th International Conference on Conceptual Modeling. LNCS, vol. 1331, pp. 113–126. Springer (1997)
7. Bordeaux, L., Katsirelos, G., Narodytska, N., Vardi, M.Y.: The complexity of integer bound propagation. Journal of Artificial Intelligence Research 40, 657–676 (2011)
8. Braga, B.F.B., Almeida, J.P.A., Guizzardi, G., Benevides, A.B.: Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal methods. Innovations in Systems and Software Engineering 6(1-2), 55–63 (2010)
9. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98). pp. 149–158 (1998)
10. Calvanese, D., De Giacomo, G., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99). pp. 84–89 (1999)
11. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: Nebel, B. (ed.) Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001). pp. 155–160. Morgan Kaufmann (2001), seattle, Washington, USA, August 4-10, 2001
12. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
13. Calvanese, D., Keet, C.M., Nutt, W., Rodríguez-Muro, M., Stefanoni, G.: Web-based graphical querying of databases through an ontology: the WONDER system. In: Shin, S.Y., Ossowski, S., Schumacher, M., Palakal, M.J., Hung, C.C. (eds.) Proceedings of ACM Symposium on Applied Computing (ACM SAC'10). pp. 1389–1396. ACM (2010), 22-26 March, 2010, Sierre, Switzerland
14. Chen, P.P.: The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
15. Curland, M., Halpin, T.: Model driven development with NORMA. In: Proceedings of the 40th International Conference on System Sciences (HICSS-40). pp. 286a–286a. IEEE Computer Society (2007), los Alamitos, Hawaii
16. Farré, C., Queralt, A., Rull, G., Teniente, E., Urpí, T.: Automated reasoning on UML conceptual schemas with derived information and queries. Information and Software Technology 55(9), 1529 – 1550 (2013)

17. Fillottrani, P.R., Keet, C.M.: Conceptual model interoperability: a metamodel-driven approach. In: Bikakis, A., et al. (eds.) Proceedings of the 8th International Web Rule Symposium (RuleML'14). LNCS, vol. 8620, pp. 52–66. Springer (2014), august 18-20, 2014, Prague, Czech Republic
18. Fillottrani, P.R., Keet, C.M.: KF metamodel formalisation. Technical report 1412.6545v1 (September 2014), arxiv.org
19. Fillottrani, P.R., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web Journal 3(3), 293–306 (2012)
20. Franconi, E., Mosca, A.: The formalisation of ORM2 and its encoding in OWL2. Technical Report KRDB12-2, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy (March 2012), http://www.inf.unibz.it/krdb/pub/TR/KRDB12-2.pdf
21. Halpin, T.: A logical analysis of information systems: static aspects of the data-oriented perspective. Ph.D. thesis, University of Queensland, Australia (1989)
22. Halpin, T., Morgan, T.: Information modeling and relational databases. Morgan Kaufmann, 2nd edn. (2008)
23. Jahangard-Rafsanjani, A., Mirian-Hosseinabadi, S.H.: Lightweight formalization and validation of ORM models. Journal of Logical and Algebraic Methods in Programming p. http://dx.doi.org/10.1016/j.jlamp.2015.03.001 (2015)
24. Jahangard Rafsanjani, A., Mirian-Hosseinabadi, S.H.: A Z approach to formalization and validation of ORM models. In: Ariwa, E., El-Qawasmeh, E. (eds.) Digital Enterprise and Information Systems, Communications in Computer and Information Science, vol. 194, pp. 513–526. Springer (2011)
25. Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted UML class diagrams. In: Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS'06). Springer Verlag (2006)
26. Kaneiwa, K., Satoh, K.: On the complexities of consistency checking for restricted UML class diagrams. Theoretical Computer Science 411, 301–323 (2010)
27. Kazakov, Y.: RIQ and SROIQ Are Harder than SHOIQ. In: Brewka, G., Lang, J. (eds.) 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08). pp. 274–284. AAAI Press (2008), 16-19 September, 2008, Sydney, Australia
28. Keet, C.M., Fillottrani, P.R.: Dataset of publicly available conceptual models and its analysis (2015), http://www.meteck.org/SAAR.html
29. Keet, C.M.: Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$. In: 20th International Workshop on Description Logics (DL'07). CEUR-WS, vol. 250, pp. 331–338 (2007), 8-10 June 2007, Bressanone, Italy
30. Keet, C.M.: Ontology-driven formal conceptual data modeling for biological data analysis. In: Elloumi, M., Zomaya, A.Y. (eds.) Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data, chap. 6, pp. 129–154. Wiley (2013)
31. Keet, C.M., Fillottrani, P.R.: Toward an ontology-driven unifying metamodel for UML class diagrams, EER, and ORM2. In: Ng, W., Storey, V.C., Trujillo, J. (eds.) 32nd International Conference on Conceptual Modeling (ER'13). LNCS, vol. 8217, pp. 313–326. Springer (2013), 11-13 November, 2013, Hong Kong
32. Object Management Group: Semantics of Business Vocabulary and Rules (SBVR) – OMG released versions of SBVR, formal/2008-01-02 (January 2008), http://www.omg.org/spec/SBVR/1.0
33. Object Management Group: Superstructure specification. Standard 2.4.1, Object Management Group (2012), http://www.omg.org/spec/UML/2.4.1/

34. Queralt, A., Artale, A., Calvanese, D., Teniente, E.: OCL-Lite: Finite reasoning on UML/OCL conceptual schemas. Data & Knowledge Engineering 73, 1–22 (2012)
35. Smaragdakis, Y., Csallner, C., Subramanian, R.: Scalable satisfiability checking and test data generation from modeling diagrams. Automation in Software Engineering 16, 73–99 (2009)
36. Solomakhin, D., Franconi, E., Mosca, A.: Logic-based reasoning support for SBVR. Fundamenta Informaticae 124(4), 543–560 (2013)
37. Song, I.Y., Chen, P.P.: Entity relationship model. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, vol. 1, pp. 1003–1009. Springer (2009)
38. Thalheim, B.: Extended entity relationship model. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, vol. 1, pp. 1083–1091. Springer (2009)
39. Toman, D., Weddell, G.E.: Fundamentals of Physical Design and Query Compilation. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2011)
40. Toman, D., Weddell, G.E.: On adding inverse features to the description logic $CFD^{\forall}{}_{\mathrm{nc}}$. In: PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. pp. 587–599 (2014)
41. Wagih, H.M., Zanfaly, D.S.E., Kouta, M.M.: Mapping Object Role Modeling 2 schemes into $\mathcal{SROIQ}(D)$ Description Logic. International Journal of Computer Theory and Engineering 5(2), 232–237 (2013)

Handling Uncertainty: An Extension of DL-Lite with Subjective Logic *

Jhonatan Garcia[1], Jeff Z. Pan[1], Achille Fokoue[2], Katia Sycara[3], Yuqing Tang[3], and Federico Cerutti[1]

[1] Computing Science, University of Aberdeen, UK
[2] IBM T. J. Watson Research Center, NY, US
[3] Carnegie Melon University, Pittsburgh, US

Abstract. Data in real world applications is often subject to some kind of uncertainty, which can be due to incompleteness, unreliability or inconsistency. This poses a great challenge for ontology-based data access (OBDA) applications, which are expected to provide a meaningful answers to queries, even under uncertain domains. Several extensions of classical OBDA systems has been proposed to address this problem, with probabilistic, possibilistic, and fuzzy OBDA being the most relevant ones. However, these extensions present some limitations with respect to their applicability. Probabilistic OBDA deal only with categorical assertions, possibilistic logic is better suited to make a ranking of axioms, and fuzzy OBDA addresses the problem of modelling vagueness, rather than uncertainty. In this paper we propose Subjective DL-Lite (SDL-Lite), an extension of DL-Lite with Subjective Logic. Subjective DL-Lite allows us to model uncertainty in the data through the application of opinions, which encapsulate our degrees of belief, disbelief and uncertainty for each given assertion. We explore the semantics of Subjective DL-Lite, clarify the main differences with respect to its classical DL-Lite counterpart, and construct a canonical model of the ontology by means of a chase that will serve as the foundation for a future construction of an OBDA system supporting opinions.

Keywords: Subjective Logic, Query Answering, OBDA, Description Logics

1 Introduction

Semantic applications that model real world scenarios often have to deal with uncertainty in the data. This is usually the case when extracting data from web sources,

* This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This work is partially supported by the EU K-Drive project.

where information might be incomplete or unreliable. Even the method used for extracting the data creates another point of uncertainty, as it is quite common to rely on heuristic algorithms that are prone to errors.

In order for a semantic application to address all these issues, such an application should be able to comply with the following list of requirements:

- Models uncertainty in the data
- Understands the meaning of the underlying data
- Provides answering services over custom user queries
- Determines the reliability of the answers given the available information

In this paper we explore some of the theoretical foundations required to develop a logic capable of supporting ontology-based data access (OBDA) applications that can fulfil these requirements.

Our contributions in this paper are twofold: A) We define the semantics for Subjective DL-Lite in section 4, and B) We present a methodology to build a chase-based canonical interpretation of subjective ontologies in section 5.

2 Related Work

Several relevant approaches to model Uncertainty have been proposed in different areas of research. Probabilistic Logic [5] extends axioms in a knowledge base with a probability value that models the degree of trust that we place on the validity of the proposition. Possibilistic Logics [9] offers a similar approach, but its possibility values express the necessity and the validity for a certain proposition, alongside with the plausibility of said proposition to be true. Fuzzy Logic [11], on the other hand, relies on a membership function to establish the degree of membership for a given proposition to a determined value of truth.

These approaches have become popular for the results that they yielded, and many implementations for specific solutions have been produced based on their premises [8, 12, 13]. However, it is our belief that each of these approaches have limitations in their expressivity for modelling Uncertainty. For instance, Probabilistic Logic is by far the most extended approach to handle uncertain information. Yet, every axiom stated in Probabilistic Logic is categorical. That is, let $A(x) : (p)$ be the axiom assigning a probability of p to the truth of the statement: "Object x belongs to concept A". Then it is implicitly implied that the probability of x not being a member of A is $1 - p$. In other words, $A(x) : (p) \implies \neg A(x) : (1 - p)$.

The use of Probabilistic Logic does not naturally allow the user to assign some amount of believe to the fact that we might be missing some information about a certain statement. We propose the use of Subjective Logic [7] to overcome this limitation. Subjective Logic was proposed by A. Jøsang as a tool to express structured argument models with an associated degree of truth. It is based on Dempster-Shafer theory of belief, and uses frames of discernment to assign belief masses to given statements. Under Subjective Logic, statements are extended with opinions. An opinion is a triple (b, d, u), where b is a degree of belief on the truth of the statement, d is the degree of disbelief associated with the statement, and u is a degree of uncertainty. These three degrees must

sum up to 1 to comply with Kolmogorov's probabilistic axioms. From an intuitive point of view, b represents the amount of evidence that support the validity of the axiom, d represents how much evidence has been collected against the statement, and u is the amount of evidence that is not available at the moment, but could tilt our confidence either for or against the validity of the axiom.

We will use this approach to model uncertainty in ontologies, extending ABox assertions with subjective opinions. In this manner, we will be able to encapsulate how much information is already known about the validity of a certain axiom, as well as how much information is currently unknown. The application of opinions to axioms will result in some constraints that must hold for the ontology to make sense. These constraints will form the foundation for our reasoning, since they will let us propagate our beliefs through the ontology.

3 Preliminaries

3.1 DL-Lite$_{core}$

We will follow the standard syntax and semantics for classical (that is, without uncertainty) description logics, and due to space constraints, will refer the reader to [2] for further details. The subfamily DL-Lite$_{core}$ will be used through this paper for simplicity sake, but the results presented in this paper could be extended to other families of description logics.

As usual, *A* denotes atomic concept names, and *r* denotes atomic role names. *B* denotes basic concepts, and *R* denotes roles or their inverses. All valid expressions for DL-Lite$_{core}$ are built using the following production rules: $P ::= r \mid r^{-}$, $B ::= A \mid \exists P \mid \exists P^{-}$.

A TBox $\mathcal{T}$ is a finite set of concept inclusions (CIs) $B \sqsubseteq B'$, or $B \sqsubseteq \neg B'$. An ABox $\mathcal{A}$ is a finite set of membership assertions of the form $A(a)$, $P(a,b)$. A DL-Lite$_{core}$ ontology $\mathcal{O}$ is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a DL-Lite$_{core}$ TBox, and $\mathcal{A}$ is a DL-Lite$_{core}$ ABox.

Following the standard semantics of description logics [2], the semantics of DL-Lite$_{core}$ is based on interpretations. An interpretation $\mathcal{I}$ is a pair $(\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\triangle^{\mathcal{I}}$ is a non-empty set of objects, and $\cdot^{\mathcal{I}}$ is an interpretation function, which maps every individual $a \in \mathcal{A}$ to an object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every class *C* into a subset $C^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$, and each role *R* to a subset $R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. An interpretation is a model of a TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$) if it satisfies all concept inclusions in $\mathcal{T}$ (resp. assertions in $\mathcal{A}$). An ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$ if $\mathcal{A}$ and $\mathcal{T}$ have a common model. We write $\mathcal{T} \models C \sqsubseteq D$ if for all models *I* of $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and say that *C* is subsumed by *D* relative to $\mathcal{T}$.

There are a number of common reasoning services that are usually provided when developing an application that deals with ontologies. Among these services, we can name:

- **Instance checking:** Given an individual x and an concept C, determine whether or not x is a member of C.
- **Instance retrieval:** Given a concept C, retrieve all the individuals that are members of C.

– **Consistency checking:** Given a knowledge base $\mathcal{KB}$, determine whether or not a model for $\mathcal{KB}$ exists.
– **Query answering:** Given a knowledge base $\mathcal{KB}$ and a query q, determine all the answers for q that satisfy every model of KB.

3.2 Subjective ABoxes

A subjective DL-Lite$_{core}$ ABox $\mathcal{SA}$ is an extension of a DL-Lite$_{core}$ ABox $\mathcal{A}$ in which every assertion in $\mathcal{A}$ is extended with an opinion. An opinion **w** over a statement x is a triple of positive numbers (b, d, u) such that $b + d + u = 1$; and in which b represents the degree of belief assigned to the truth of x, d represents the degree assigned to the falsehood of x, and u measures the degree of uncertainty associated with x. If, during the execution of any reasoning task, an opinion **w** is produced such that $b + d + u > 1$, we say that **w** is invalid. We denote with b(**w**), d(**w**), and u(**w**) the degrees of belief, disbelief and uncertainty associated with an opinion **w**, respectively, and with $\mathcal{W}$ the set of all possible opinions.

Definition 1. *Let $w_1 = (b_1, d_1, u_1)$ and $w_2 = (b_2, d_2, u_2)$ be two opinions about the same assertion α. We call w_1 a specialisation of w_2 ($w_1 \preceq w_2$) iff $b_2 \leq b_1$ and $d_2 \leq d_1$ (implies $u_1 \leq u_2$). Similarly, we call w_1 a generalisation of w_2 ($w_2 \preceq w_1$) iff $b_1 \leq b2$ and $d1 \leq d_2$ (implies $u_2 \leq u_1$).*

3.3 Example Scenario

In order to help us illustrate the many properties of the different aspects of Subjective DL-Lite, we present in this subsection a running example set in a medical domain. More specifically, we will consider a medical clinic, in which patients come seeking for a doctor to treat their illnesses. We can have the knowledge domain modelled by an ontology, with relevant relations represented in the TBox, and data for patients and clinical cases instantiated in the ABox. Table 1 illustrates the ontology that we are going to use for our example.

Table 1. Scenario knowledge base

$t_1 : GraveDisease \sqsubseteq Disease$	$t_2 MinorDisease \sqsubseteq Disease$
$t_3 : GraveDisease \sqsubseteq \neg MinorDisease$	$t_4 : \exists hasSymptom \sqsubseteq SickPatient$
$t_5 : CriticalPatient \sqsubseteq Patient \sqcap \exists hasGraveDisease$	$t_6 : PandemicDisease \sqsubseteq GraveDisease$

$a_1 : presentsPainIn(patientA, abdomen) : (1, 0, 0)$
$a_2 : hasSymptom(patientA, nausea) : (0.4, 0, 0.6)$
$a_3 : hasSymptom(patientA, migraine) : (0.4, 0, 0.6)$
$a_4 : hasFamilyCondition(patientA, IBS) : (0, 0, 1)$
$a_5 : hasPositiveTest(patientA, bloodTest) : (0.9, 0.05, 0.05)$

In our scenario, a patient sees the doctor due to an acute abdominal pain that he is suffering. Being the main reason for the visit, and having no reason to doubt the patient, the doctor proceeds to instantiate with total certainty the fact that the patient suffers an abdominal pain. This is covered by axiom a_1.

Next, our patient tells the doctor that he also suffers something that he cannot describe very well, midway between a nausea or a migraine. This uncertain claim, stemming from the fact that the patient lacks the expertise to differentiate two distinct symptoms, can be easily modelled in a subjective ontology with axioms a_2 and a_3. The

rational for this approach is based on the doctor having some reasons to belief that the patient suffers one of the symptoms, but not having enough information that could justify the choice of one over the other. It could be argued that, since either outcome is equally probable, both axioms should be extended with the opinion (0.5,0,0.5) instead. However, this is where the potential of Subjective Logic becomes clear, since such an opinion would reject any other option as the cause of the discomfort. By choosing to have a buffer of 0.1 degree in our uncertainty, we are leaving an door open for any other possible symptom that could be responsible for the malady. Indeed, it could be easily the case that the patient was suffering neither a nausea or a migraine, and instead had an ear infection. This capability of modelling what is unknown to us at the moment is what mainly differentiates Subjective Logic from similar approaches.

Continuing with our example, the doctor wants to know whether the Irritable Bowel Syndrome is present in any member of his relatives. Not even knowing about the disease itself, the patient finds himself unable to confirm, nor discard, any presence of it in his family. This is modelled by axiom a_4, in which there is no commitment towards the truth nor the falsehood of the claim. The opinion (0,0,1), representing total uncertainty about an axiom, is the most general opinion possible, since any other opinion must necessarily be a specialisation of it. We will call (0,0,1) the default opinion, and assume that any axioms that do not explicitly appear in our ABox are extended with it. With this approach, we reflect the fact that anything not stated in our ontology is unknown, rather than false.

Finally, the doctor decides to run a blood test on the patient, to discard possible diseases that could be responsible for the symptoms. After a couple of days the results arrive, and the test shows that all the values for the patient fall within standard nominal ranges. However, the doctor is aware that these tests have an error margin of five percent, in which either a false positive or negative can be delivered instead of the real result. Knowing this limitation of the tests, the doctor only commits 90% of his confidence to axiom a_5, reflecting the fact that the test itself is fallible by assigning 5% of his confidence to the disbelief degree of the axiom, and covering for some possible exceptional situations with the use of some uncertainty.

4 Semantics

4.1 Subjective DL-Lite Semantics

The semantics for a $\mathcal{S}$DL-Lite$_{core}$ ABox is given in terms of subjective interpretations. A subjective interpretation $\mathcal{I}$ is a pair $(\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the domain $\triangle^{\mathcal{I}}$ is a non-empty set of objects, and $\cdot^{\mathcal{I}}$ is a subjective function that maps:

- an individual a to an element $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$
- a named class A to a function $A^{\mathcal{I}} : \triangle^{\mathcal{I}} \to \mathcal{W}$
- a named property R to a function $R^{\mathcal{I}} : \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}} \to \mathcal{W}$

Following the example set in [10], we summarise the semantics for various axiomatic relations in $\mathcal{S}$DL-Lite$_{core}$ through Table 2. Top ($\top$) and bottom ($\bot$) are special

Table 2. Semantics of Subjective DL-Lite$_{core}$

	Syntax	Semantics
s_1 :	$\top$	$\top^{\mathcal{I}}(o) = (1, 0, 0)$
s_2 :	$\bot$	$\bot^{\mathcal{I}}(o) = (0, 1, 0)$
s_3 :	$\exists\, R$	$b\,((\exists\, R)^{\mathcal{I}}(o_1)) \geq \max \bigcup_{\forall o_2} \{b(R^I(o_1,o_2))\}$ and $d((\exists R)^{\mathcal{I}}(o_1)) \leq \min \bigcup_{\forall o_2} \{d(R^I(o_1, o_2))\}$
s_4 :	$\neg B$	$(\neg B)^{\mathcal{I}}(o) = \neg B^{\mathcal{I}}(o)$
s_5 :	R^-	$(R^-)^{\mathcal{I}}(o_2, o_1) = R^{\mathcal{I}}(o_1, o_2)$
s_6 :	$B_1 \sqsubseteq B_2$	$\forall o \in \Delta^{\mathcal{I}}, b(B_1^{\mathcal{I}}(o)) \leq b(B_2^{\mathcal{I}}(o))$ and $d(B_2^{\mathcal{I}}(o)) \leq d(B_1^{\mathcal{I}}(o))$
s_7 :	$B_1 \sqsubseteq \neg B_2$	$\forall o \in \Delta^{\mathcal{I}}, b(B_1^{\mathcal{I}}(o)) \leq d(B_2^{\mathcal{I}}(o))$ and $b(B_2^{\mathcal{I}}(o)) \leq d(B_1^{\mathcal{I}}(o))$
s_8 :	$B(\mathtt{a})$:w	$b(w) \leq b(B^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}))$ and $d(w) \leq d(B^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}))$
s_9 :	$R(\mathtt{a}, \mathtt{b})$:$w$	$b(w) \leq b(R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}}))$ and $d(w) \leq d(R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}}))$

concepts in our ontology. Every object in our domain is a member of top with total certainty. Likewise, we know with total certainty that no object in our domain is a member of bottom. For the rest of the axiomatic rules, the constraints given in Table 2 must hold for every object in the domain. To illustrate how the semantics might be applied, we can have a look at the scenario presented in section 3.3. It is clear that the most trivial interpretation possible is the one that links distinct object to each one of the individual appearing in the ABox, and then assigns to each required axiom the same opinion that it already has in the ABox. We could then proceed to infer new axioms by applying the constraints given by the semantics. For instance, imagine that we agree that the flu is usually a mild sickness, although at least 2% of the population experiment a virulent outcome every year. We can then instantiate the flu for the year 2015 with the following axiom: $a_6 : MinorDisease(flu2015) : (0.9, 0.02, 0.08)$. This opinion encapsulates our perception that the flu is usually a mild sickness, that this is not the case for 2% of the cases, and that there is a margin for which the actual statistical values of mild cases versus severe cases will fall this year. Now, from table 1, and the semantic rule s_6 from table 2, we can infer the axiom $a_7 : Disease(flu2015) : (0.9, 0, 0.1)$. Notice how only the belief is propagated from the subclass to the inferred superclass, since any amount committed to the disbelief that $flu2015$ is a $MinorDisease$ does not justify stating that it is not a $Disease$. Certainly it could be the case that $flu2015$ is later declared a $PandemicDisease$ instead, thus making it a $GraveDisease$, but a $Disease$ nonetheless. Also notice that the semantics for rule s_6 require a_7 to have a belief degree equal or greater than 0.9, but does not specify any exact value. One could argue that any value falling in the range [0.9,1] could be chosen as the resulting belief for the inferred axiom, since any of these values comply with the semantics. However, by selecting precisely the lowest possible value in the range, we are maximising the use of available information in our ontology, at the same time that minimise the commitment for the resulting axiom. In other words, this is precisely the value that yields the most general

opinion ω that complies with the semantics. Any other opinion of a_7 that complies with the semantics must be a specialisation of ω, and vice versa.

One interesting point to remark is that, in subjective environments, positive inclusions can lead to inconsistencies. This is not the case for classical knowledge bases, where inconsistencies were produced by violation of negative inclusions. We can illustrate this property going back to our example scenario. Imagine that the flu for 2015 gets declared a pandemic due to the high rate of spread in the population. Since we apply a series of guidelines and well-defined rules to check the criteria for the declaration of pandemics, we can state that the flu falls within the category of pandemia with total confidence using the following axiom $a_8 : PandemicDisease(flu2015) : (1, 0, 0)$. It is clear that this axiom introduces an inconsistency in the ontology. Intuitively, it does not make sense to declare $flu2015$ to be a minor disease with 90% of confidence at the same time that we state with total certainty that $flu2015$ is also a grave disease (as inferred through t_6). A direct application of the semantic constraint s_7 lets us spot the inconsistency.

One last note before continuing to the formal definition of inconsistencies for subjective ontologies. In our example, the inconsistency arose due to some dynamic behaviour present in our ontology. That is, there was some initial statement that was refined at a later stage. Although extremely interesting, the task of introducing some dynamic aspect to our ontologies will be left for a future work. We will consider our ABoxes to be static in nature, and the only inconsistencies will arise due to the implicit relations given by axioms at the TBox level.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{S}$DL-Lite$_{core}$knowledge base, α be an axiom of $\mathcal{K}$, and $\mathcal{I}$ and interpretation of $\mathcal{K}$. The following will provide a formal definition of consistency for subjective ontologies:

Definition 2. *$\mathcal{I}$ is a model of α, denoted $\mathcal{I} \models \alpha$, if $\alpha^{\mathcal{I}}$ satisfies all the constraints presented in table 2.*

Definition 3. *$\mathcal{I}$ is a model of $\mathcal{K}$, denoted $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I} \models \alpha$ for each $\alpha \in \mathcal{K}$*

Definition 4. *$\mathcal{K}$ is consistent if it has at least one model*

Definition 5. *$\mathcal{K}$ models α, denoted $\mathcal{K} \models \alpha$, if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of $\mathcal{K}$.*

Definition 6. *σ is an answer for a query q over $\mathcal{K}$, denoted $\mathcal{K} \models_q \sigma$, if σ is an answer of q for every possible model of $\mathcal{K}$.*

Finally we need to redefine the meaning of some common reasoning tasks for a subjective ontology $\mathcal{K}$:

- **Instance checking:** Given an individual x and an concept C, determine the most general opinion ω such that $\mathcal{K} \models C(x) : \omega$.
- **Instance retrieval:** Given a concept C, return the set $\{C(x) : \omega \mid \omega$ is the most general opinion such that $\mathcal{K} \models C(x) : \omega\}$.
- **Query answering:** Given a query q, return the set $\{ \sigma : \omega \mid \omega$ is the most general opinion such that $\mathcal{K} \models_q \sigma : \omega\}$.

5 Canonical Interpretation

Following the example presented in [3], we now will provide a methodology to build a canonical interpretation of a subjective knowledge base $\mathcal{SK}$. To achieve this goal, we will follow the notion of chase [1]. In particular, we will adapt the notion of restricted chase adopted by Johnson and Klug in [6]. This restricted chase will be constructed in an iterative manner by applying a series of rules based on TBox axioms. For easiness of exposition, we assume that every assertion α that does not explicitly appear in the subjective ABox $\mathcal{A}$ has the vacuous opinion (0,0,1) associated to it. In a more formal way, our assumption states that we work with the extended subjective ABox $\mathcal{A}'$ given by $\mathcal{A}' = \mathcal{A} \cup \{\alpha : (0,0,1)\}$, if $\alpha : w \notin \mathcal{A}$ for any opinion w. We will also make use of the function ga, that takes as input a basic role and two constants, and returns a membership assertion as specified below:

$$ga(R,a,b) = \begin{cases} P(a,b), & if R = P \\ P(b,a), & if R = P^- \end{cases} \tag{1}$$

Definition 7. *Let $\mathcal{S}$ be a set of DL-Lite$_{core}$ membership assertions, and let $\mathcal{T}_\alpha$ be a set of DL-Lite$_{core}$ TBox axioms. Then, an axiom $\alpha \in \mathcal{T}_\alpha$ is applicable in $\mathcal{S}$ to a membership assertion $f \in \mathcal{S}$ if*

- ***(cr1)*** $\alpha = A_1 \sqsubseteq A_2, f = A_1(a) : w$, *and* $A_2(a) : w' \in \mathcal{S}$, *with* $b(w) > b(w')$
- ***(cr2)*** $\alpha = A_1 \sqsubseteq A_2, f = A_2(a) : w$, *and* $A_1(a) : w' \in \mathcal{S}$, *with* $d(w) > d(w')$
- ***(cr3)*** $\alpha = A \sqsubseteq \exists R, f = A(a) : w$, *and there does not exist any constant b such that ga(R,a,b) : w' $\in \mathcal{S}$, with* $b(w') > b(w)$
- ***(cr4)*** $\alpha = \exists R \sqsubseteq A$, *f = ga(R,a,b) : w, and A(a) : w' $\in \mathcal{S}$, with* $b(w) > b(w')$
- ***(cr5)*** $\alpha = \exists R \sqsubseteq A$, *f = ga(R,a,b) : w, and A(a) : w' $\in \mathcal{S}$, with* $d(w') > d(w)$
- ***(cr6)*** $\alpha = A_1 \sqsubseteq \neg A_2, f = A_1(a) : w$, *and* $A_2(a) : w' \in \mathcal{S}$, *with* $b(w') > d(w)$
- ***(cr7)*** $\alpha = A_2 \sqsubseteq \neg A_1, f = A_1(a) : w$, *and* $A_2(a) : w' \in \mathcal{S}$, *with* $b(w') > d(w)$
- ***(cr8)*** $\alpha = A \sqsubseteq \neg\exists R, f = ga(R,a,b) : w$, *and* $A(a) : w' \in \mathcal{S}$, *with* $b(w') > d(w)$
- ***(cr9)*** $\alpha = A \sqsubseteq \neg\exists R, f = A(a) : w$, *and* $ga(R,a,b) : w' \in \mathcal{S}$, *with* $b(w') > d(w)$
- ***(cr10)*** $\alpha = \exists R \sqsubseteq \neg A, f = ga(R,a,b) : w$, *and* $A(a) : w' \in \mathcal{S}$, *with* $b(w) > d(w')$
- ***(cr11)*** $\alpha = \exists R \sqsubseteq \neg A, f = A(a) : w$, *and* $ga(R,a,b) : w' \in \mathcal{S}$, *with* $b(w) > d(w')$

Applicable axioms can be used, i.e., applied, in oder to construct the chase of a knowledge base. The chase of a $\mathcal{S}$DL-Lite$_{core}$KB is a (possibly infinite) set of membership assertions, constructed step-by-step starting from the ABox $\mathcal{A}$. At each step of the process, an axiom $\alpha \in \mathcal{T}$ is applied to a membership assertion $f \in \mathcal{S}$. Applying an axiom means refining our opinion about a certain f', that might not appear explicitly in $\mathcal{S}$. The outcome of the application is a new set S' in which α is no longer applicable to f.

This construction process heavily depends on the order in which we select both the TBox axiom and the membership assertion in each iteration, as well as what constants we introduce when required. Therefore, we can produce a number of syntactically distinct sets of membership assertions following this process. However, it is possible to

show that the result is unique up to renaming of constants occurring in each such set. In order to achieve this, we select TBox axioms, membership assertions and constant symbols in lexicographic order. We denote with Γ_A the set of all constant symbols occurring in $\mathcal{A}$. We assume to have an infinite set Γ_N of constant symbols not occurring in $\mathcal{A}$. Finally, the set $\Gamma_C = \Gamma_A \cup \Gamma_N$ is ordered in lexicographic order.

Definition 8. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathcal{SDL\text{-}Lite}_{core}$KB, let $\mathcal{T}_\alpha$ be the set of assertions in $\mathcal{T}$, let n be the number of membership assertions in $\mathcal{A}$, and let Γ_N be the set of constants defined above. Assume that the membership assertions in $\mathcal{A}$ are numbered from 1 to n following their lexicographic order, and consider the following definition*

- $\mathcal{S}_0 = \mathcal{A}$
- $\mathcal{S}_{j+1} = \{\mathcal{S}_j \setminus f_{old}\} \cup \{f_{new}\}$

Then, we call chase of $\mathcal{K}$, denoted $chase(\mathcal{K})$, the set of membership assertions obtained as the (possibly infinite) union of all S_j, i.e.,

$$chase(\mathcal{K}) = \bigcup_{j \in \mathbb{N}} \mathcal{S}_j \tag{2}$$

The element f_{old}, presented in definition 8, is the axiom whose opinion is being refined by f_{new}. The membership assertion f_{new}, numbered with $n + j + 1$ in $\mathcal{S}_{j+1}$, is obtained as follows:

Definition 9. *Let f be the first membership assertion in S_j such that there exists a $\alpha \in \mathcal{T}_\alpha$ applicable in $\mathcal{S}_j$ to f, let α be the lexicographically first TBox axiom applicable in $\mathcal{S}_j$ to f, and let a_{new} be the constant of Γ_N that follows lexicographically all constants occurring in $\mathcal{S}_j$*

***case** α, f of*

(cr1) $\alpha = A_1 \sqsubseteq A_2, f = A_1(a) : w, A_2(a) : w' \in \mathcal{S}$
then $f_{new} = A_2(a) : (b(w), d(w'), 1 - b(w) - d(w'))$

(cr2) $\alpha = A_1 \sqsubseteq A_2, f = A_2(a) : w, A_1(a) : w' \in \mathcal{S}$
then $f_{new} = A_1(a) : (b(w'), d(w), 1 - b(w') - d(w))$

(cr3) $\alpha = A \sqsubseteq \exists R, f = A(a) : w, \exists R(a) : w' \in \mathcal{S}$
then $f_{new} = ga(R, a, a_{new}) : (b(w), d(w'), 1 - b(w) - d(w'))$

(cr4) $\alpha = \exists R \sqsubseteq A, f = ga(R, a, b) : w, A(a) : w' \in \mathcal{S}$
then $f_{new} = A(a) : (b(w), d(w'), 1 - b(w) - d(w'))$

(cr5) $\alpha = \exists R \sqsubseteq A$, *f = ga(R,a,b) : w, and A(a) : w'* $\in \mathcal{S}$
then $f_{new} = ga(R, a, b) : (b(w), d(w'), 1 - b(w) - d(w'))$

(cr6) $\alpha = A_1 \sqsubseteq \neg A_2, f = A_1(a) : w$, *and* $A_2(a) : w' \in \mathcal{S}$
then $f_{new} = A_1(a) : (b(w), b(w'), 1 - b(w) - b(w'))$

(cr7) $\alpha = A_2 \sqsubseteq \neg A_1, f = A_1(a) : w$, *and* $A_2(a) : w' \in \mathcal{S}$
then $f_{new} = A_1(a) : (b(w), b(w'), 1 - d(w) - b(w'))$

(cr8) $\alpha = A \sqsubseteq \neg \exists R, f = ga(R, a, b) : w$, *and* $A(a) : w' \in \mathcal{S}$
then $f_{new} = ga(R, a, b) : (b(w), b(w'), 1 - b(w) - b(w'))$

(cr10) $\alpha = \exists R \sqsubseteq \neg A$, *f = ga(R,a,b) : w, and A(a) : w'* $\in \mathcal{S}$
then $f_{new} = ga(R, a, b) : (b(w), b(w'), 1 - b(w) - b(w'))$

(cr11) $\alpha = \exists R \sqsubseteq \neg A, f = A(a) : w$, *and* $ga(R, a, b) : w' \in \mathcal{S}$
then $f_{new} = A(a) : (b(w), b(w'), 1 - b(w) - b(w'))$

It is worth noting that the application of chase rules can be a source of inconsistencies in the ontology. By increasing the belief degree of an opinion (resp. disbelief), we may put it in conflict with its disbelief degree (resp. belief), rendering the opinion invalid. Having an invalid opinion in our KB means that no interpretation will be able to satisfy it.

With the notion of chase in place we can introduce the notion of canonical interpretation. We define $can(\mathcal{K})$ as the interpretation $\langle \triangle^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})} \rangle$, where:

- $\triangle^{can(\mathcal{K})} = \Gamma_C$
- $a^{can(\mathcal{K})} = a$, for each constant a occurring in $chase(\mathcal{K})$
- $A^{can(\mathcal{K})} : \Gamma_C \rightarrow \mathcal{W}$, such that $A(a) : w \in chase(\mathcal{K}) \implies A^{can(\mathcal{K})}(a) = w$
- $P^{can(\mathcal{K})} : \Gamma_C \times \Gamma_C \rightarrow \mathcal{W}, P(a_1, a_2) : w \in chase(\mathcal{K}) \implies P^{can(\mathcal{K})}(a, b) = w$

We can also define $can_i(\mathcal{K}) = \langle \triangle^{can(\mathcal{K})}, \cdot^{can_i(\mathcal{K})} \rangle$ as the interpretation relative to $chase_i(\mathcal{K})$ instead of $chase(\mathcal{K})$.

Lemma 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathcal{SDL}$-Lite$_{core}$ knowledge base, then $can(\mathcal{K})$ is a model of $\mathcal{K}$ iff every opinion w that appears in $can(\mathcal{K})$ is valid.* □

$Proof.(Sketch)$

$\Leftarrow$ If any of the opinions w that appear in the canonical interpretation is invalid, i.e., $b(w) + d(w) > 1$, then it is obvious that the canonical interpretation is not a model of $\mathcal{K}$.

$\Rightarrow$ The fact that $can(K)$ satisfies all membership assertions in $\mathcal{A}$ follows from the fact that $A \subseteq chase(\mathcal{K})$. □

Lemma 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathcal{SDL}$-Lite$_{core}$ knowledge base, then if $can(\mathcal{K})$ is a model of $\mathcal{K}$, every other model of $\mathcal{K}$ is a specialisation of $can(\mathcal{K})$.* □

$Proof.(Sketch)$

Let m be a model of $\mathcal{K}$, and $m(\alpha) = \omega$ the opinion assigned by m to the assertion $\alpha \in \mathcal{K}$. Let $can(\alpha) = \omega_c$ be the opinion assigned by the canonical model to α, with $\omega_c \preceq \omega$. This means that, according to m, ω is a perfectly valid opinion for α. However, since ω_c is a specialisation of ω, we can infer that, while building the chase, there was a semantic constraint applicable to α that it is not satisfied by m. Given that there is at least one semantic constraint applicable to α that is not covered by m, it is clear that m is not a model of $\mathcal{K}$. We conclude for these reasons that every model of $\mathcal{K}$ must be at most as general as can.

□

The implications from lemma 2 are profound and very relevant. Knowing that every model of $\mathcal{K}$ is a specialisation of the canonical interpretation, we can focus on answering queries over this canonical interpretation. Any answer that is valid for the canonical interpretation will be valid for any other possible interpretation.

Of course, from a practical point of view, we will never construct the chase nor use directly the canonical model, since it might not be feasible to construct the chase for huge collections of data in a reasonable amount of time. Instead, we will apply the chase rules during the rewriting of the query, in such a way that we simulate the propagation

of the beliefs performed during the chase into the final query. Following this approach, we can be sure to obtain a valid answer for our original query, since this will be an answer for the canonical model and, through virtue of lema 2, an answer for any other interpretation of $\mathcal{K}$.

6 Conclusions

It is expected that the capability to handle uncertainty in query answering solutions will be a critical requirement for future applications. Precisely to address this problem we propose a subjective extension of DL-Lite, to combine the efficient query answering properties of DL-Lite with the uncertainty modelling of Subjective Logic. Our main contributions come in the form of the theoretical foundation for the justification of the semantics used in Subjective DL-Lite, and the construction of a canonical model through a chase. We have shown that the theory behind this approach is sound, and could be used to develop a query answering application with support for uncertainty. For our future works we still need to demonstrate that every possible interpretation is a specialisation of the canonical model. Thus, any answer given for the canonical model will be an answer for the rest of the interpretation. Finally, in order to develop our query answering application, we need to define the algorithms that will perform inference over the set of axioms of the ontology and collect the answers to the queries. The initial results are promising, and encourages us to continue in this interesting, though challenging, line of research.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. ADDISON WESLEY Publishing Company Incorporated, 1995.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[3] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 2007.

[4] D. Dubois and H. Prade. Possibility theory, probability theory and multiple-valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):35–66, 2001.

[5] V. Gutiérrez-Basulto, J. C. Jung, C. Lutz, and L. Schröder. A closer look at the probabilistic description logic Prob-EL. In *AAAI*, 2011.

[6] D. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, 28(1):167 – 189, 1984.

[7] A. Jøsang. *Subjective Logic*. Book Draft, 2011.

[8] G. Qi and J. Pan. A tableau algorithm for possibilistic description logic $\mathcal{ALC}$. In D. Calvanese and G. Lausen, editors, *Web Reasoning and Rule Systems*, volume 5341 of *Lecture Notes in Computer Science*, pages 238–239. Springer Berlin Heidelberg, 2008.

[9] G. Qi, J. Z. Pan, and Q. Ji. A possibilistic extension of description logics. In *In Proc. of DL07, 2007*, 2007.

[10] M. Sensoy, A. Fokoue, J. Z. Pan, T. J. Norman, Y. Tang, N. Oren, and K. Sycara. Reasoning about uncertain information and conflict resolution through trust revision. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 837–844, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[11] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:2001, 2001.

[12] U. Straccia and I. cnr Via G. Moruzzi. Transforming fuzzy description logics into classical description logics. In *In Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04), number 3229 in Lecture Notes in Computer Science*, pages 385–399. Springer Verlag, 2004.

[13] A. W. Systeme, T. Lukasiewicz, and T. Lukasiewicz. Probabilistic description logics for the semantic web, 2007.

DysToPic: a Multi-Engine Theorem Prover for Preferential Description Logics

Laura Giordano[1], Valentina Gliozzi[2],
Nicola Olivetti[3], Gian Luca Pozzato[2], and Luca Violanti[4]

[1] DISIT - U. Piemonte Orientale - Alessandria, Italy - laura.giordano@unipmn.it
[2] Dip. Informatica - Universitá di Torino - Italy
{valentina.gliozzi,gianluca.pozzato}@unito.it
[3] Aix Marseille Université - ENSAM, Université de Toulon, LSIS UMR 7296 - France
nicola.olivetti@univ-amu.fr
[4] NCR Edinburgh - United Kingdom - luca.violanti@gmail.com

Abstract. We describe DysToPic, a theorem prover for the preferential Description Logic $\mathcal{ALC} + \mathbf{T}_{min}$. This is a nonmonotonic extension of standard $\mathcal{ALC}$ based on a typicality operator $\mathbf{T}$, which enjoys a preferential semantics. DysToPic is a multi-engine Prolog implementation of a labelled, two-phase tableaux calculus for $\mathcal{ALC} + \mathbf{T}_{min}$ whose basic idea is that of performing these two phases by different machines. The performances of DysToPic are promising, and significantly better than the ones of its predecessor PreDeLo 1.0.

1 Introduction

Nonmonotonic extensions of Description Logics (DLs) have been actively investigated since the early 90s [3, 1, 4, 6, 7, 13, 5]. A simple but powerful nonmonotonic extension of DLs is proposed in [7, 13, 12]: in this approach "typical" or "normal" properties can be directly specified by means of a "typicality" operator $\mathbf{T}$ enriching the underlying DL; the typicality operator $\mathbf{T}$ is essentially characterized by the core properties of nonmonotonic reasoning axiomatized by either *preferential logic* [14] or *rational logic* [15]. In these logics one can consistently express defeasible inclusions and exceptions such as "normally, kids eat chocolate, but typical kids who have lactose intolerance do not":

$\mathbf{T}(Kid) \sqsubseteq ChocolateEater$
$\mathbf{T}(Kid \sqcap \exists HasIntolerance.Lactose) \sqsubseteq \neg ChocolateEater.$

In order to perform useful inferences, in [13] we have introduced a nonmonotonic extension of the logic $\mathcal{ALC}$ plus $\mathbf{T}$ based on a minimal model semantics. Intuitively, the idea is to restrict our consideration to models that maximize typical instances of a concept: more in detail, we introduce a preference relation among $\mathcal{ALC}$ plus $\mathbf{T}$ models, then we define a *minimal entailment* restricted to models that are minimal with respect to such preference relation. The resulting logic, called $\mathcal{ALC} + \mathbf{T}_{min}$, supports typicality assumptions, so that if one knows that Roman is a kid, one can nonmonotonically assume that he is also a *typical* kid and therefore that he eats chocolate. As an example, for a TBox specified by the inclusions above, in $\mathcal{ALC} + \mathbf{T}_{min}$ we can infer that:

TBox $\models_{\mathcal{ALC}+\mathbf{T}_{min}} \mathbf{T}(Kid \sqcap Tall) \sqsubseteq ChocolateEater$[5]
TBox $\cup \{Kid(daniel)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}} ChocolateEater(daniel)$
TBox $\cup \{Kid(daniel), \exists HasIntolerance.Lactose(daniel)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}}$

[5] Being tall is irrelevant with respect to eating chocolate or not.

$$\models_{\mathcal{ALC}+\mathbf{T}_{min}} \neg ChocolateEater(daniel)^{6}$$
$$\text{TBox} \cup \{Kid(daniel),\ Tall(daniel)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}} ChocolateEater(daniel)$$
$$\text{TBox} \cup \{\exists HasBrother.\ Kid(seth)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}} \exists\ HasBrother.\ ChocolateEater(seth)^{7}$$

In this work we focus on theorem proving for nonmonotonic extensions of DLs. We introduce `DysToPic`, a theorem prover for preferential Description Logic $\mathcal{ALC} + \mathbf{T}_{min}$. `DysToPic` implements the labelled tableaux calculus for this logic introduced in [13] performing a two-phase computation: in the first phase, candidate models falsifying a given query are generated (complete open branches); in the second phase the minimality of candidate models is checked by means of an auxiliary tableau construction. `DysToPic` is a multi-engine theorem prover, whose basic idea is that the two phases of the calculus are performed by different machines: a "master" machine M, called the *employer*, executes the first phase of the tableaux calculus, whereas other computers are used to perform the second phase on open branches detected by M. When M finds an open branch, it invokes the second phase on the calculus on a different "slave" machine, called *worker*, S_1, while M goes on performing the first phase on other branches, rather than waiting for the result of S_1. When another open branch is detected, then another machine S_2 is involved in the procedure in order to perform the second phase of the calculus on that branch. In this way, the second phase is performed simultaneously on different branches, leading to a significant increase of the performances.

Labelled tableaux calculi are implemented in Prolog, following the line of the predecessor `PreDeLo 1.0`, introduced in [11]: `DysToPic` is inspired by the methodology introduced by the system leanTAP [2], even if it does not fit its style in a rigorous manner. The basic idea is that each axiom or rule of the tableaux calculi is implemented by a Prolog clause of the program: the resulting code is therefore simple and compact.

In general, the literature contains very few proof methods for nonmonotonic extensions of DLs. We provide some experimental results to show that the performances of `DysToPic` are promising, in particular comparing them to the ones of `PreDeLo 1.0`. `DysToPic` is available for free download at:

`http://www.di.unito.it/~pozzato/theoremprovers.html`

2 The Logic $\mathcal{ALC} + \mathbf{T}_{min}$

The logic $\mathcal{ALC} + \mathbf{T}_{min}$ is obtained by adding to the standard $\mathcal{ALC}$ the typicality operator $\mathbf{T}$ [7, 12]. The intuitive idea is that $\mathbf{T}(C)$ selects the *typical* instances of a concept C. We can therefore distinguish between the properties that hold for all instances of concept C ($C \sqsubseteq D$), and those that only hold for the normal or typical instances of C ($\mathbf{T}(C) \sqsubseteq D$).

The language $\mathcal{L}$ of the logic is defined by distinguishing *concepts* and *extended concepts* as follows. Given an alphabet of concept names $\mathcal{C}$, of role names $\mathcal{R}$, and of individual constants $\mathcal{O}$, $A \in \mathcal{C}$ and $\top$ are *concepts* of $\mathcal{L}$; if $C, D \in \mathcal{L}$ and $R \in \mathcal{R}$, then $C \sqcap D, C \sqcup D, \neg C, \forall R.C, \exists R.C$ are *concepts* of $\mathcal{L}$. If C is a concept, then C and $\mathbf{T}(C)$ are *extended concepts*, and all the boolean combinations of extended concepts

[6] Giving preference to more specific information.

[7] Minimal consequence applies to individuals not explicitly named in the ABox as well, without any ad-hoc mechanism.

are extended concepts of $\mathcal{L}$. A KB is a pair (TBox,ABox). TBox contains inclusion relations (subsumptions) $C \sqsubseteq D$, where C is an extended concept of the form either C' or $\mathbf{T}(C')$, and $D \in \mathcal{L}$ is a concept. ABox contains expressions of the form $C(a)$ and $R(a,b)$, where $C \in \mathcal{L}$ is an extended concept, $R \in \mathcal{R}$, and $a, b \in \mathcal{O}$.

In order to provide a semantics to the operator $\mathbf{T}$, we extend the definition of a model used in "standard" logic $\mathcal{ALC}$. The idea is that the operator $\mathbf{T}$ is characterized by a set of postulates that are essentially a reformulation of the Kraus, Lehmann and Magidor's axioms of *preferential logic* **P** [14]. Intuitively, the assertion $\mathbf{T}(C) \sqsubseteq D$ corresponds to the conditional assertion $C \mathrel{|\!\sim} D$ of **P**. $\mathbf{T}$ has therefore all the "core" properties of nonmonotonic reasoning as it is axiomatized by **P**. The idea is that there is a global preference relation among individuals, in the sense that $x < y$ means that x is "more normal" than y, and that the typical members of a concept C are the minimal elements of C with respect to this relation. In this framework, an element $x \in \Delta$ is a *typical instance* of some concept C if $x \in C^I$ and there is no element in C^I *more typical* than x. The typicality preference relation is partial.

Definition 1. *Given an irreflexive and transitive relation $<$ over Δ, we define $Min_<(S) = \{x : x \in S$ and $\nexists y \in S$ s.t. $y < x\}$. We say that $<$ is well-founded if and only if, for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$.*

Definition 2. *A model of $\mathcal{ALC} + \mathbf{T}_{min}$ is any structure $\langle \Delta, <, I \rangle$, where: Δ is the domain; I is the extension function that maps each extended concept C to $C^I \subseteq \Delta$, and each role R to a $R^I \subseteq \Delta \times \Delta$; $<$ is an irreflexive, transitive and well-founded (Definition 1) relation over Δ. I is defined in the usual way (as for $\mathcal{ALC}$) and, in addition, $(\mathbf{T}(C))^I = Min_<(C^I)$.*

Given a model $\mathcal{M}$ of Definition 2, I can be extended so that it assigns to each individual a of $\mathcal{O}$ a distinct element a^I of the domain Δ (unique name assumption). We say that $\mathcal{M}$ satisfies an inclusion $C \sqsubseteq D$ if $C^I \subseteq D^I$, and that $\mathcal{M}$ satisfies $C(a)$ if $a^I \in C^I$ and $R(a,b)$ if $(a^I, b^I) \in R^I$. Moreover, $\mathcal{M}$ satisfies TBox if it satisfies all its inclusions, and $\mathcal{M}$ satisfies ABox if it satisfies all its formulas. $\mathcal{M}$ satisfies a KB (TBox,ABox), if it satisfies both TBox and ABox.

The semantics of the typicality operator can be specified by modal logic. The interpretation of $\mathbf{T}$ can be split into two parts: for any x of the domain Δ, $x \in (\mathbf{T}(C))^I$ just in case (i) $x \in C^I$, and (ii) there is no $y \in C^I$ such that $y < x$. Condition (ii) can be represented by means of an additional modality $\Box$, whose semantics is given by the preference relation $<$ interpreted as an accessibility relation. The interpretation of $\Box$ in $\mathcal{M}$ is as follows: $(\Box C)^I = \{x \in \Delta \mid$ for every $y \in \Delta$, if $y < x$ then $y \in C^I\}$. We immediately get that $x \in (\mathbf{T}(C))^I$ if and only if $x \in (C \sqcap \Box \neg C)^I$.

Even if the typicality operator $\mathbf{T}$ itself is nonmonotonic (i.e. $\mathbf{T}(C) \sqsubseteq E$ does not imply $\mathbf{T}(C \sqcap D) \sqsubseteq E$), what is inferred from a KB can still be inferred from any KB' with KB $\subseteq$ KB'. In order to perform *nonmonotonic* inferences, in [13] we have strengthened the above semantics by restricting entailment to a class of minimal (or preferred) models. Intuitively, the idea is to restrict our consideration to models that *minimize the non-typical instances of a concept*.

Given a KB, we consider a finite set $\mathcal{L}_\mathbf{T}$ of concepts: these are the concepts whose non-typical instances we want to minimize. We assume that the set $\mathcal{L}_\mathbf{T}$ contains at least

all concepts C such that $\mathbf{T}(C)$ occurs in the KB or in the query F, where a *query* F is either an assertion $C(a)$ or an inclusion relation $C \sqsubseteq D$. As we have just said, $x \in C^I$ is typical for C if $x \in (\Box\neg C)^I$. Minimizing the non-typical instances of C therefore means to minimize the objects falsifying $\Box\neg C$ for $C \in \mathcal{L}_\mathbf{T}$. Hence, for a model $\mathcal{M} = \langle \Delta, <, I\rangle$, we define $\mathcal{M}^{\Box^-}_{\mathcal{L}_\mathbf{T}} = \{(x, \neg\Box\neg C) \mid x \notin (\Box\neg C)^I, \text{ with } x \in \Delta, C \in \mathcal{L}_\mathbf{T}\}$.

Definition 3 (Preferred and minimal models). *Given a model $\mathcal{M} = \langle \Delta, <, I\rangle$ of a knowledge base* KB*, and a model $\mathcal{M}' = \langle \Delta', <', I'\rangle$ of* KB*, we say that $\mathcal{M}$ is preferred to $\mathcal{M}'$ w.r.t. $\mathcal{L}_\mathbf{T}$, and we write $\mathcal{M} <_{\mathcal{L}_\mathbf{T}} \mathcal{M}'$, if (i) $\Delta = \Delta'$, (ii) $\mathcal{M}^{\Box^-}_{\mathcal{L}_\mathbf{T}} \subset \mathcal{M}'^{\Box^-}_{\mathcal{L}_\mathbf{T}}$, (iii) $a^I = a^{I'}$ for all $a \in \mathcal{O}$. $\mathcal{M}$ is a* minimal model *for* KB *(w.r.t. $\mathcal{L}_\mathbf{T}$) if it is a model of* KB *and there is no other model $\mathcal{M}'$ of* KB *such that $\mathcal{M}' <_{\mathcal{L}_\mathbf{T}} \mathcal{M}$.*

Definition 4 (Minimal Entailment in $\mathcal{ALC} + \mathbf{T}_{min}$). *A query F is minimally entailed in $\mathcal{ALC} + \mathbf{T}_{min}$ by* KB *with respect to $\mathcal{L}_\mathbf{T}$ if F is satisfied in all models of* KB *that are minimal with respect to $\mathcal{L}_\mathbf{T}$. We write* KB $\models_{\mathcal{ALC}+\mathbf{T}_{min}} F$.

As an example, consider the TBox of the Introduction.
We have that TBox $\cup\ \{Kid(daniel)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}} ChocolateEater(daniel)$, since $daniel^I \in (Kid \sqcap \Box\neg Kid)^I$ for all minimal models $\mathcal{M} = \langle \Delta <, I\rangle$ of the TBox. In contrast, by the nonmonotonic character of minimal entailment, TBox $\cup\ \{Kid(daniel), \exists HasIntolerance.Lactose(daniel)\} \models_{\mathcal{ALC}+\mathbf{T}_{min}} \neg ChocolateEater(daniel)$.

3 A Tableau Calculus for $\mathcal{ALC} + \mathbf{T}_{min}$

In this section we recall the tableau calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ for deciding whether a query F is minimally entailed from a KB in $\mathcal{ALC} + \mathbf{T}_{min}$ introduced in [13]. The calculus performs a two-phase computation: in the first phase, a tableau calculus, called $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, simply verifies whether KB $\cup\ \{\neg F\}$ is satisfiable in a model of Definition 2, building candidate models; in the second phase another tableau calculus, called $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$, checks whether the candidate models found in the first phase are *minimal* models of KB, i.e. for each open branch of the first phase, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ tries to build a model of KB which is preferred to the candidate model w.r.t. Definition 3. The whole procedure is formally defined at the end of this section (Definition 5).

$\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ tries to build an open branch representing a minimal model satisfying KB $\cup\ \{\neg F\}$, where $\neg F$ is the negation of the query F and is defined as follows: if $F \equiv C(a)$, then $\neg F \equiv (\neg C)(a)$; if $F \equiv C \sqsubseteq D$, then $\neg F \equiv (C \sqcap \neg D)(x)$, where x does not occur in KB. $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ makes use of labels, denoted with $x, y, z, \ldots$. Labels represent individuals either named in the ABox or implicitly expressed by existential restrictions. These labels occur in *constraints*, that can have the form $x \xrightarrow{R} y$ or $x : C$, where x, y are labels, R is a role and C is a concept of $\mathcal{ALC} + \mathbf{T}_{min}$ or has the form $\Box\neg D$ or $\neg\Box\neg D$, where D is a concept.

3.1 The Tableaux Calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$

A tableau of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is a tree whose nodes are pairs $\langle S \mid U\rangle$. S is a set of constraints, whereas U contains formulas of the form $C \sqsubseteq D^L$, representing inclusion relations $C \sqsubseteq D$ of the TBox. L is a list of labels, used in order to ensure the termination of the tableau calculus. A branch is a sequence of nodes $\langle S_1 \mid U_1\rangle, \langle S_2 \mid$

$U_2\rangle, \ldots, \langle S_n \mid U_n\rangle \ldots$, where each node $\langle S_i \mid U_i\rangle$ is obtained from its immediate predecessor $\langle S_{i-1} \mid U_{i-1}\rangle$ by applying a rule of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, having $\langle S_{i-1} \mid U_{i-1}\rangle$ as the premise and $\langle S_i \mid U_i\rangle$ as one of its conclusions. A branch is closed if one of its nodes is an instance of a (Clash) axiom, otherwise it is open. A tableau is closed if all its branches are closed.

The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are presented in Fig. 1. Rules $(\exists^+)$ and $(\Box^-)$ are called *dynamic* since they can introduce a new variable in their conclusions. The other rules are called *static*. We do not need any extra rule for the positive occurrences of $\Box$, since these are taken into account by the computation of $S^M_{x\rightarrow y}$ of $(\Box^-)$. The (cut) rule ensures that, given any concept $C \in \mathcal{L}_{\mathbf{T}}$, an open branch built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ contains either $x : \Box\neg C$ or $x : \neg\Box\neg C$ for each label x: this is needed in order to allow $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ to check the minimality of the model corresponding to the open branch. As mentioned above, given a node $\langle S \mid U\rangle$, each formula $C \sqsubseteq D$ in U is equipped with the list L of labels to which the rule $(\sqsubseteq)$ has already been applied. This avoids multiple applications of such rule to the same subsumption by using the same label.

In order to check the satisfiability of a KB, we build its *corresponding constraint system* $\langle S \mid U\rangle$, and we check its satisfiability. Given KB=(TBox,ABox), its *corresponding constraint system* $\langle S \mid U\rangle$ is defined as follows: $S = \{a : C \mid C(a) \in \text{ABox}\} \cup \{a \xrightarrow{R} b \mid R(a,b) \in \text{ABox}\}$; $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in \text{TBox}\}$. KB is satisfiable if and only if its corresponding constraint system $\langle S \mid U\rangle$ is satisfiable. In order to verify the satisfiability of KB $\cup \{\neg F\}$, we use $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ to check the satisfiability of the constraint system $\langle S \mid U\rangle$ obtained by adding the constraint corresponding to $\neg F$ to S', where $\langle S' \mid U\rangle$ is the corresponding constraint system of KB. To this purpose, the rules of the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are applied until either a contradiction is generated (*clash*) or a model satisfying $\langle S \mid U\rangle$ can be obtained.

The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are applied with the following *standard strategy*: 1. apply a rule to a label x only if no rule is applicable to a label y such that $y \prec x$ (where $y \prec x$ says that label x has been introduced in the tableaux later than y); 2. apply dynamic rules only if no static rule is applicable.

Theorem 1. *Given $\mathcal{L}_{\mathbf{T}}$, KB $\models_{\mathcal{ALC}+\mathbf{T}_{min}} F$ if and only if there is no open branch* **B** *in the tableau built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ for the constraint system corresponding to KB $\cup \{\neg F\}$ such that the model represented by* **B** *is a minimal model of KB.*

Thanks to the side conditions on the application of the rules and the blocking machinery adopted by the dynamic ones, in [13] it has been shown that any tableau generated by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ for $\langle S \mid U\rangle$ is finite.

3.2 The Tableaux Calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$

Let us now introduce the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ which checks whether each open branch **B** built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ represents a minimal model of the KB.

Given an open branch **B** of a tableau built from $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, let $\mathcal{D}(\mathbf{B})$ be the set of labels occurring in **B**. Moreover, let $\mathbf{B}^{\Box^-}$ be the set of formulas $x : \neg\Box\neg C$ occurring in **B**, that is to say $\mathbf{B}^{\Box^-} = \{x : \neg\Box\neg C \mid x : \neg\Box\neg C \text{ occurs in } \mathbf{B}\}$.

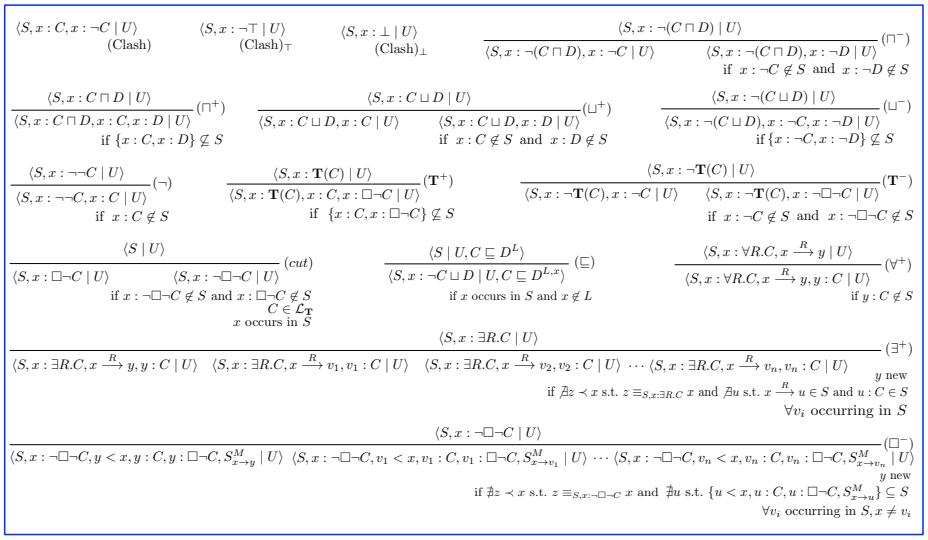


Fig. 1. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$.

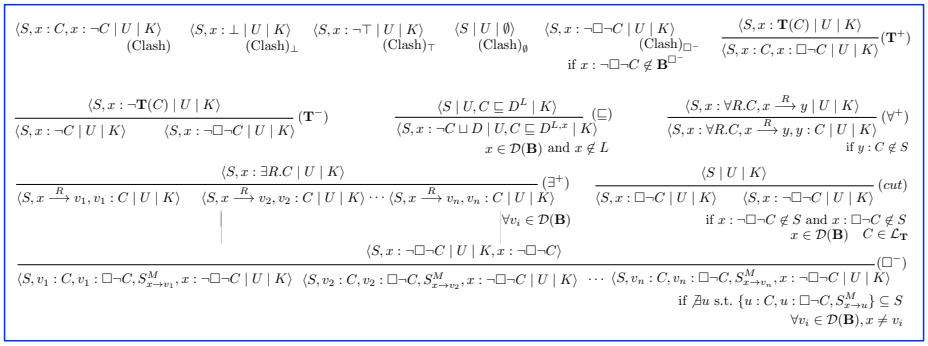


Fig. 2. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$. To save space, we only include the most relevant rules.

A tableau of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is a tree whose nodes are tuples of the form $\langle S \mid U \mid K \rangle$, where S and U are defined as in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, whereas K contains formulas of the form $x : \neg\Box\neg C$, with $C \in \mathcal{L}_{\mathbf{T}}$. The basic idea of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is as follows. Given an open branch $\mathbf{B}$ built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ and corresponding to a model $\mathcal{M}^{\mathbf{B}}$ of KB $\cup \{\neg F\}$, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ checks whether $\mathcal{M}^{\mathbf{B}}$ is a minimal model of KB by trying to build a model of KB which is preferred to $\mathcal{M}^{\mathbf{B}}$. To this purpose, it keeps track (in K) of the negated box formulas used in $\mathbf{B}$ ($\mathbf{B}^{\Box^-}$) in order to check whether it is possible to build a model of KB containing less negated box formulas.

The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ are shown in Figure 2. The tableau built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ closes if it is not possible to build a model smaller than $\mathcal{M}^{\mathbf{B}}$, it remains open otherwise. Since by Definition 3 two models can be compared only if they have the same domain, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ tries to build an open branch containing all the labels appearing in $\mathbf{B}$, i.e. those in $\mathcal{D}(\mathbf{B})$. To this aim, the dynamic rules use labels in $\mathcal{D}(\mathbf{B})$ instead of introducing

new ones in their conclusions. The rule ($\sqsubseteq$) is applied to *all the labels of* $\mathcal{D}(\mathbf{B})$ (and not only to those appearing in the branch). The rule ($\Box^-$) is applied to a node $\langle S, x : \neg\Box\neg C \mid U \mid K, x : \neg\Box\neg C\rangle$, that is to say when the negated box formula $x : \neg\Box\neg C$ also belongs to the open branch $\mathbf{B}$. Also in this case, the rule introduces a branch on the choice of the individual $v_i \in \mathcal{D}(\mathbf{B})$ to be used in the conclusion. In case a tableau node has the form $\langle S, x : \neg\Box\neg C \mid U \mid K\rangle$, and $x : \neg\Box\neg C \notin \mathbf{B}^{\Box^-}$, then $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ detects a clash, called $(\text{Clash})_{\Box^-}$: this corresponds to the situation where $x : \neg\Box\neg C$ does not belong to $\mathbf{B}$, while the model corresponding to the branch being built contains $x : \neg\Box\neg C$, and hence is *not* preferred to the model represented by $\mathbf{B}$. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ also contains the clash condition $(\text{Clash})_{\emptyset}$. Since each application of ($\Box^-$) removes the negated box formulas $x : \neg\Box\neg C$ from the set K, when K is empty all the negated boxed formulas occurring in $\mathbf{B}$ also belong to the current branch. In this case, the model built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ satisfies the same set of $x : \neg\Box\neg C$ (for all individuals) as $\mathbf{B}$ and, thus, it is not preferred to the one represented by $\mathbf{B}$.

Let KB be a knowledge base whose corresponding constraint system is $\langle S \mid U\rangle$. Let F be a query and let S' be the set of constraints obtained by adding to S the constraint corresponding to $\neg F$. $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is *sound and complete* in the following sense: an open branch $\mathbf{B}$ built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ for $\langle S' \mid U\rangle$ is satisfiable in a minimal model of KB iff the tableau in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ for $\langle S \mid U \mid \mathbf{B}^{\Box^-}\rangle$ is closed.

The termination of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is ensured by the fact that dynamic rules make use of labels belonging to $\mathcal{D}(\mathbf{B})$, which is finite, rather than introducing "new" labels in the tableau. Also, it is possible to show that the problem of verifying that a branch $\mathbf{B}$ represents a minimal model for KB in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is in NP in the size of $\mathbf{B}$.

The overall procedure $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is defined as follows:

Definition 5. *Let KB be a knowledge base whose corresponding constraint system is* $\langle S \mid U\rangle$. *Let* F *be a query and let* S' *be the set of constraints obtained by adding to* S *the constraint corresponding to* $\neg F$. *The calculus* $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ *checks whether a query* F *can be minimally entailed from a* KB *by means of the following procedure:*

- *the calculus* $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ *is applied to* $\langle S' \mid U\rangle$*;*
- *if, for each branch* $\mathbf{B}$ *built by* $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$*, either: (i)* $\mathbf{B}$ *is closed or (ii) the tableau built by the calculus* $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ *for* $\langle S \mid U \mid \mathbf{B}^{\Box^-}\rangle$ *is open, then the procedure says* YES *else the procedure says* NO

In [13] we have shown that $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is a sound and complete decision procedure for verifying if KB $\models^{\mathcal{L}_{\mathbf{T}}}_{\mathcal{ALC}+\mathbf{T}_{min}} F$, and that the problem is in CO-NEXP$^{\text{NP}}$.

4 Design of `DysToPic`

In this section we present `DysToPic`, a multi-engine theorem prover for reasoning in $\mathcal{ALC} + \mathbf{T}_{min}$. `DysToPic` is a SICStus Prolog implementation of the tableaux calculi $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ introduced in the previous section, wrapped by a Java interface which relies on the Java RMI APIs for the distribution of the computation. The system is designed for scalability and based on a "worker/employer" paradigm: the computational

burden for the "employer" can be spread among an arbitrarily high number of "workers" which operate in complete autonomy, so that they can be either deployed on a single machine or on a computer grid.

The basic idea underlying `DysToPic` is as follows: there is no need for the first phase of the calculus to wait for the result of one elaboration of the second phase on an open branch, before generating another candidate branch. Indeed, in order to prove whether a query F entails from a KB, the first phase can be executed on a machine; every time that a branch remains open after the first phase, the execution of the second phase for this branch can be performed in parallel, on a different machine. Meanwhile, the main machine (worker), instead of waiting for the termination of the second phase on that branch, can carry on with the computation of the first phase (potentially generating other branches). If a branch remains open in the second phase, then F is not minimally entailed from KB (we have found a counterexample), so the computation process can be interrupted early.

4.1 The Whole Architecture

In order to describe the architecture of `DysToPic` we refer to the *worker-employer* metaphor. The system is characterized by: (i) a single *employer*, which is in charge of verifying the query and yielding the final result. It also implements the first phase of the calculus and uses $\mathcal{TAB}_{PH1}^{\mathcal{ALC}+\mathbf{T}}$ to generate branches: the ones that it cannot close (representing candidate models of KB $\cup\{\neg F\}$), it passes to a *worker*; (ii) an unlimited number of *workers*, which use $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$ to evaluate the models generated by the *employer*; (iii) a *repository*, which stores all the answers coming from the *workers*.
First, each worker registers to the employer. When checking whether KB $\models_{\mathcal{ALC}+\mathbf{T}_{min}}$ F, the employer executes $\mathcal{TAB}_{PH1}^{\mathcal{ALC}+\mathbf{T}}$. If the employer needs to check whether an open branch generated by the first phase represents a minimal model of the KB, then it delegates the execution of the second phase to one of the registered workers, and consequently proceeds with its computation on other branches generated in the first phase. When a worker terminates its execution, it reports its result to the repository.

If every branch has been processed and each worker has answered affirmatively, i.e. each tableaux built in the second phase by $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$ is open, the employer can conclude that KB $\models_{\mathcal{ALC}+\mathbf{T}_{min}}$ F. Otherwise, the employer can conclude the proof as soon as the first negative answer comes into the repository, since (at least) a worker found a closed tableaux in $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$ for an open branch (candidate model) generated by the employer, in this case we have that KB $\not\models_{\mathcal{ALC}+\mathbf{T}_{min}}$ F. It is worth noticing that the employer has to keep a continuous dialogue with the repository.

The library `se.sics.jasper` is used in order to combine Java and SICStus Prolog to decouple the two phases of the calculus. In detail, the employer handles the query in `Employer.java`, a piece of Java code which presents it to `alct1.pl`, the Prolog core implementing $\mathcal{TAB}_{PH1}^{\mathcal{ALC}+\mathbf{T}}$. Every time that an open branch is generated, `alct1.pl` invokes `Phase1RMIStub.java`, another piece of Java code which will send it to the correct worker. Workers will then have to process the open branches with $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$, which is implemented in `alct2.pl`.

Concurrency is the main goal of our implementation, since we want the execution of the first phase of the calculus to be independent from the second one. Java natively

supports concurrency via multithreading. The employer uses a separate thread (implemented in `Phase1Thread.java`) to perform the current invocation of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ on a query, while its main thread polls the repository waiting for termination (the procedure can be stopped when the first counterexample is found, even if not all of the branches have been explored). During the execution of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, every time that the employer wants to ask a worker to verify a branch, a new thread is spawned. The worker itself makes use of threads: its main thread simply enqueues each request coming from the employer and spawns a new thread which performs $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$.

4.2 The Implementation of the Tableaux Calculi

Concerning the implementation of the tableaux calculi $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$, each machine of the system runs a SICStus Prolog implementation which is strongly related to the implementation of the calculi given by `PreDeLo 1.0`, introduced in [11]. The implementation is inspired by the "lean" methodology of leanTAP, even if it does not follow its style in a rigorous manner. The program comprises a set of clauses, each one implementing a rule or axiom of the tableau calculi. The proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional mechanism.

`DysToPic` comprises two main predicates, called `prove` and `prove_phase2`, implementing, respectively, the first and the second phase of the tableau calculi.

Phase 1: the `prove` predicate. Concerning the first phase of the calculi, executed by the employer, `DysToPic` represents a tableaux node $\langle S \mid U \rangle$ with two Prolog lists: `S` and `U`. Elements of `S` are either pairs `[X, F]`, representing formulas of the form $x : F$, or triples of the form either `[X,R,Y]` or `[X,<,Y]`, representing either roles $x \xrightarrow{R} y$ or the preference relation $x < y$, respectively. Elements of `U` are pairs of the form `[[C inc D],L]`, representing $C \sqsubseteq D^L \in U$ described in Section 3.1.

The calculi $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ are implemented by a top-level predicate

```
prove(+ABox,+TBox,[+X,+F],-Tree).
```

This predicate succeeds if and only if the query $x : F$ is minimally entailed from the KB represented by `TBox` and `ABox`. When the predicate succeeds, then the output term `Tree` matches a Prolog term representing the closed tableaux found by the prover. The top-level predicate `prove/4` invokes a second-level one:

```
prove(+S,+U,+Lt,+Labels,+ABOX,-Tree)
```

having 6 arguments. In detail, `S` corresponds to `ABox` enriched by the negation of the query $x : F$, whereas `Lt` is a list corresponding to the set of concepts $\mathcal{L}_{\mathbf{T}}$. `Labels` is the set of labels belonging to the current branch, whereas `ABOX` is used to store the initial ABox (i.e. without the negation of the query) in order to eventually invoke the second phase on it, in order to look for minimal models of the initial KB.

Each clause of the `prove/6` predicate implements an axiom or rule of the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$. To search a closed tableaux for $\langle S \mid U \rangle$, `DysToPic` proceeds as follows. First of all, if $\langle S \mid U \rangle$ is a clash, the goal will succeed immediately by using one of the clauses implementing axioms. As an example, the following clause implements (Clash):

```
prove(S,U,_,_,_,tree(clash)):-
      member([X,C],S),member([X, neg C],S),!.
```

If $\langle S \mid U \rangle$ is not an instance of the axioms, then the first applicable rule will be chosen, e.g. if `S` contains an intersection `[X,C and D]`, then the clause implementing the $(\sqcap^+)$ rule will be chosen, and `DysToPic` will be recursively invoked on its unique conclusion. `DysToPic` proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the dynamic rules is postponed as much as possible: this implements the strategy ensuring the termination of the calculi described in the previous section. As an example, the clause implementing $(\mathbf{T}^+)$ is as follows:

```
1. prove(S,U,Lt,Labels,ABOX,tree(...,Tree)):-member([X,ti C],S),
2.     (\+(member([X,C],S)); \+(member([X, box neg C],S))),!,
3.    prove([[X,C]|[[X, box neg C]|S]],U,Lt,Labels,ABOX,Tree),!.
```

In line 1, the standard Prolog predicate `member` is used in order to find a formula of the form $x : \mathbf{T}(C)$ in the list `S`. In line 2, the side conditions on the applicability of such a rule are checked: the rule can be applied if either $x : C$ or $x : \Box\neg C$ do not belong to `S`. In line 3 `DysToPic` is recursively invoked on the unique conclusion of the rule, in which $x : C$ and $x : \Box\neg C$ are added to the list `S`. The last clause of `prove` is:

```
prove(...) :- ... , jasper_call(JVM,
    method('employer/Phase1RMIStub','solveViaRMI',[static]),...,
    solve_via_rmi(NextWorkerName, 'toplevelphase2(...)' ) ),!.
```

invoked when no other clauses are applicable. In this case, the branch built by the employer represents a model for the initial set of formulas, then the `toplevelphase2` predicate is invoked on a worker in order to check whether such a model is minimal for KB.

Phase 2: the `prove_phase2` predicate. Given an open branch built by the first phase, the predicate `toplevelphase2` is invoked on a worker. It first applies an optimization preventing useless applications of $(\sqsubseteq)$, then it invokes the predicate

```
prove_phase2(+S,+U,+Lt,+K,+Bb,+Db).
```

`S` and `U` contain the initial KB (without the query), whereas `K`, `Bb` and `Db` are Prolog lists representing K, $\mathbf{B}^{\Box^-}$ and $\mathcal{D}(\mathbf{B})$ as described in Section 3.2. `Lt` is as for `prove/6`.

Also in this case, each clause of `prove_phase2` implements an axiom or rule of the calculi $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$. To search for a closed tableaux, `DysToPic` first checks whether the current node $\langle S \mid U \mid K \rangle$ is a clash. otherwise the first applicable rule will be chosen, and `DysToPic` will be recursively invoked on its conclusions. As an example, the clause implementing $(\mathbf{T}^+)$ is as follows:

```
prove_phase2(S,U,Lt,K,Bb,Db) :- select([X,ti C],S,S1),
       prove_phase2([[X,C]|[[X,box neg C]|S1]],U,Lt,K,Bb,Db),!.
```

Notice that, according to the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$, the principal formula to which the rule is applied is removed from the current node: to this aim, the SICStus Prolog predicate `select` is used rather than `member`.

4.3 Preliminary Performance Testing of `DysToPic`

We have made a preliminary attempt to show how `DysToPic` performs, especially in comparison with its predecessor `PreDeLo 1.0`. The performances of `DysToPic` are promising. We have tested both the provers by running SICStus Prolog 4.1.1 on Ubuntu

14.04.1 64 bit machines. Concerning DysToPic, we have tested it on 4 machines, namely: 1. a desktop PC with an Intel Core i5-3570K CPU (3.4-3.8GHz, 4 cores, 4 threads, 8GB RAM); 2. a desktop PC with an Intel Pentium G2030 CPU (3.0GHz, 2 cores, 2 threads, 4GB RAM); 3. a Lenovo X220 laptop with an Intel Core i7-2640M CPU (2.8-3.5GHz, 2 cores, 4 threads, 8GB RAM); 4. a Lenovo X230 laptop with an Intel Core i7-3520M CPU (2.9-3.6GHz, 2 cores, 4 threads, 8GB RAM).

We have performed two kinds of tests. On the one hand, we have randomly generated KBs with different sizes (from 10 to 100 ABox formulas and TBox inclusions) as well as different numbers of named individuals: in less than 10 seconds, both DysToPic and PreDeLo 1.0 are able to answer in more than the 75% of tests. Notice that, as far as we know, it does not exist a set of acknowledged benchmarks for defeasible DLs. On the other hand, we have tested the two theorem provers on specific examples. As expected, DysToPic is better in than the competitor in answering that a query F is not minimally entailed from a given KB. Surprisingly enough, its performances are better than the ones of PreDeLo 1.0 also in case the provers conclude that F follows from KB, as in the following example:

Example 1. Given TBox=$\{\mathbf{T}(Student) \sqsubseteq \neg IncomeTaxPayer, WorkingStudent \sqsubseteq Student, \mathbf{T}(WorkingStudent) \sqsubseteq IncomeTaxPayer\}$ and ABox=$\{Student(mario), WorkingStudent(mario), Tall(mario), Student(carlo), WorkingStudent(carlo), Tall(carlo), Student(giuseppe), WorkingStudent(giuseppe), Tall(giuseppe)\}$, we have tested both the provers in order to check whether $IncomeTaxPayer(mario)$ is minimally entailed from KB=(TBox,ABox). This query generates 1090 open branches in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, each one requiring the execution of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$. PreDeLo 1.0 answers in 370 seconds, whereas DysToPic answers in 210 seconds if only two machines are involved (employer + one worker). If 4 workers are involved, DysToPic only needs 112 seconds to conclude its computation.

Example 1 witnesses that the advantages obtained by distributing the computation justify the overhead introduced by the machinery needed for such distribution.

5 Conclusions

We have introduced DysToPic, a multi-engine theorem prover implementing tableaux calculi for reasoning in preferential DL $\mathcal{ALC} + \mathbf{T}_{min}$. DysToPic implements a distributed version of the tableaux calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ introduced in [13], exploiting the fact that the two phases characterizing such a calculus can be computed in parallel.

We aim at extending DysToPic to the *lightweight* DLs of the DL-Lite and $\mathcal{EL}$ family. Despite their relatively low expressivity, they are relevant for several applications, in particular in the bio-medical domain. Extensions of $\mathcal{EL}^{\bot}$ and of DL-Lite$_{core}$ with the typicality operator $\mathbf{T}$ have been proposed in [10], where it has also been shown that minimal entailment is in Π^p_2 (for $\mathcal{EL}^{\bot}$, if restricted to a specific fragment). Tableaux calculi performing a two phases computation, similar to $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$, have been proposed in [9, 8].

This research is partially supported by INDAM- GNCS Project 2015 "Logiche descrittive e ragionamento non monotono".

References

1. Baader, F., Hollunder, B.: Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. Journal of Automated Reasoning (JAR) 15(1), 41–68 (1995)
2. Beckert, B., Posegga, J.: leantap: Lean tableau-based deduction. Journal of Automated Reasoning (JAR) 15(3), 339–358 (1995)
3. Bonatti, P.A., Lutz, C., Wolter, F.: DLs with Circumscription. In: KR. pp. 400–410 (2006)
4. Bonatti, P.A., Faella, M., Sauro, L.: Defeasible inclusions in low-complexity dls: Preliminary notes. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 696–701 (2009)
5. Casini, G., Straccia, U.: Rational closure for defeasible description logics. In: Janhunen, T., Niemelä, I. (eds.) Logics in Artificial Intelligence - Proceedings of the12th European Conference (JELIA 2010). Lecture Notes in Computer Science (LNCS), vol. 6341, pp. 77–90. Springer (2010)
6. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logics (ToCL) 3(2), 177–225 (2002)
7. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: ALC+T: a preferential extension of description logics. Fundamenta Informaticae 96, 341–372 (2009)
8. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of $\mathcal{EL}^{\perp}$. In: Brünnler, K., Metcalfe, G. (eds.) Proceedings of the 20th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2011). Lecture Notes in Artificial Intelligence (LNAI), vol. 6793, pp. 180–195. Springer-Verlag, Bern, Switzerland (July 2011)
9. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A tableau calculus for a nonmonotonic extension of the Description Logic $DL - Lite_{core}$. In: Pirrone, R., Sorbello, F. (eds.) AI*IA 2011: Artificial Intelligence Around Man and Beyond - XIIth International Conference of the Italian Association for Artificial Intelligence, Palermo, Italy, September 15-17, 2011. Proceedings. Lecture Notes in Artificial Intelligence (LNAI), vol. 6934, pp. 164–176. Springer-Verlag, Palermo, Italy (Sptember 2011)
10. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Reasoning about typicality in low complexity DLs: the logics $\mathcal{EL}^{\perp}\mathbf{T}_{min}$ and *DL-Lite*$_c\mathbf{T}_{min}$. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 894–899. IOS Press, Barcelona, Spain (2011)
11. Giordano, L., Gliozzi, V., Jalal, A., Olivetti, N., Pozzato, G.L.: Predelo 1.0: a theorem prover for preferential description logics. In: Baldoni, M., Baroglio, C., Boella, G., Micalizio, R. (eds.) Proceedings of AI*IA 2013. Lecture Notes in Artificial Intelligence LNAI, vol. 8249, pp. 60 – 72. Springer, Torino (December 2013)
12. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Preferential vs Rational Description Logics: which one for Reasoning About Typicality? . In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010). FAIA (Frontiers in Artificial Intelligence and Applications), vol. 215, pp. 1069 – 1070. IOS Press, Lisbon, Portugal (August 2010)
13. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A NonMonotonic Description Logic for Reasoning About Typicality. Artificial Intelligence 195, 165 – 202 (2013)
14. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
15. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55(1), 1–60 (1992)

Saturation-Based Forgetting in the Description Logic $\mathcal{SIF}$

Patrick Koopmann, Renate A. Schmidt

The University of Manchester, UK
{koopmanp, schmidt}@cs.man.ac.uk

Abstract. Forgetting, which has also been studied under the names uniform interpolation, projection and variable elimination, deals with the elimination of symbols from an input ontology in such a way that all entailments in the remaining signature are preserved. The computed result, the uniform interpolant, can be seen as a restricted view of the original ontology, projected to a specified subset of the signature. Forgetting has applications in ontology reuse, ontology analysis and information hiding, and has been studied for a variety of description logics in the last years. However, forgetting in description logics with functional role restrictions and inverse roles has been an open problem so far. In this paper, we study the problem of forgetting concept symbols in the description logic $\mathcal{SIF}$, an expressive description logic supporting transitive roles, inverse roles and functional role restrictions. Saturation-based reasoning has been proven to be a well-suited technique for computing uniform interpolants practically in recently introduced methods. Since existing methods are usually optimised towards a specific aim such as satisfiability checking or classification, they cannot always directly be used for forgetting. In this paper we present a new saturation technique that is complete for forgetting concept symbols, and show how it can be used for computing uniform interpolants.

1 Introduction

We present a method for forgetting concept symbols from $\mathcal{SIF}$-ontologies. Modern applications, especially in bio-informatics or medical domains, lead to the development of ontologies that cover a large vocabulary. With rising complexity, these ontologies become harder to maintain and modify. It can therefore be advantageous to have tool-support for reducing the vocabulary used in an ontology. Forgetting restricts the vocabulary in an ontology in such a way that all entailments over the restricted vocabulary are preserved. In contrast to modules, uniform interpolants are completely in the desired signature and may contain different axioms than the input ontology.

There are numerous applications of forgetting that make the problem worth studying, of which we give a few examples. **Ontology Summary.** Comprehending a complex ontology can be hindered by a too large vocabulary used in the ontology. If the central concepts and roles of the ontology are known, forgetting

all but the central concepts of an ontology can be used to compute a more focused high-level summary of the ontology. **Ontology Analysis.** With increasing complexity of the ontology, understanding the relations between concepts and roles involved becomes more difficult. By forgetting all but a few chosen symbols, one can obtain a direct representation of the interactions between them. **Logical Difference.** In order to maintain an evolving ontology, it is necessary to be able to exhibit changes between different ontology versions. However, a syntactical diff between text representations of the ontologies is rarely useful in practice. In contrast, the logical difference between two ontologies is a semantic notion, which is defined by the difference of logical entailments in the common signature of these ontologies, or in a specified signature [8,7]. [16] show that the logical difference can easily be computed using the uniform interpolants of the ontologies. **Information Hiding.** As pointed out in [4], ontology-based systems are increasingly used in a range of applications that deal with sensitive information. Forgetting can be used as a means to eliminate confidential information if an ontology is to be shared between users with differing privileges.

Methods for forgetting have been investigated for a range of description logics, including DL-Lite [29], $\mathcal{EL}$ [9,17,20], $\mathcal{ALC}$ [28,16,12,11,14,15], $\mathcal{ALCH}$ [10] and $\mathcal{SHQ}$ [13]. So far, forgetting for description logics with inverse roles and functional role restrictions was an open problem.

As with most expressive description logics, $\mathcal{SIF}$ does not have uniform interpolation. This means that the result of forgetting may not always be finitely representable in $\mathcal{SIF}$. Take as an example the ontology $\mathcal{O} = \{A \sqsubseteq \exists r.B, B \sqsubseteq \exists r.B\}$. If we only use the expressivity $\mathcal{SIF}$ provides, forgetting B from $\mathcal{O}$ would result in an infinite ontology of the form $\{A \sqsubseteq \exists r.\exists r.\exists r. \ldots\}$. A solution to this problem is to use fixpoint operators in the resulting ontology [19,12]. Fixpoints are not a common formalism for description logics, but can be simulated in classical description logics using additional concept symbols, and can serve as a basis for approximating of the result of forgetting [12].

As is shown in [18,20], if finite and if fixpoints are not used in the result, already for the description logics $\mathcal{EL}$ and $\mathcal{ALC}$, the result of forgetting can be of size triple exponential in the size of the input. By using fixpoint operators, we obtain a double-exponential upper-bound, which also holds for the description logic $\mathcal{SIF}$. Since this is still of high complexity, a goal-oriented approach is required in order to be able to compute uniform interpolants practically. Practicality has been achieved in [10,13,14,16] by using a saturation-based approach, which eliminates concept symbols by resolving on these symbols. This approach is also followed in this paper. Saturation-based reasoning has recently received increased interest in the description logic community, due to the success of consequence-based reasoning methods for classification [6,22,24,25,26]. However, these methods are optimised for specific reasoning tasks, and can not directly be used for forgetting.

In this paper, we present a new sound and refutationally complete saturation-procedure for $\mathcal{SIF}$, and show that it can be used for forgetting concept symbols in a goal-oriented manner. The method is based on the methods presented

in [12,10], which we extend to incorporate transitive roles, inverse roles and functional role restrictions.

2 Definition of $\mathcal{SIF}\nu$ and Forgetting

We define the description logic $\mathcal{SIF}\nu$, which is $\mathcal{SIF}$ extended with fixpoint operators.

Let N_c, N_r, N_i and N_v be four pair-wise disjoint sets of respectively *concept symbols*, *role symbols*, *individuals* and *concept variables*. A *role* is either of the form r or r^-, where $r \in N_r$. We define the *inverse of a role* $\mathsf{Inv}(R)$ as $\mathsf{Inv}(r) = r^-$ and $\mathsf{Inv}(r^-) = r$. An *RBox* $\mathcal{R}$ is a set of *transitivity axioms* $\mathsf{trans}(R)$, where R is a role. A role R is *transitive* in $\mathcal{R}$ if $\mathsf{trans}(R) \in \mathcal{R}$ or $\mathsf{trans}(\mathsf{Inv}(R)) \in \mathcal{R}$.

$\mathcal{SIF}\nu$-concepts have the following form:

$$\bot \mid A \mid X \mid \neg C \mid C_1 \sqcup C_2 \mid \exists R.C \mid \leq 1R.\top \mid \nu X.C[X],$$

where $A \in N_c$, $X \in N_v$, C, C_1 and C_2 are arbitrary concepts and R is any role, and $C[X]$ is a concept expression in which X occurs under an even number of negations. We define further concept expressions as abbreviations: $\top = \neg\bot$, $C_1 \sqcap C_2 = \neg(\neg C_1 \sqcup \neg C_2)$, $\forall R.C = \neg(\exists R.\neg C)$, $\geq 2R.\top = \neg\leq 1R.\top$. Concepts of the form $\leq 1R.\top$ are called *functional role restrictions*. Concepts of the form $\nu X.C[X]$ are called *greatest fixpoint expressions*. $\nu X.C[X]$ denotes the *greatest fixpoint* of $C[X]$, and ν is the *greatest fixpoint operator*. A concept variable X is *bound* if it occurs in the scope $C[X]$ of a fixpoint expression $\nu X.C[X]$. Otherwise it is *free*. A concept is *closed* if it does not contain any free variables, otherwise it is *open*.

A *TBox* $\mathcal{T}$ is a set of *concept axioms* of the forms $C \sqsubseteq D$ (*concept inclusion*) and $C \equiv D$ (*concept equivalence*), where C and D are closed concepts. $C \equiv D$ is short-hand for the two concept axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. We further require greatest fixpoint expressions to occur only *positively* in an axiom, that is, on the right-hand side of a concept inclusion and only under an even number of negations. An *ontology* $\mathcal{O} = \langle \mathcal{T}, \mathcal{R} \rangle$ consists of a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$ with the additional restriction that roles that are transitive in $\mathcal{R}$ do not occur under functional role restrictions. This is a common restriction that has been used to guarantee decidability of common reasoning tasks for description logics with number restrictions [5] , and our forgetting method assumes that it is satisfied.

In the definition of the semantics of $\mathcal{SIF}\nu$, an *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of the *domain* $\Delta^{\mathcal{I}}$, a nonempty set, and the *interpretation function* $\cdot^{\mathcal{I}}$, which assigns to each concept symbol $A \in N_c$ a subset of $\Delta^{\mathcal{I}}$ and to each role symbol $r \in N_r$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to $\mathcal{SIF}\nu$-concepts as follows.

$$\bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\geq nr.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#\{(x,y) \in r^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\} \geq n\}$$

The semantics of fixpoint expressions is defined following [2]. Whereas concept symbols are assigned fixed subsets of the domain, concept variables range over

arbitrary subsets, which is why only closed concepts have a fixed interpretation. Open concepts are interpreted using *valuations* ρ that map concept variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation ρ and a set $W \subseteq \Delta^{\mathcal{I}}$, $\rho[X \mapsto W]$ denotes a valuation identical to ρ except that $\rho[X \mapsto W](X) = W$. Given an interpretation $\mathcal{I}$ and a valuation ρ, the function $\cdot^{\mathcal{I}}_{\rho}$ is $\cdot^{\mathcal{I}}$ extended with the cases $X^{\mathcal{I}}_{\rho} = \rho(X)$ and

$$(\nu X.C)^{\mathcal{I}}_{\rho} = \bigcup\{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I}}_{\rho[X \mapsto W]}\}.$$

If C is closed, we define $C^{\mathcal{I}} = C^{\mathcal{I}}_{\rho}$ for any valuation ρ. Since C does not contain any free variables in this case, this defines $C^{\mathcal{I}}$ uniquely.

A concept inclusion $C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and a transitivity axiom $\mathsf{trans}(R)$ is true in $\mathcal{I}$ if for any domain elements $x, y, z \in \Delta^{\mathcal{I}}$ we have $(x, z) \in R^{\mathcal{I}}$ if $(x, y) \in R^{\mathcal{I}}$ and $(y, z) \in R^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ if all axioms in $\mathcal{O}$ are true in $\mathcal{I}$. An ontology $\mathcal{O}$ is *satisfiable* if a model exists for $\mathcal{O}$, otherwise it is *unsatisfiable*. Two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are *equi-satisfiable* if every model of $\mathcal{T}_1$ can be extended to a model of $\mathcal{T}_2$, and vice versa. $\mathcal{T} \models C \sqsubseteq D$ holds iff in every model $\mathcal{I}$ of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. If an axiom α is true in all models of $\mathcal{O}$, we write $\mathcal{O} \models \alpha$.

Let $sig(E)$ denote the concept and role symbols occurring in E, where E can denote a concept, an axiom, a TBox, an RBox or an ontology.

Definition 1 (Forgetting). *Let A be a concept symbol and $\mathcal{O}$ and $\mathcal{O}^{-A}$ be ontologies. An ontology $\mathcal{O}^{-A}$ is a* result of forgetting *A from $\mathcal{O}$ if the following conditions hold:*

1. *$A \notin sig(\mathcal{O}^{-A})$, and*
2. *for all concept inclusions α with $A \notin sig(\alpha)$: $\mathcal{O}^{-A} \models \alpha$ iff $\mathcal{O} \models \alpha$.*

Given an ontology $\mathcal{O}$ and a set of concept symbols $\mathcal{S}$, a result of forgetting $\mathcal{S}$ from $\mathcal{O}$ is a result of forgetting each symbol in $\mathcal{S}$ one by one. An ontology $\mathcal{O}^{\mathcal{S}}$ is a uniform interpolant of $\mathcal{O}$ for $\mathcal{S}$ *iff $\mathcal{O}$ is a result of forgetting every symbol from $\mathcal{O}$ that is not in $\mathcal{S}$.*

3 Normalisation

The saturation method works on ontologies of a specific normal form, which is defined as follows.

Definition 2. *Let $N_d \subseteq N_c$ be a set of designated concept symbols called* definer symbols *or, simply,* definers*. A $\mathcal{SIF}$* literal *is a concept of one of the following forms:*

$$A \mid \neg A \mid \exists R.D \mid \forall R.D \mid \geq 2R.\top \mid \leq 1R.\top,$$

where $A \in N_c$, $D \in N_d$, and R is of the form r or r^-, with $r \in N_r$. A $\mathcal{SIF}$ clause *is a transitivity axiom or an axiom of the form $\top \sqsubseteq L_1 \sqcup \ldots \sqcup L_n$, where $L_1, \ldots, L_n$ are $\mathcal{SIF}$ literals. We may omit the leading "$\top \sqsubseteq$" in clauses, and*

Non-Cyclic Definer Elimination:

$$\frac{\mathcal{O} \cup \{D \sqsubseteq C\}}{\mathcal{O}^{[D/C]}} \quad \text{provided } D \notin \mathit{sig}(C)$$

Definer Purification:

$$\frac{\mathcal{O}}{\mathcal{O}^{[D/\top]}} \quad \text{provided } D \text{ occurs only positively in } \mathcal{O}$$

Cyclic Definer Elimination:

$$\frac{\mathcal{O} \cup \{D \sqsubseteq C[D]\}}{\mathcal{O}^{[D/\nu X.C[X]]}} \quad \text{provided } D \in \mathit{sig}(C[D])$$

Fig. 1. Rules for eliminating definer symbols

assume clauses are represented as sets, that is, they do not contain duplicate elements and the order is not important.

An ontology $\mathcal{N}$ is in $\mathcal{SIF}$ normal form *if every axiom in $\mathcal{N}$ is a $\mathcal{SIF}$ clause, and if for every clause $\mathsf{trans}(R) \in \mathcal{N}$, there is also a clause $\mathsf{trans}(\mathsf{Inv}(R)) \in \mathcal{N}$.*

Note that the description logic $\mathcal{SIF}$ allows for number restrictions of the form $\geq 2R.\top$, since $\mathcal{SIF}$ concepts are closed under negation, and $\geq 2R.\top \equiv \neg(\leq 1R.\top)$.

Any $\mathcal{SIF}\nu$ ontology can be transformed into an ontology in $\mathcal{SIF}$ normal form using standard structural transformation and CNF transformation techniques.

Example 1. Consider the following ontology $\mathcal{O}_1$:

$$A_1 \sqcap B_1 \sqsubseteq \bot \qquad A_1 \sqsubseteq \exists r^-.B_2 \qquad B_2 \sqsubseteq \leq 1r.\top \qquad B_2 \sqsubseteq \exists r.(B_1 \sqcup A_2)$$

The $\mathcal{SIF}$ normal form of $\mathcal{O}_1$ is the following set of clauses:

1. $\neg A_1 \sqcup \neg B_1$
2. $\neg A_1 \sqcup \exists r^-.D_1$
3. $\neg D_1 \sqcup B_2$
4. $\neg B_2 \sqcup \leq 1r.\top$
5. $\neg B_2 \sqcup \exists r.D_2$
6. $\neg D_2 \sqcup B_1 \sqcup A_2$

Given a set $\mathcal{N}$ of $\mathcal{SIF}$ clauses such that every clause contains at most one literal of the form $\neg D$, where $D \in N_d$, it is possible to eliminate all definer literals in $\mathcal{N}$ using the rewrite rules shown in Figure 1. This is crucial for our method for forgetting concept symbols, since the method described in the next section operates on sets of $\mathcal{SIF}$ clauses. For the rules in Figure 1, it is assumed that for every definer D occurring negatively in $\mathcal{N}$, the set of clauses of the form $\neg D \sqcup C_1$, $\ldots$, $\neg D \sqcup C_n$ is replaced by a single axiom $D \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$. Note that the last rule in Figure 1 introduces fixpoint operators to the ontology. The rules in Figure 1 are applications of Ackermann's Lemma [1] and its generalisation [21], which have been used in the context of second-order quantifier elimination to eliminate predicate symbols [3]. The result of applying these rules does not contain any definers, but preserves all entailments of input that do not involve definer symbols, which is a consequence of these lemmata.

Transitivity Rule:

$$\frac{C \sqcup \forall R.D \qquad \mathsf{trans}(R)}{C \sqcup \forall R.D_{\mathsf{trans}} \quad \neg D' \sqcup D \quad \neg D' \sqcup \forall R.D_{\mathsf{trans}}}$$

Universalisation Rule:

$$\frac{C_1 \sqcup \exists R.D \qquad C_2 \sqcup \leq 1R.\top}{C_1 \sqcup C_2 \sqcup \forall R.D}$$

Fig. 2. Rules used in the first stage of reasoning.

4 The Calculus

In order to forget a concept symbol A, we infer all inferences on that symbol using a saturation-based approach, and then eliminate occurrences of that symbol from the resulting clause set. Using the transformation rules in Figure 1, we can then eliminate all definer symbols introduced by the normalisation or throughout the reasoning process.

Due to possible interactions between the rules of our calculus, we process the clause set in several stages, where only certain rules are allowed in each stage. This is a major difference between this method and the methods for $\mathcal{ALC}$ and $\mathcal{ALCH}$ presented in [12,10], and is necessary to guarantee termination of the method. In the first stage, we only apply rules that infer clauses with universal restrictions. The function of this stage is to infer all clauses that can serve as premise in the second stage of the calculus. In the second stage, we handle inverse roles in universal restrictions using the role inversion rule. In the last stage, we apply all remaining inferences of the calculus with the aim of computing inferences on A. We describe the stages one by one, and illustrate them on the example introduced in the last section.

Stage 1. The rules for the first stage are shown in Figure 2. The transitivity rule is directly taken from [13], which presents a similar calculus for the description logic $\mathcal{SHQ}$. The rule works in a similar fashion as common rewriting rules to reduce reasoning with transitivity roles to reasoning without (see for example [27,23]).

The universalisation rule infers from a functional role restriction and an existential role restriction a universal restriction. The soundness of this rule can be explained as follows. If a domain element x in a model $\mathcal{I}$ has an R-successor that satisfies D ($x \in (\exists R.D)^{\mathcal{I}}$), and if x has maximally one R-successor ($x \in (\leq 1R.\top)^{\mathcal{I}}$), then every R-successor of x satisfies D, since there is only one such successor ($x \in (\forall R.D)^{\mathcal{I}}$). This means $(\exists R.D \sqcap \leq 1R.\top) \models \forall R.D$, and implies that the universalisation rule is sound.

Role Inversion Rule:

$$\frac{C \sqcup \forall R.D}{D \sqcup \forall \mathsf{Inv}(R).D_{\mathsf{Inv}} \qquad \neg D_{\mathsf{Inv}} \sqcup C}$$

Fig. 3. Role inversion rule used in the second stage of reasoning.

Resolution Rule:

$$\frac{C_1 \sqcup A \qquad C_2 \sqcup \neg A}{C_1 \sqcup C_2}$$

$\forall$-Rule:

$$\frac{C_1 \sqcup \forall R.D_1 \qquad C_2 \sqcup \mathsf{Q}R.D_2}{C_1 \sqcup C_2 \sqcup \mathsf{Q}R.D_{12}}$$

where $\mathsf{Q} \in \{\forall, \exists\}$ and D_{12} is a possibly new definer representing $D_1 \sqcap D_2$.

$\exists$-Rule:

$$\frac{C_1 \sqcup \exists R.D_1 \qquad C_2 \sqcup \exists R.D_2}{C_1 \sqcup C_2 \sqcup \exists R.D_{12} \sqcup \geq 2R.\top}$$

where D_{12} is a possibly new definer representing $D_1 \sqcap D_2$.

Fig. 4. Rules used in the third stage of reasoning.

Example 2. Following the clause set constructed in the last example, in Stage 1, we apply the universalisation rule on Clause 4 and 5 to infer the following clause.

$$7.\ \neg B_2 \sqcup \forall r.D_2 \qquad \textit{(Universalisation 4, 5)}$$

Stage 2. In this stage, the inversion rule shown in Figure 3 is applied.

Example 3. Continuing the previous example, there is only one clause in the current clause set that has a universal role restriction, which is Clause 7 that was derived in Stage 1. By applying the role inversion rule, the following new clauses are derived:

$$8.\ D_2 \sqcup \forall r^{-}.D_{\mathsf{Inv}} \qquad \textit{(Role Inversion 7)}$$
$$9.\ \neg D_{\mathsf{Inv}} \sqcup \neg B_2 \qquad \textit{(Role Inversion 7)}$$

Stage 3. This is the main reasoning stage, where we compute all inferences on the symbols we want to forget. The rules for this stage are shown in Figure 4. The rules of this stage are an extension of the calculus used for forgetting presented in [12,10]. The $\exists$-rule is influenced by the calculus for reasoning in $\mathcal{SHQ}$ presented in [13].

A key for ensuring termination is the dynamic introduction of new symbols through the rules of the calculus. This is necessary to preserve the normal form

and still be able to infer all clauses that are required for the forgetting result. There are two rules in the first stage that introduce new definers: the transitivity rule and the role inversion rule. In the third stage, the $\forall$-rule and the $\exists$-rule may introduce new definers that represent conjunctions of existing definers. More specifically, given two definers D_1 and D_2, we may introduce a new definer D_{12} representing $D_1 \sqcap D_2$ by adding the clauses $\neg D_{12} \sqcup D_1$ and $\neg D_{12} \sqcup D_2$. By doing this in an optimised way, the number of definer symbols introduced can be restricted to by 2^n, where n is the number of definer symbols present in the input clause set [12].

The resolution rule is standard in resolution-based reasoning. The $\forall$-rule are directly taken from [12]. The $\forall$-rule propagates concepts below a universal role restrictions into other role restrictions. If $\forall R.D_1$ is satisfied by some element x of the domain, we know that all R-successors of x have to satisfy D_1. This implies that, if x furthermore satisfies $\mathsf{Q}R.D_2$ for some $\mathsf{Q} \in \{\exists, \forall\}$, then it also satisfies $\mathsf{Q}R.(D_1 \sqcap D_2)$.

The $\exists$-rule is new and necessary in order to preserve entailments that use a $\geq$2-restriction. Assume we have a model with a domain element x that has at least one R-successor x_1 satisfying D_1 and at least one R-successor x_2 that satisfies D_2. If $x_1 = x_2$, then x satisfies $\exists R.(D_1 \sqcap D_2)$. If $x_1 \neq x_2$, then x satisfies $\geq 2R.\top$. Hence, we have $(C_1 \sqcup \exists R.D_1) \sqcap (C_2 \sqcup \exists R.D_2) \models (C_1 \sqcup C_2 \sqcup \exists R.(D_1 \sqcap D_2) \sqcup \geq 2R.\top)$, and the $\exists$-rule is sound.

In order to forget a concept symbol A, we compute all resolvents on A and on definer literals that lead to clauses with maximally one negative definer literal. Clauses with multiple negative definer literals are not needed for the computation, which can be argued in similar ways as in [12].

Note that even though only the resolution rule directly infers clauses on a concept symbol, the $\forall$- and the $\exists$-rule may introduce new definers, which subsequently makes new inferences on the symbol to be forgotten possible. For the forgetting result, only inferences with maximally one negative definer literal are required. The role inversion rule may derive clauses with more than one negative definer literal, but these inferences are only required if they allow for further inferences of clauses with maximally one negative definer.

In the resulting clause set, we omit all clauses containing A or more than one negative definer literal. Then, we eliminate all definers using the technique described in the last section. The ontology obtained is the result of forgetting A from the original ontology.

Example 4. Assume we want to forget B_1 and B_2. We begin by computing inferences on B_1, for which only one inference step is needed.

$$10.\ \neg D_2 \sqcup \neg A_1 \sqcup A_2 \qquad \textit{(Resolution 1, 6)}$$

We continue by computing inferences on B_2. This time, in order to make all inferences possible, we have to use the $\forall$-rule first.

11. $\neg D_1 \sqcup \leq 1 r.\top$	*(Resolution 3, 4)*
12. $\neg D_1 \sqcup \exists r.D_2$	*(Resolution 3, 5)*
13. $\neg D_1 \sqcup \forall r.D_2$	*(Resolution 3, 7)*
14. $\neg A_1 \sqcup D_2 \sqcup \exists r^-.D_{1\mathsf{Inv}}$	*($\forall$-rule 2, 8)*
15. $\neg D_{1\mathsf{Inv}} \sqcup D_1$	*($D_{1\mathsf{Inv}} \sqsubseteq D_1 \sqcap D_{\mathsf{Inv}}$)*
16. $\neg D_{1\mathsf{Inv}} \sqcup D_{\mathsf{Inv}}$	*($D_{1\mathsf{Inv}} \sqsubseteq D_1 \sqcap D_{\mathsf{Inv}}$)*
17. $\neg A_1 \sqcup A_2 \sqcup \exists r^-.D_{1\mathsf{Inv}}$	*(Resolution 10, 14)*
18. $\neg D_{1\mathsf{Inv}} \sqcup B_2$	*(Resolution 3, 15)*
19. $\neg D_{1\mathsf{Inv}} \sqcup \neg B_2$	*(Resolution 9, 16)*
20. $\neg D_{1\mathsf{Inv}}$	*(Resolution 18, 19)*

Even though we did not discuss redundancy elimination techniques here, one can easily see that further inferences of clauses of the form $\neg D_{1\mathsf{Inv}} \sqcup C$ are not needed, since we already inferred the unary clause $\neg D_{1\mathsf{Inv}}$. For the same reason, we do not have to include any other clause containing $\neg D_{1\mathsf{Inv}}$ in the result. In fact, no further inferences are necessary for the forgetting result. If we ignore all clauses that do contain the symbols B_1 and B_2 we are forgetting, we are left with Clauses 2, 7, 10–13, 17 and 20. Eliminating the definers in these clauses results in the following ontology:

$$A_1 \sqsubseteq \exists r^-.(\leq 1 r.\top \sqcap \exists r.(\neg A_1 \sqcup A_2) \sqcap \forall r.(\neg A_1 \sqcup A_2))$$
$$A_1 \sqsubseteq A_2 \sqcup \exists r^-.\bot$$

We can simplify the second axiom to $A_1 \sqsubseteq A_2$, which allows us to further simplify the first axiom, and obtain as result of forgetting B_1 and B_2 the following:

$$A_1 \sqsubseteq \exists r^-.(\leq 1 r.\top \sqcap \exists r.\top) \qquad\qquad A_1 \sqsubseteq A_2$$

5 Correctness of the Method

In order to prove that the method is correct, we show that the resulting ontology preserves all entailments of axioms that do not use the symbols to be forgotten. More formally, if $\mathcal{O}^{-A}$ denotes the output of our method for an ontology $\mathcal{O}$ and a concept symbol A, we show that $\mathcal{O}^{-A} \models \alpha$ iff $\mathcal{O} \models \alpha$ for all axioms α with $A \notin \mathit{sig}(\alpha)$. If $\alpha = C_1 \sqsubseteq C_2$, we can prove $\mathcal{O} \models \alpha$ by showing that $\mathcal{O} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$ is unsatisfiable, where r^* is a role not occurring in $\mathcal{O}$.

In order to show that our forgetting method works correctly, we extend our reasoning method to a refutational complete calculus, that, in order to prove the satisfiability of a clause set, first infers inferences on a designated concept symbol A using only the rules of the forgetting method. If this calculus is refutational complete, we have that a contradiction can be derived from $\mathcal{O} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$

$\exists$-Elimination Rule:

$$\frac{C_1 \sqcup \exists R.D \qquad \neg D}{C_1}$$

$\geq$2-Elimination Rule I:

$$\frac{C_1 \sqcup \geq 2R.\top \qquad C_2 \sqcup \leq 1R.\top}{C_1 \sqcup C_2}$$

$\geq$2-Elimination Rule II:

$$\frac{C_1 \sqcup \geq 2R.\top \qquad C_2 \sqcup \forall R.D \qquad \neg D}{C_1 \sqcup C_2}$$

Fig. 5. Additional rules needed for refutational completeness.

exactly in the same cases as it can from $\mathcal{O}^{-A} \cup \{\exists r^*.(C_1 \sqcap \neg C_2)\}$. This implies $\mathcal{O}^{-A} \models C_1 \sqsubseteq C_2$ iff $\mathcal{O} \models C_1 \sqsubseteq C_2$ for all concept inclusions $C_1 \sqsubseteq C_2$ with $A \notin sig(C_1 \sqsubseteq C_2)$, as required.

In order to obtain a refutationally complete calculus, we additionally need the rules shown in Figure 5. Whereas the rules for the forgetting procedure are aimed at making inferences on concept symbols possible, they are not sufficient for detecting whether a set of clauses is unsatisfiable. The rules in Figure 5 are aimed at detecting inconsistencies between clauses and eliminating unsatisfiable literals, and transform the set of rules into a decision procedure for $\mathcal{SIF}$-ontology satisfiability.

The $\exists$-elimination rule eliminates unsatisfiable literals of the form $\exists R.D$. The $\geq$2-elimination rule I resolves on literals of the form $\geq 2R.\top$ and $\leq 1R.\top$. The rule is sound since an individual can only satisfy $\geq 2R.\top$ or $\leq 1R.\top$ at the same time. The $\geq$2-elimination rule II eliminates unsatisfiable literals, similarly to the $\exists$-elimination rule. If the definer D is unsatisfiable, any instance of $\forall R.D$ cannot have R-successors. Therefore, we can resolve between $\forall R.D$ and $\geq 2R.\top$.

We obtain the reasoning procedure $Ref_{\mathcal{SIF}}$ by extending the forgetting procedure presented in the last section by the rules in Figure 5, which are applied in the last reasoning stage. We further refine the calculus with an ordering. The reasoning procedure $Ref^A_{\mathcal{SIF}}$ uses the same rules as $Ref_{\mathcal{SIF}}$, but for clauses containing the concept symbol A, it only performs inferences on literals of the form A or $\neg A$.

We have the following theorem.

Theorem 1. *Let A be any concept symbol, $Ref_{\mathcal{SIF}}$ and $Ref^A_{\mathcal{SIF}}$ are sound and refutationally complete, that is, for any set $\mathcal{N}$ of clauses, one can infer the empty clause iff $\mathcal{N}$ is inconsistent.*

Proof (Sketch). In order to prove refutational completeness, we have to show that we can build a model based on any saturated set $\mathcal{N}^*$ of clauses such that $\bot \notin \mathcal{N}^*$.

Such a model can be obtained by adapting the model construction presented in [10]. This construction first constructs a model fragment for each definer, and then connects these elements to a complete model, where each definer is represented by a designated domain element. Different to $\mathcal{ALCH}$, $\mathcal{SIF}$ does not have the finite model property. By unravelling the possibly cyclic models created in [10] to a possibly infinite tree, it is however possible to construct a model for $\mathcal{N}^*$, which can be verified by a careful analysis of the cases used in the proof in [10]. □

This theorem allows us to establish that the forgetting procedure described in the last section is correct.

Theorem 2. *For any $\mathcal{SIF}$-ontology $\mathcal{O}$ and any concept symbol A, the described method always terminates and computes the result of forgetting A from $\mathcal{O}$. A uniform interpolant for any ontology $\mathcal{O}$ and signature $\mathcal{S}$ with $N_r \subseteq \mathcal{S}$ can be computed by step-wise forgetting every symbol in $sig(\mathcal{O}) \setminus \mathcal{S}$.*

6 Conclusion and Future Work

We described a method for forgetting concept symbols from ontologies formulated in the description logic $\mathcal{SIF}$, where results are represented in $\mathcal{SIF}\nu$. Forgetting eliminates a specified set of symbols from an ontology in such a way, that all entailments that do not use these symbols are preserved. The method uses a resolution-based saturation procedure to compute inferences on the symbols to be eliminated. By extending this saturation procedure to a refutationally complete reasoning method, which performs inferences on the symbols to be forgotten first, we could prove that our method indeed preserves all entailments in the desired signature.

In order to properly handle the interactions between functional role restrictions and inverse roles without losing termination, the method works in three stages. In the first stage all clauses with a universal restriction are computed, on which in the second stage inferences based on inverse roles are performed. An interesting question is whether this approach can also be used in connection with role hierarchies or with cardinality restrictions. A method for forgetting in the description logic $\mathcal{SHQ}$ is presented in [13]. A simple combination of the rules presented in this paper and presented in [13] is not sufficient to obtain a refutational complete method already for $\mathcal{SHIF}$. An open question is whether this also affects the appropriateness of such a calculus for forgetting concept symbols, and whether a sufficient calculus can be developed by adding additional rules and possibly further extending the underlying description logic, for example with role conjunctions.

We are working on a prototypical implementation of the method, which we aim to integrate into our forgetting tool LETHE.[1] Experiments on similar methods for $\mathcal{ALC}$, $\mathcal{ALCH}$ and $\mathcal{SHQ}$ had promising results [11,10,13,15], and we are confident that similar results can be obtained for $\mathcal{SIF}$.

[1] `www.cs.man.ac.uk/~koopmanp/lethe`

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110(1), 390–413 (1935)
2. Calvanese, D., De Giacomo, G., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99. pp. 84–89. Morgan Kaufmann (1999)
3. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications, Studies in Logic, vol. 12. College Publications (2008)
4. Grau, B.C.: Privacy in ontology-based information systems: A pending matter. Semantic Web 1(1-2), 137–141 (2010)
5. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8(3), 239—263 (2000)
6. Kazakov, Y.: Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In: Boutilier, C. (ed.) Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 2040–2045. AAAI Press (2009)
7. Konev, B., Ludwig, M., Walther, D., Wolter, F.: The logical difference for the lightweight description logic $\mathcal{EL}$. Journal of Artificial Intelligence Research 44(1), 633–708 (2012)
8. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 5195, pp. 259–274. Springer (2008)
9. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proceedings of the Internation Conference on Artificial Intelligence (IJCAI-09). pp. 830–835. AAAI Press (2009)
10. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: Logic for Programming, Artificial Intelligence and Reasoning. Lecture Notes in Computer Science, vol. 8312, pp. 552–567. Springer (2013)
11. Koopmann, P., Schmidt, R.A.: Implementation and evaluation of forgetting in $\mathcal{ALC}$-ontologies. In: Proceedings of the 7th International Workshop on Modular Ontologies (WoMO'13). CEUR Workshop Proceedings, vol. 1081, pp. 37–48. CEUR-WS.org (2013)
12. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Frontiers of Combining Systems. Lecture Notes in Computer Science, vol. 8152, pp. 87–102. Springer (2013)
13. Koopmann, P., Schmidt, R.A.: Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Automated Reasoning. Lecture Notes in Computer Science, vol. 8562, pp. 434–448. Springer (2014)
14. Koopmann, P., Schmidt, R.A.: Forgetting and uniform interpolation for $\mathcal{ALC}$-ontologies with ABoxes. In: Proceedings of the 27th International Workshop of Description Logics (DL 2014). CEUR Workshop Proceedings, vol. 1193, pp. 245–257. CEUR-WS.org (2014)
15. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). vol. 1, pp. 175–181. AAAI-Press (2015)
16. Ludwig, M., Konev, B.: Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In: Proc. KR'14. AAAI Press (2014)

17. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference (KR-12). pp. 286–296. AAAI Press (2012)
18. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-11). pp. 989–995. AAAI Press (2011)
19. Nikitina, N.: Forgetting in general $\mathcal{EL}$ terminologies. In: Proceedings of the 2011 International Workhop on Description Logics (DL2011). CEUR Workshop Proceedings, vol. 745, pp. 345–355. CEUR-WS.org (2011)
20. Nikitina, N., Rudolph, S.: (Non-) Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. Artificial Intelligence 215, 120–140 (2014)
21. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Orlowska, E. (ed.) Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa, Studies in Fuzziness and Soft Computing, vol. 24, pp. 307–328. Springer (1999)
22. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR-10) (2010)
23. Schmidt, R.A., Hustadt, U.: A principle for incorporating axioms into the first-order translation of modal formulae. In: Automated Deduction–CADE-19, pp. 412–426. Springer (2003)
24. Simancík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11). vol. 22, pp. 1093–1098. AAAI Press (2011)
25. Simancík, F., Motik, B., Krötzsch, M.: Fixed parameter tractable reasoning in DLs via decomposition. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011). CEUR Workshop Proceedings, vol. 745, pp. 400–410. CEUR-WS.org (2011)
26. Steigmiller, A., Glimm, B., Liebig, T.: Coupling tableau algorithms for expressive description logics with completion-based saturation procedures. In: Automated Reasoning, Lecture Notes of Computer Science, vol. 8562, pp. 449–463. Springer (2014)
27. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)
28. Wang, K., Wang, Z., Topor, R.W., Pan, J.Z., Antoniou, G.: Eliminating concepts and roles from ontologies in expressive descriptive logics. Computational Intelligence 30(2), 205–232 (2014)
29. Wang, Z., Wang, K., Topor, R.W., Pan, J.Z.: Forgetting for knowledge bases in DL-Lite. Annals of Mathematics and Artificial Intelligence 58(1–2), 117–151 (2010)

Incremental Learning of TBoxes from Interpretation Sequences with Methods of Formal Concept Analysis

Francesco Kriegel

Institute for Theoretical Computer Science, TU Dresden, Germany
francesco.kriegel@tu-dresden.de
http://lat.inf.tu-dresden.de/~francesco

Abstract. Formal Concept Analysis and its methods for computing minimal implicational bases have been successfully applied to axiomatise minimal $\mathcal{EL}$-TBoxes from models, so called bases of GCIs. However, no technique for an adjustment of an existing $\mathcal{EL}$-TBox w.r.t. a new model is available, i.e., on a model change the complete TBox has to be recomputed. This document proposes a method for the computation of a minimal extension of a TBox w.r.t. a new model. The method is then utilised to formulate an incremental learning algorithm that requires a stream of interpretations, and an expert to guide the learning process, respectively, as input.

Keywords: description logics, formal concept analysis, base of GCIs, implicational base, TBox extension, incremental learning

1 Introduction

More and more data is generated and stored thanks to the ongoing technical development of computers in terms of processing speed and storage space. There is a vast number of databases, some of them freely available on the internet (e.g., `dbpedia.org` and `wikidata.org`), that are used in research and industry to store assertional knowledge, i.e., knowledge on certain objects and individuals. Examples are databases of online stores that besides contact data also store purchases and orders of their customers, or databases which contain results from experiments in biology, physics, psychology etc. Due to the large size of these databases it is difficult to quickly derive conclusions from the data, especially when only *terminological knowledge* is of interest, i.e., knowledge that does not reference certain objects or individuals but holds for all objects or individuals in the dataset.

So far there have been several successful approaches for the combination of description logics and formal concept analysis as follows. In [5, 6, 28] Baader, Ganter, and Sertkaya have developed a method for the completion of ontologies by means of the exploration algorithm for formal contexts. Rudolph has invented an exploration algorithm for $\mathcal{FLE}$-interpretations in [26, 27]. Furthermore, Baader and Distel presented in [3, 4, 12] a technique for the computation of bases of concept inclusions for finite interpretations in $\mathcal{EL}$ that has been extended with error-tolerance by Borchmann in [7, 8]. Unfortunately, none of these methods and algorithms provide the possibility of extension or adaption of an already existing ontology (or TBox). Hence, whenever

a new dataset (in form of an interpretation or description graph) is observed then the whole base has to be recomputed completely which can be a costly operation, and moreover the changes are not explicitely shown to the user. In this document we propose an extension of the method of Baader and Distel in [3, 4, 12] that allows for the construction of a minimal extension of a TBox w.r.t. a model. The technique is then utilised to introduce an incremental learning algorithm that requires a stream of interpretations as input, and uses an expert to guide to exploration process.

In Sections 2 and 3 we introduce the neccessary notions from description logics, and formal concept analysis, respectively. Section 4 presents the results on bases of GCIs for interpretations relative to a TBox. Section 5 defines experts and adjustments that are neccessary to guide the incremental learning algorithm that is shown in Section 6. Finally, Section 7 compares the incremental learning approach with the existing single-step learning approach.

2 The Description Logic $\mathcal{EL}^\bot$

At first we introduce the light-weight description logic $\mathcal{EL}^\bot$. Let (N_C, N_R) be an arbitrary but fixed signature, i.e., N_C is a set of *concept names* and N_R is a set of *role names*. Every concept name $A \in N_C$, the *top concept* $\top$, and the *bottom concept* $\bot$ are $\mathcal{EL}^\bot$-concept descriptions. When C and D are $\mathcal{EL}^\bot$-concept descriptions and $r \in N_R$ is a role name then also the *conjunction* $C \sqcap D$ and the *existential restriction* $\exists r.C$ are $\mathcal{EL}^\bot$-concept descriptions. We denote the set of all $\mathcal{EL}^\bot$-concept descriptions over (N_C, N_R) by $\mathcal{EL}^\bot(N_C, N_R)$.

The semantics of $\mathcal{EL}^\bot$ are defined by means of interpretations. An *interpretation* $\mathcal{I}$ over (N_C, N_R) is a pair $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a set $\Delta^\mathcal{I}$, called *domain*, and an *extension function* $\cdot^\mathcal{I} \colon N_C \cup N_R \to 2^{\Delta^\mathcal{I}} \cup 2^{\Delta^\mathcal{I} \times \Delta^\mathcal{I}}$ that maps concept names $A \in N_C$ to subsets $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and role names $r \in N_R$ to binary relations $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. Furthermore, the extension function is then canonically extended to all $\mathcal{EL}^\bot$-concept descriptions:

$$
\begin{aligned}
(C \sqcap D)^\mathcal{I} &:= C^\mathcal{I} \cap D^\mathcal{I} \\
(\exists r.C)^\mathcal{I} &:= \left\{ d \in \Delta^\mathcal{I} \;\middle|\; \exists e \in \Delta^\mathcal{I} \colon (d,e) \in r^\mathcal{I} \wedge e \in C^\mathcal{I} \right\}
\end{aligned}
$$

$\mathcal{EL}^\bot$ allows to express terminological knowledge with so called *concept inclusions*. A *general concept inclusion* (abbr. *GCI*) in $\mathcal{EL}^\bot$ over (N_C, N_R) is of the form $C \sqsubseteq D$ where C and D are $\mathcal{EL}^\bot$-concept descriptions over (N_C, N_R). An *$\mathcal{EL}^\bot$-TBox* $\mathcal{T}$ is a set of $\mathcal{EL}^\bot$-GCIs. An interpretation $\mathcal{I}$ is a *model* of an $\mathcal{EL}^\bot$-GCI $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$, if the set inclusion $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds; and $\mathcal{I}$ is a *model* of an $\mathcal{EL}^\bot$-TBox $\mathcal{T}$, symbolized as $\mathcal{I} \models \mathcal{T}$, if it is a model of all its $\mathcal{EL}^\bot$-concept inclusions. An $\mathcal{EL}^\bot$-GCI $C \sqsubseteq D$ *follows* from an $\mathcal{EL}^\bot$-TBox $\mathcal{T}$, denoted as $\mathcal{T} \models C \sqsubseteq D$, if every model of $\mathcal{T}$ is also a model of $C \sqsubseteq D$.

3 Formal Concept Analysis

This section gives a brief overview on the basic definitions of formal concept analysis and the neccessary lemmata and theorems cited in this paper.

The basic structure of formal concept analysis is a *formal context* $\mathbb{K} = (G, M, I)$ that consists of a set G of *objects*, a set M of *attributes*, and an *incidence relation* $I \subseteq G \times M$. Instead of $(g, m) \in I$ we rather use the infix notation $g\ I\ m$, and say that g *has* m. For brevity we may sometimes drop the adjective "formal". From each formal context $\mathbb{K}$ two so-called *derivation operators* arise: For subsets $A \subseteq G$ we define A^I as a subset of M that contains all attributes which the objects in A have in common, i.e.,

$$A^I := \{m \in M \mid \forall g \in A\colon g\ I\ m\}\,.$$

Dually for $B \subseteq M$ we define B^I as the set of all objects that have all attributes in B, i.e., $B^I := \{g \in G \mid \forall m \in B\colon g\ I\ m\}$. An *intent* is a subset $B \subseteq M$ such that $B = B^{II}$ holds.

A *formal implication* over M is of the form $X \to Y$ where $X, Y \subseteq M$. It *holds* in the context (G, M, I) if all objects having all attributes from X also have all attributes from Y, i.e., iff $X^I \subseteq Y^I$ is satisfied. An *implication set* over M is a set of implications over M, and it *holds* in a context $\mathbb{K}$ if all its implications hold in $\mathbb{K}$. An implication $X \to Y$ *follows* from an implication set $\mathcal{L}$ if $X \to Y$ holds in all contexts in which $\mathcal{L}$ holds, or equivalently iff $Y \subseteq X^{\mathcal{L}}$ where $X^{\mathcal{L}}$ is defined as the least superset of X that satisfies the implication $A \subseteq X^{\mathcal{L}} \Rightarrow B \subseteq X^{\mathcal{L}}$ for all implications $A \to B \in \mathcal{L}$.

Stumme has extended the notion of implicational bases as defined by Duquenne and Guigues in [20] and by Ganter in [13] towards background knowledge in form of an implication set. We therefore skip the original definitions and theorems and just cite those from Stumme in [29]. If $\mathcal{S}$ is an implication set holding in a context $\mathbb{K}$, then an *implicational base of* $\mathbb{K}$ *relative to* $\mathcal{S}$ is defined as an implication set $\mathcal{L}$, such that $\mathcal{L}$ holds in $\mathbb{K}$, and furthermore each implication that holds in $\mathbb{K}$ follows from $\mathcal{S} \cup \mathcal{L}$.

A set $P \subseteq M$ is called a *pseudo-intent of* $\mathbb{K}$ *relative to* $\mathcal{S}$ if $P = P^{\mathcal{S}}$, $P \neq P^{II}$, and for each pseudo-intent $Q \subsetneq P$ of $\mathbb{K}$ relative to $\mathcal{S}$ it holds that $Q^{II} \subseteq P$. Then the following set is the *canonical implicational base of* $\mathbb{K}$ *relative to* $\mathcal{S}$:

$$\left\{P \to P^{II} \;\middle|\; P \text{ is a pseudo-intent of } \mathbb{K} \text{ relative to } \mathcal{S}\right\}.$$

4 Relative Bases of GCIs w.r.t. Background TBox

In this section we extend the definition of a base of GCIs for interpretations as introduced by Baader and Distel in [3, 4, 12] towards the possibility to handle background knowledge in form of a TBox. Therefore, we simply assume that there is already a set of GCIs that holds in an interpretation, and are just interested in a minimal extension of the TBox such that the union of the TBox and this *relative base* indeed entails all GCIs which hold in the interpretation.

In the following text we always assume that $\mathcal{I}$ is an interpretation, and $\mathcal{T}$ is a TBox that has $\mathcal{I}$ as a model, and both are defined over the signature (N_C, N_R).

Definition 1 (Relative Base w.r.t. Background TBox). *An $\mathcal{EL}^\perp$-base for $\mathcal{I}$ relative to $\mathcal{T}$ is defined as an $\mathcal{EL}^\perp$-TBox $\mathcal{B}$ that fulfills the following conditions:*

(sound) *All GCIs in $\mathcal{T} \cup \mathcal{B}$ hold in $\mathcal{I}$, i.e., $\mathcal{I} \models \mathcal{T} \cup \mathcal{B}$.*
(complete) *All GCIs that hold in $\mathcal{I}$ also follow from $\mathcal{T} \cup \mathcal{B}$, i.e., $\mathcal{I} \models C \sqsubseteq D$ implies $\mathcal{T} \cup \mathcal{B} \models C \sqsubseteq D$.*

Furthermore, we call $\mathcal{B}$ irredundant, *if none of its concept inclusions follows from the others, i.e., if $(\mathcal{T} \cup \mathcal{B}) \setminus \{C \sqsubseteq D\} \not\models C \sqsubseteq D$ holds for all $C \sqsubseteq D \in \mathcal{B}$; and* minimal, *if it has minimal cardinality among all $\mathcal{EL}^\bot$-bases for $\mathcal{I}$ relative to $\mathcal{T}$. Of course all minimal bases are irredundant but not vice versa.*

The previous definition is a straightforward generalization of bases of GCIs for interpretations since in case of an empty TBox $\mathcal{T} = \emptyset$ both definitions coincide.

The term of a model-based most-specific concept description has been introduced by Baader and Distel in [3, 4, 12]. The next definition extends their notion to model-based most-specific concept descriptions relative to a TBox.

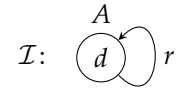
Definition 2 (Relative model-based most-specific concept description). *An $\mathcal{EL}^\bot$-concept description C over (N_C, N_R) is called* relative model-based most-specific concept description of $X \subseteq \Delta^\mathcal{I}$ w.r.t. $\mathcal{I}$ and $\mathcal{T}$ *if the following conditions are satisfied:*

1. *$X \subseteq C^\mathcal{I}$.*
2. *If $X \subseteq D^\mathcal{I}$ holds for a concept description $D \in \mathcal{EL}^\bot(N_C, N_R)$ then $\mathcal{T} \models C \sqsubseteq D$.*

The definition implies that all relative model-based most-specific concept descriptions of a subset $X \subseteq \Delta^\mathcal{I}$ are equivalent w.r.t. the TBox $\mathcal{T}$. Hence, we use the symbol $X^{\mathcal{I}_\mathcal{T}}$ for *the* relative mmsc of X w.r.t. $\mathcal{I}$ and $\mathcal{T}$.

However, as an immediate consequence from the definition it follows that the model-based most-specific concept description of $X \subseteq \Delta^\mathcal{I}$ w.r.t. $\mathcal{I}$ is always a relative model-based most-specific concept description of X w.r.t. $\mathcal{I}$ and $\mathcal{T}$.

There are situations where the relative mmsc exists but not the standard mmsc. Consider the interpretation $\mathcal{I}$ described by the following graph:



Then d has no model-based most specific concept in $\mathcal{EL}^\bot$ but has a relative mmsc w.r.t. $\mathcal{T} := \{A \sqsubseteq \exists r.\, A\}$, in particular it holds $d^{\mathcal{I}_\mathcal{T}} = A$. Of course, d has a role-depth bounded mmsc, and a mmsc in $\mathcal{EL}^\bot_{\mathsf{gfp}}$ with greatest fixpoint semantics, respectively.

For the following statements on the construction of relative bases of GCIs we strongly need the fact that all model-based most-specific concept descriptions exist. If computability is neccessary, too, then we further have to enforce that there are only finitely many model-based most-specific concept descriptions (up to equivalence) and that the interpretation only contains finitely many individuals; of course the second requirement implies the first.

The model-based most-specific concept description of every individual x w.r.t. $\mathcal{I}$ clearly exists if the interpretation $\mathcal{I}$ is finite and acyclic. Relative model-based most-specific concept descriptions exist if we can find suitable synchronised simulations on a description graph constructed from the interpretation and the TBox. A detailed characterisation and appropriate proofs will be subject of a future paper.

In case we cannot ensure the existence of mmscs for all individuals of the interpretation we may also adopt role-depth bounds. Further details are given in [10]. Then we modify the definition of a relative base of GCIs to only involve GCIs whose subsumee and subsumer satisfy the role-depth bound. This is both applied to the GCIs in the base and the GCIs that must be entailed. As a consequence we are able to treat cyclic interpretations whose cycles are not already modeled in the background TBox.

As in the default case without a TBox, the definition of relative model-based most-specific concept descriptions yields a quasi-adjunction between the powerset lattice $(2^{\Delta^{\mathcal{I}}}, \subseteq)$ and the quasiordered set $(\mathcal{EL}^{\bot}(N_C, N_R), \sqsubseteq_{\mathcal{T}})$.

Lemma 1 (Properties of Relative mmscs). *For all subsets $X, Y \subseteq \Delta^{\mathcal{I}}$ and all concept descriptions $C, D \in \mathcal{EL}^{\bot}(N_C, N_R)$ the following statements hold:*

1. $X \subseteq C^{\mathcal{I}}$ *if and only if* $\mathcal{T} \models X^{\mathcal{I}_{\mathcal{T}}} \sqsubseteq C$
2. $X \subseteq Y$ *implies* $\mathcal{T} \models X^{\mathcal{I}_{\mathcal{T}}} \sqsubseteq Y^{\mathcal{I}_{\mathcal{T}}}$
3. $\mathcal{T} \models C \sqsubseteq D$ *implies* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
4. $X \subseteq X^{\mathcal{I}_{\mathcal{T}}\mathcal{I}}$
5. $\mathcal{T} \models C^{\mathcal{I}\mathcal{I}_{\mathcal{T}}} \sqsubseteq C$
6. $\mathcal{T} \models X^{\mathcal{I}_{\mathcal{T}}} \equiv X^{\mathcal{I}_{\mathcal{T}}\mathcal{I}\mathcal{I}_{\mathcal{T}}}$
7. $C^{\mathcal{I}} = C^{\mathcal{I}\mathcal{I}_{\mathcal{T}}\mathcal{I}}$

In order to obtain a first relative base of GCIs for $\mathcal{I}$ w.r.t. $\mathcal{T}$ we can prove that it suffices to have mmscs as the right-hand-sides of concept inclusions in a relative base. More specifically, it holds that the set

$$\left\{ C \sqsubseteq C^{\mathcal{I}\mathcal{I}_{\mathcal{T}}} \;\middle|\; C \in \mathcal{EL}^{\bot}(N_C, N_R) \right\}$$

is a relative of GCIs for $\mathcal{I}$ w.r.t. $\mathcal{T}$. This statement is a simple consequence of the fact that a GCI $C \sqsubseteq D$ only holds in $\mathcal{I}$ if it follows from $\mathcal{T} \cup \left\{ C \sqsubseteq C^{\mathcal{I}\mathcal{I}_{\mathcal{T}}} \right\}$.

In the following text we want to make a strong connection to formal concept analysis in a similar way as Baader and Distel did in [3, 4, 12]. We therefore define a set $\mathcal{M}_{\mathcal{I},\mathcal{T}}$ of $\mathcal{EL}^{\bot}$-concept descriptions such that all relative model-based most-specific concept descriptions can be expressed as a conjunction over a subset of $\mathcal{M}_{\mathcal{I},\mathcal{T}}$. We use similar techniques like lower approximations and induced contexts but in an extended way to be explicitly able to handle background knowledge in a TBox.

For an $\mathcal{EL}$-concept description in its normal form $C \equiv \bigsqcap_{A \in U} A \sqcap \bigsqcap_{(r,D) \in \Pi} \exists r. D$ we define its *lower approximation* w.r.t. $\mathcal{I}$ and $\mathcal{T}$ as the $\mathcal{EL}$-concept description

$$\lfloor C \rfloor_{\mathcal{I},\mathcal{T}} := \bigsqcap_{A \in U} A \sqcap \bigsqcap_{(r,D) \in \Pi} \exists r. D^{\mathcal{I}\mathcal{I}_{\mathcal{T}}}.$$

As a consequence of the definition we get that $\mathcal{T}$ entails the concept inclusion $\lfloor C \rfloor_{\mathcal{I},\mathcal{T}} \sqsubseteq C$. The explicit proof uses Lemma 1 5 and the fact that both conjunction and existential restrictions are monotonic. This also justifies the name of a lower approximation of C.

Furthermore, it is readily verified that all lower approximations can be expressed in terms of the set $\mathcal{M}_{\mathcal{I},\mathcal{T}}$ which is defined as follows:

$$\mathcal{M}_{\mathcal{I},\mathcal{T}} := \{\bot\} \cup N_C \cup \left\{ \exists r. X^{\mathcal{I}_{\mathcal{T}}} \;\middle|\; r \in N_r, \emptyset \neq X \subseteq \Delta^{\mathcal{I}} \right\}$$

In order to prove that also each model-based most-specific concept description is expressible in terms of $\mathcal{M}_{\mathcal{I},\mathcal{T}}$ it suffices to show that every model-based most-specific concept description is equivalent to its lower approximation. We already know from Lemma 1 5 that $\mathcal{T}$ entails the concept inclusion $C^{\mathcal{I}\mathcal{I}_\mathcal{T}} \sqsubseteq C$. Furthermore, for all concept descriptions $C, D \in \mathcal{EL}^\bot(N_C, N_R)$ and all role names $r \in N_R$ it may be easily shown by means of Lemma 1 7 that the following two statements hold:

1. $(C \sqcap D)^\mathcal{I} = (C \sqcap D^{\mathcal{I}\mathcal{I}_\mathcal{T}})^\mathcal{I}$.
2. $(\exists r.C)^\mathcal{I} = (\exists r.C^{\mathcal{I}\mathcal{I}_\mathcal{T}})^\mathcal{I}$.

As a consequence it follows that both the mmsc $C^{\mathcal{I}\mathcal{I}_\mathcal{T}}$ and the lower approximation $\lfloor C \rfloor_{\mathcal{I},\mathcal{T}}$ have the same extensions w.r.t. $\mathcal{I}$, and then Lemma 1 1 yields that $\mathcal{T}$ entails $C^{\mathcal{I}\mathcal{I}_\mathcal{T}} \sqsubseteq \lfloor C \rfloor_{\mathcal{I},\mathcal{T}}$. In summary, it follows that $\mathcal{T}$ entails the concept inclusions

$$C^{\mathcal{I}\mathcal{I}_\mathcal{T}} \equiv C^{\mathcal{I}\mathcal{I}_\mathcal{T}\mathcal{I}\mathcal{I}_\mathcal{T}} \sqsubseteq \lfloor C^{\mathcal{I}\mathcal{I}_\mathcal{T}} \rfloor_{\mathcal{I},\mathcal{T}} \sqsubseteq C^{\mathcal{I}\mathcal{I}_\mathcal{T}},$$

and hence each relative model-based most-specific concept description is equivalent to its lower approximation w.r.t. $\mathcal{T}$, and thus is expressible in terms of $\mathcal{M}_{\mathcal{I},\mathcal{T}}$.

Now we are ready to use methods of formal concept analysis to construct a minimal base of GCIs for $\mathcal{I}$ w.r.t. $\mathcal{T}$. We therefore first introduce the *induced context* w.r.t. $\mathcal{I}$ and $\mathcal{T}$ which is defined as $\mathbb{K}_{\mathcal{I},\mathcal{T}} := (\Delta^\mathcal{I}, \mathcal{M}_{\mathcal{I},\mathcal{T}}, I)$ where $(d, C) \in I$ if and only if $d \in C^\mathcal{I}$ holds. Additionally, the *background knowledge* is defined as the implication set

$$\mathcal{S}_{\mathcal{I},\mathcal{T}} := \{\{C\} \to \{D\} \mid C, D \in \mathcal{M}_{\mathcal{I},\mathcal{T}} \text{ and } \mathcal{T} \models C \sqsubseteq D\}.$$

One the one hand we need this background implications to skip the computation of trivial GCIs $C \sqsubseteq D$ where the implication $\{C\} \to \{D\}$ is not neccessarily trivial in $\mathbb{K}_{\mathcal{I},\mathcal{T}}$, and on the other hand it is needed to avoid the generation of GCIs that are already entailed by $\mathcal{T}$.

As an immediate consequence of the definition it follows that $(\bigsqcap U)^\mathcal{I} = U^I$ holds for all subsets $U \subseteq \mathcal{M}_{\mathcal{I},\mathcal{T}}$, and hence we infer that for all subsets $U, V \subseteq \mathcal{M}_{\mathcal{I},\mathcal{T}}$ the GCI $\bigsqcap U \sqsubseteq \bigsqcap V$ holds in $\mathcal{I}$ if and only if the implication $U \to V$ holds in the induced context $\mathbb{K}_{\mathcal{I},\mathcal{T}}$. Furthermore, it is true that conjunctions of intents of $\mathbb{K}_{\mathcal{I},\mathcal{T}}$ are exactly the mmscs w.r.t. $\mathcal{I}$ and $\mathcal{T}$, i.e., $\mathcal{T} \models \bigsqcap U^{II} \equiv (\bigsqcap U)^{\mathcal{I}\mathcal{I}_\mathcal{T}}$ holds for all subsets $U \subseteq \mathcal{M}_{\mathcal{I},\mathcal{T}}$. Eventually, the previous statements allow for the transformation of a minimal implicational base of the induced context $\mathbb{K}_{\mathcal{I},\mathcal{T}}$ w.r.t. the background knowledge $\mathcal{S}_{\mathcal{I},\mathcal{T}}$ into a minimal base of GCIs for $\mathcal{I}$ relative to the background TBox $\mathcal{T}$.

Theorem 1 (Minimal Relative Base of GCIs). *Assume that all model-based most-specific concept descriptions of $\mathcal{I}$ relative to $\mathcal{T}$ exist. Let $\mathcal{L}$ be a minimal implicational base of the induced context $\mathbb{K}_{\mathcal{I},\mathcal{T}}$ w.r.t. the background knowledge $\mathcal{S}_{\mathcal{I},\mathcal{T}}$. Then $\{\bigsqcap U \sqsubseteq \bigsqcap U^{II} \mid U \to U^{II} \in \mathcal{L}\}$ is a minimal base of GCIs for $\mathcal{I}$ relative to $\mathcal{T}$.*

Eventually, the following set is the (minimal) *canonical base for $\mathcal{I}$ relative to $\mathcal{T}$*:

$$\mathcal{B}_{\mathcal{I},\mathcal{T}} := \left\{ \bigsqcap P \sqsubseteq \bigsqcap P^{II} \;\middle|\; P \text{ is a pseudo-intent of } \mathbb{K}_{\mathcal{I},\mathcal{T}} \text{ relative to } \mathcal{S}_{\mathcal{I},\mathcal{T}} \right\}.$$

All of the results presented in this section are generalisations of those from Baader and Distel in [3, 4, 12], and for the special case of an empty background TBox $\mathcal{T} = \emptyset$ the definitions and propositions coincide. In particular, the last Theorem 1 constructs the same base of GCIs as [12, Theorem 5.12, Corollary 5.13] for $\mathcal{T} = \emptyset$.

5 Experts in the Domain of Interest

We have seen how to extend an existing TBox $\mathcal{T}$ with concept inclusions holding in an interpretation $\mathcal{I}$ that is a model of $\mathcal{T}$. However, there might be situations where we want to adjust a TBox $\mathcal{T}$ with information from an interpretation $\mathcal{I}$ that is not a model of $\mathcal{T}$. In order to use the results from the previous section on relative bases it is neccessary to adjust the interpretation or the TBox such that as much knowledge as possible is preserved and the adjusted interpretation models the adjusted TBox. However, an automatic approach can hardly decide whether counterexamples in the interpretation are valid in the domain of interest, or whether concept inclusions hold in the domain of interest. We therefore need some external information to decide whether a concept inclusion should be considered true or false in the domain of interest.

Beforehand, we define the notion of adjustments as follows.

Definition 3 (Adjustment). *Let $\mathcal{I}$ be an interpretation that does not model the GCI $C \sqsubseteq D$.*

1. *An interpretation $\mathcal{J}$ is called an* adjustment *of $\mathcal{I}$ w.r.t. $C \sqsubseteq D$ if it satisfies the following conditions:*
   (a) $\mathcal{J} \models C \sqsubseteq D$.
   (b) $\Delta^{\mathcal{I}} \setminus X \subseteq \Delta^{\mathcal{J}}$.
   (c) $A^{\mathcal{I}} \setminus X \subseteq A^{\mathcal{J}}$ *holds for all concept names* $A \in N_C$.
   (d) $r^{\mathcal{I}} \setminus (\Delta^{\mathcal{I}} \times X \cup X \times \Delta^{\mathcal{I}}) \subseteq r^{\mathcal{J}}$ *holds for all role names* $r \in N_R$.
   The set $X := C^{\mathcal{I}} \setminus D^{\mathcal{I}}$ denotes the set of all counterexamples in $\mathcal{I}$ against $C \sqsubseteq D$.
   We call an adjustment $\mathcal{J}$ minimal *if the value $\sum_{A \in N_C} |A^{\mathcal{I}} \triangle A^{\mathcal{J}}| + \sum_{r \in N_R} |r^{\mathcal{I}} \triangle r^{\mathcal{J}}|$ is minimal among all adjustments of $\mathcal{I}$ w.r.t. $C \sqsubseteq D$.*
2. *A general concept inclusion $E \sqsubseteq F$ is called an* adjustment *of $C \sqsubseteq D$ w.r.t. $\mathcal{I}$ if it satisfies the following conditions:*
   (a) $\mathcal{I} \models E \sqsubseteq F$.
   (b) $E \sqsubseteq C$.
   (c) $D \sqsubseteq F$.
   An adjustment $E \sqsubseteq F$ is called minimal *if there is no adjustment $X \sqsubseteq Y$ such that $E \sqsubseteq X$ and $Y \sqsubseteq F$ holds.*

As next step we introduce the definition of an expert that is used to guide the incremental exploration process, i.e., it ensures that the new interpretation is always adjusted such that it models the adjusted TBox.

Definition 4 (Expert). *An* expert *is a mapping from pairs of interpretations $\mathcal{I}$ and GCIs $C \sqsubseteq D$ where $\mathcal{I} \not\models C \sqsubseteq D$ to adjustments. We say that the expert* accepts *$C \sqsubseteq D$ if it adjusts the interpretation, and that it* declines *$C \sqsubseteq D$ if it adjusts the GCI.*

Furthermore, the following requirements must be satisfied:

1. *Acceptance must be independent of $\mathcal{I}$, i.e., if χ accepts $C \sqsubseteq D$ w.r.t. $\mathcal{I}$ then χ must also accept $C \sqsubseteq D$ w.r.t. any other interpretation $\mathcal{J}$.*
2. *Adjusted interpretations must model previously accepted GCIs and must not model previously declined GCIs.*
3. *Adjustments of declined GCIs must be accepted.*

An expert is called minimal *if it always returns minimal adjustments.*

5.1 Examples for Experts

Of course, we may use a *human expert* who is aware of the full knowledge holding in the domain of interest. However, the problem of the construction of *automatically acting experts* is left for future research. We will only present some first ideas.

An expert may be defined by means of the confidence measure that has been introduced by Borchmann in [7, 8]. For a GCI $C \sqsubseteq D$ and an interpretation $\mathcal{I}$ it is defined by

$$\mathsf{conf}_{\mathcal{I}}(C \sqsubseteq D) := \frac{\left|(C \sqcap D)^{\mathcal{I}}\right|}{\left|C^{\mathcal{I}}\right|} \in [0,1].$$

Note that $\mathsf{conf}_{\mathcal{I}}(C \sqsubseteq D) = 1$ iff $\mathcal{I} \models C \sqsubseteq D$. This confidence can give a hint whether an expert should accept or decline the GCI. Assume that $c \in [0,1)$ is a *confidence threshold*. In case $1 > \mathsf{conf}_{\mathcal{I}}(C \sqsubseteq D) \geq c$, i.e., if there are some but not too many counterexamples against $C \sqsubseteq D$ in $\mathcal{I}$, the expert accepts the GCI and has to adjust $\mathcal{I}$, and otherwise declines the GCI and returns an adjustment of it.

Another approach is as follows. Let $\mathcal{I} = \mathcal{I}_t \uplus \mathcal{I}_u$ be a disjoint union of the *trusted subinterpretation* $\mathcal{I}_t$ (which is assumed to be error-free) and the *untrusted subinterpretation* $\mathcal{I}_u$. Then the expert accepts $C \sqsubseteq D$ if it holds in $\mathcal{I}_t$, and declines otherwise.

Of course, the automatic construction of adjustments is not addressed with both approaches as they only provide methods for the decision whether the expert should accept or decline. The next section presents possibilities for adjustment construction.

5.2 Construction of Adjustments

Adjusting the general concept inclusion Consider a general concept inclusion $C \sqsubseteq D$ that does not hold in the interpretation $\mathcal{I}$. The expert now wants to decline the GCI by adjusting it. According to the definition of adjustments of GCIs it is both possible to shrink the premise C and to enlarge the conclusion D to construct a GCI that holds in $\mathcal{I}$ but is more general than $C \sqsubseteq D$. Of course, it is always simply possible to return the adjustment $\bot \sqsubseteq \top$ but this may not be a good practise since then no knowledge that is enclosed in $C \sqsubseteq D$ and holds in $\mathcal{I}$ would be preserved.

In order to adjust the GCI more carefully the expert has the following options:

1. Add a conjunct to C, or choose an existential restriction $\exists r.E$ in C and modify E such that the resulting concept description is more specific than E, e.g., by adding a conjunct, or by adjusting an existential restriction.
2. Choose a conjunct in D and remove it, or choose an existential restriction $\exists r.E$ in D and modify E such that the resulting concept description is more general than E, e.g., by removing a conjunct, or adjusting an existential restriction.

In order to obtain a minimal adjustment the expert should only apply as few changes as possible.

Adjusting the interpretation To generate an adjustment of $\mathcal{I}$ w.r.t. $C \sqsubseteq D$ the expert may either remove or modify the counterexamples in $\mathcal{I}$, or introduce new individuals, to enforce the GCI. The simplest solution is to just remove all counterexamples against $C \sqsubseteq D$ from $\mathcal{I}$, and this may always be done by automatic experts. Of course, the

removal of a counterexample from the interpretation is an impractical solution since in most cases there will only be few errors in the dataset. A more intelligent solution involves the modification of the concept and role name extensions occuring in the premise or conclusion of the GCI at hand.

Let $x \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$ be a counterexample. Then the expert may proceed as follows:

1. Remove x from the interpretation.
2. For a *modification of the premise* it suffices to choose one conjunct E of C and modify its extension such that $x \notin E^{\mathcal{I}}$ holds. The expert may choose either a concept name A in C and remove the element x from the extension $A^{\mathcal{I}}$, or an existential restriction $\exists\, r.\, E$ in C and modify the interpretation such that x is not in the extension $(\exists\, r.\, E)^{\mathcal{I}}$ anymore. This may either be done by removing all r-successors of x that are elements of $E^{\mathcal{I}}$, or by modifying all r-successors such that they are not elements of the extension $E^{\mathcal{I}}$ anymore.
3. For a *modification of the conclusion* the expert has to modify all conjuncts E of D with $x \notin E^{\mathcal{I}}$. If $E = A$ is a concept name in D then the expert simply has to add x to the extension of A. If $E = \exists\, r.\, F$ is an existential restriction in D then the expert has the following choices:
   (a) Choose an existing r-successor y of x and modify y such that $y \in E^{\mathcal{I}}$ holds. In case of E containing an existential restriction as a subconcept a modification or introduction of further successors may be neccessary.
   (b) Introduce a new r-successor y of x such that $y \in E^{\mathcal{I}}$ holds. If E contains an existential restriction as a subconcept then this action requires the introduction of further new elements in the interpretation.

6 An Incremental Learning Algorithm

By means of experts it is possible to adjust an interpretation $\mathcal{I}$ and a TBox $\mathcal{T}$ such that $\mathcal{I} \models \mathcal{T}$. This enables us to use the techniques for the computation of relative bases as described in Section 4. Based on these results and definitions, we now want to formulate an *incremental learning algorithm* which takes a possibly empty initial TBox $\mathcal{T}_0$ and a sequence $(\mathcal{I}_n)_{n\geq 1}$ of interpretations as input, and iteratively adjusts the TBox in order to compute a TBox of GCIs holding in the domain of interest. This is modeled as a sequence $(\mathcal{T}_n)_{n\geq 0}$ of TBoxes where each TBox $\mathcal{T}_n$ is defined as the base of the adjustment of $\mathcal{I}_n$ relative to the adjustment of $\mathcal{T}_{n-1}$. Of course, we also have to presuppose an expert χ that has full knowledge on the domain of interest and provides the neccessary adjustments during the algorithm's run. The algorithm is briefly described as follows and also given in pseudo-code in Algorithm 1.

(Start) Assume that the TBox $\mathcal{T}_{n-1}$ has been constructed and a new interpretation $\mathcal{I}_n$ is available. In case $\mathcal{I}_n \models \mathcal{T}_{n-1}$ we may skip the next step, and otherwise we first have to adjust both the TBox and interpretation in the next step.

(Adjustment) For each GCI $C \sqsubseteq D \in \mathcal{T}_{n-1}$ ask the expert χ whether it accepts it. If yes then set $\mathcal{I}_n$ to the returned adjustment $\chi(C \sqsubseteq D, \mathcal{I}_n)$. If it otherwise declines it then replace the GCI $C \sqsubseteq D$ with the returned adjustment $\chi(C \sqsubseteq D, \mathcal{I}_n)$ in $\mathcal{T}_n$. After all GCIs have been processed then we have that $\mathcal{I}_n \models \mathcal{T}_n$ holds.

(Computation of the Relative Base) As a next step we compute the base $\mathcal{B}_n$ of $\mathcal{I}_n$ relative to $\mathcal{T}_{n-1}$ and set $\mathcal{T}_n := \mathcal{T}_{n-1} \cup \mathcal{B}_n$. Set $n := n + 1$ and goto (Start).

It may occur that a previously answered question is posed to the expert again during the algorithm's run. Of course, we may apply caching techniques, i.e., store a set of accepted GCIs and a set of declined GCIs but this will raise the problem how an adjustment of the interpretation (for acceptance), or of the GCI (for decline), respectively, can be constructed, when it is not returned from the expert itself. Some simple solutions are given in the previous section, e.g., one may just remove all counterexamples for an accepted GCI from the interpretation, or replace the GCI with the adjusted one that has been previously returned by the expert. For this purpose the algorithm may build a set $\mathcal{T}_\chi$ of accepted GCIs to avoid a second question for the same concept inclusion, and a set $\mathcal{F}_\chi$ of pairs of declined GCIs and their adjustments.

Algorithm 1 Incremental Learning Algorithm

```
Require: a domain expert χ, a TBox T (initial knowledge)
Let T_χ := ∅ and F_χ := ∅.
while a new interpretation I has been observed do
    while I ⊭ T do
        for all C ⊑ D ∈ T do
            if I ⊭ C ⊑ D then
                if C ⊑ D ∈ T_χ then
                    Remove all counterexamples against C ⊑ D from I.
                else if (C ⊑ D, E ⊑ F) ∈ F_χ then
                    Remove C ⊑ D from T.
                else if χ accepts C ⊑ D then
                    Set I := χ(C ⊑ D, I).
                    Set T_χ := T_χ ∪ {C ⊑ D}.
                else
                    Let E ⊑ F := χ(C ⊑ D, I) be the adjustment of the GCI.
                    Set T := (T \ {C ⊑ D}) ∪ {E ⊑ F}.
                    Set F_χ := F_χ ∪ {(C ⊑ D, E ⊑ F)}.
                    Set T_χ := T_χ ∪ {E ⊑ F}.
    Let T := T ∪ B where B is a base of I relative to T.
return T.
```

With slight restrictions on the expert and the interpretations used as input data during the algorithm's run we may prove soundness (w.r.t. the domain of interest) and completeness (w.r.t. the processed input interpretations) of the final TBox that is returned after no new interpretation has been observed.

Proposition 1 (Soundness and Completeness). *Assume that $\mathcal{I}$ is the domain of interest, and $\mathcal{T}_0$ is the initial TBox where $\mathcal{I} \models \mathcal{T}_0$. Furthermore, let χ be an expert that has full knowledge of $\mathcal{I}$, i.e., does not decline any GCI holding in $\mathcal{I}$; and let $\mathcal{I}_1, \ldots, \mathcal{I}_n$ be a sequence of sound interpretations, i.e., each $\mathcal{I}_k$ only models GCIs holding in $\mathcal{I}$. Then the final TBox $\mathcal{T}_n$ is sound for $\mathcal{I}$, i.e., only contains GCIs holding in $\mathcal{I}$, and is complete for the adjustment of each $\mathcal{I}_k$, i.e., all GCIs holding in the adjustment of any $\mathcal{I}_k$ are entailed by $\mathcal{T}_n$.*

Proof. We prove the claim by induction on k. Since we have that $\mathcal{T}_k$ holds in $\mathcal{I}$ the expert does not adjust $\mathcal{T}_k$ but constructs an adjustment $\mathcal{I}'_{k+1}$ of $\mathcal{I}_{k+1}$ that is a model of $\mathcal{T}_k$. Then the next TBox is obtained as $\mathcal{T}_{k+1} := \mathcal{T}_k \cup \mathcal{B}_{k+1}$ where $\mathcal{B}_{k+1}$ is a base of

$\mathcal{I}'_{k+1}$ relative to $\mathcal{T}_k$. By assumption, $\mathcal{I}$ is a model of $\mathcal{B}_{k+1}$, i.e., also of $\mathcal{T}_{k+1}$, and by construction $\mathcal{T}_{k+1}$ is complete for $\mathcal{I}'_{k+1}$. Finally, $\mathcal{T}_\ell \subseteq \mathcal{T}_k$ holds for all $\ell \leq k$, and thus we conclude that $\mathcal{T}_{k+1}$ must be complete for all adjusted interpretations $\mathcal{I}'_1, \ldots, \mathcal{I}'_{k+1}$. □

7 Comparison with the existing single-step-learning approaches

In comparison with a single-step approach that explores the canonical base of the disjoint union $\biguplus \mathcal{I}_k$ there are several drawbacks. The first problem is to store all the observed interpretations. Secondly, upon input of a new interpretation there is no output of the refuted old GCIs, and newly obtained GCIs, respectively. Thirdly, a major disadvantage is the computational complexity. In order to iteratively compute the relative bases the model-based most-specific concept descriptions of each interpretation $\mathcal{I}_k$ are neccessary, and their number can be exponential in the size of the domain of $\mathcal{I}_k$. Hence for n input interpretations up to $m := (2^{|\Delta^{\mathcal{I}_1}|} - 1) + \ldots + (2^{|\Delta^{\mathcal{I}_n}|} - 1)$ mmscs have to be constructed. In order to compute the canonical base of the disjoint sum $\biguplus \mathcal{I}_k$ we have to compute up to $m' := 2^{|\Delta^{\mathcal{I}_1}| + \ldots + |\Delta^{\mathcal{I}_n}|} - 1$ mmscs. Obviously, m is much smaller than m' for a sufficently large number n of input interpretations.

Furthermore, the upper bound for the size of the induced context $\mathbb{K}_{\mathcal{I}_k}$ is $|\Delta^{\mathcal{I}_k}| \cdot (1 + |N_C| + |N_R| \cdot (2^{|\Delta^{\mathcal{I}_k}|} - 1))$, and the number of implications may be exponential in the size of the context, i.e., double-exponential in the size of the interpretation $\mathcal{I}_k$, hence in the iterative approach we get an upper bound of

$$2^{|\Delta^{\mathcal{I}_1}| \cdot (1 + |N_C| + |N_R| \cdot (2^{|\Delta^{\mathcal{I}_1}|} - 1))} + \ldots + 2^{|\Delta^{\mathcal{I}_n}| \cdot (1 + |N_C| + |N_R| \cdot (2^{|\Delta^{\mathcal{I}_n}|} - 1))}$$

GCIs, and in the single-step approach the upper bound is given as

$$2^{(|\Delta^{\mathcal{I}_1}| + \ldots + |\Delta^{\mathcal{I}_n}|) \cdot (1 + |N_C| + |N_R| \cdot (2^{|\Delta^{\mathcal{I}_1}| + \ldots + |\Delta^{\mathcal{I}_n}|} - 1))}.$$

It is easy to see that $\sum_k 2^{2^{|\Delta^{\mathcal{I}_k}|}}$ is much smaller than $2^{2^{\sum_k |\Delta^{\mathcal{I}_k}|}}$.

8 Conclusion

We have presented a method for the construction of a minimal extension of a TBox w.r.t. a model, and utilised it to formulate an algorithm that learns $\mathcal{EL}^\perp$-concept inclusions from interpretation streams with external support by means of an expert in the domain of interest. The approach can be applied to a wide range of input data as there are various cryptomorphic structures for interpretations, like description graphs, binary power context families, RDF-graphs (with some effort for a transformation) etc. It may be extended towards more expressive description logics, e.g., $\mathcal{FLE}$, or $\mathcal{ALE}$, or to include the construction of RBoxes in DLs allowing for role hierarchies or complex role inclusions. Another direction for future research is the construction of bases for temporal terminological knowledge.

This document has not considered the problem of the computation of relative model-based most-specific concept descriptions. However, it is possible to use synchronised simulations for their construction, and a detailed characterisation of the existence and construction will be subject of a future paper.

Bibliography

[1] Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. pp. 319–324 (2003)

[2] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)

[3] Baader, F., Distel, F.: A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In: Formal Concept Analysis, 6th International Conference, ICFCA 2008, Montreal, Canada, February 25-28, 2008, Proceedings. pp. 46–61 (2008)

[4] Baader, F., Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\text{gfp}}$. In: Formal Concept Analysis, 7th International Conference, ICFCA 2009, Darmstadt, Germany, May 21-24, 2009, Proceedings. pp. 146–161 (2009)

[5] Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. LTCS-Report 06-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Dresden, Germany (2006)

[6] Baader, F., Sertkaya, B.: Applying formal concept analysis to description logics. In: Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings. pp. 261–286 (2004)

[7] Borchmann, D.: Towards an error-tolerant construction of $\mathcal{EL}^{\perp}$-ontologies from data using formal concept analysis. In: Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings. pp. 60–75 (2013)

[8] Borchmann, D.: Learning Terminological Knowledge with High Confidence from Erroneous Data. Ph.D. thesis, Dresden University of Technology, Dresden, Germany (2014)

[9] Borchmann, D., Distel, F.: Mining of $\mathcal{EL}$-gcis. In: Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011. pp. 1083–1090 (2011)

[10] Borchmann, D., Distel, F., Kriegel, F.: Axiomatization of General Concept Inclusions from Finite Interpretations. LTCS-Report 15-13, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2015)

[11] Cohen, W.W., Hirsh, H.: Learning the classic description logic: Theoretical and experimental results. In: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994. pp. 121–133 (1994)

[12] Distel, F.: Learning Description Logic Knowledge Bases from Data using Methods from Formal Concept Analysis. Ph.D. thesis, Dresden University of Technology, Dresden, Germany (2011)

[13] Ganter, B.: Two basic algorithms in concept analysis. Preprint 831, Darmstadt University of Technology (1984)

[14] Ganter, B.: Attribute exploration with background knowledge. Theor. Comput. Sci. 217(2), 215–233 (1999)
[15] Ganter, B.: Begriffe und Implikationen. Preprint, Chair for Algebraic Structure Theory, Institute for Algebra, Dresden University of Technology, Dresden, Germany (2000)
[16] Ganter, B.: Two basic algorithms in concept analysis. In: Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings. pp. 312–340 (2010)
[17] Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) Conceptual Structures: Broadening the Base, Lecture Notes in Computer Science, vol. 2120, pp. 129–142. Springer Berlin Heidelberg (2001)
[18] Ganter, B., Wille, R.: Contextual attribute logic. In: Conceptual Structures: Standards and Practices, 7th International Conference on Conceptual Structures, ICCS '99, Blacksburg, Virginia, USA, July 12-15, 1999, Proceedings. pp. 377–388 (1999)
[19] Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer (1999)
[20] Guigues, J.-L., D.V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 95, 5–18 (1986)
[21] Kriegel, F.: Concept Explorer FX (2010-2015), `https://github.com/francesco-kriegel/conexp-fx`, Software for Formal Concept Analysis
[22] Kriegel, F.: Visualization of Conceptual Data with Methods of Formal Concept Analysis. Diploma thesis, Dresden University of Technology, Dresden, Germany (2013)
[23] Kriegel, F.: Incremental computation of concept diagrams. In: Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Supplemental Proceedings. pp. 45–61 (2014)
[24] Kriegel, F.: Next closures – parallel exploration of constrained closure operators. LTCS-Report 15-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Dresden, Germany (2015)
[25] Küsters, R., Molitor, R.: Structural subsumption and least common subsumers in a description logic with existential and number restrictions. Studia Logica 81(2), 227–259 (2005)
[26] Rudolph, S.: Exploring relational structures via $\mathcal{FLE}$. In: Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS 2004, Huntsville, AL, USA, July 19-23, 2004. Proceedings. pp. 196–212 (2004)
[27] Rudolph, S.: Relational Exploration - Combining Description Logics and Formal Concept Analysis for Knowledge Specification. Ph.D. thesis, Dresden University of Technology, Dresden, Germany (2006)
[28] Sertkaya, B.: Formal Concept Analysis Methods for Description Logics. Ph.D. thesis, Dresden University of Technology, Dresden, Germany (2007)
[29] Stumme, G.: Attribute exploration with background implications and exceptions. In: Bock, H.H., Polasek, W. (eds.) Data Analysis and Information Systems, pp. 457–469. Studies in Classification, Data Analysis, and Knowledge Organization, Springer Berlin Heidelberg (1996)
[30] Valiant, L.G.: A theory of the learnable. Commun. ACM 27(11), 1134–1142 (1984)
[31] Zickwolff, M.: Rule Exploration: First Order Logic in Formal Concept Analysis. Ph.D. thesis, Technische Hochschule Darmstadt, Germany (1991)

A higher-order semantics for OWL 2 QL ontologies

Maurizio Lenzerini, Lorenzo Lepore, Antonella Poggi

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti"- Sapienza Università di Roma
Via Ariosto 25, I-00183 Roma, Italy
lastname@dis.uniroma1.it

Recent OBDA projects have pointed out that one of the drawbacks of OWL 2 is the lack of metamodeling and metaquerying capabilities, i.e., features for specifying and reasoning about metaconcepts and metaproperties [1]. Roughly speaking, a metaconcept is a concept whose instances can be themselves concepts, and a metaproperty is a relationship between metaconcepts. Although OWL 2 provides syntactic support for metamodeling through punning (by which the same name can be used to denote ontology elements of different categories, such as a class and an individual), we argue that the official semantics of OWL 2, the so-called *Direct Semantics* (DS), treats punning in a way that is not adequate for representing metaconcepts and metaproperties. The reason is simply that proper metamodeling requires that the *same* element plays the role of individual and class (or, class and relation), while DS sanctions that an individual and a class with the same name are different elements. This is confirmed by the fact that the *Direct Semantics Entailment Regime* (DSER), which is the logic-based semantics of the SPARQL 1.1 query language when applied to OWL 2 QL, forces queries to obey the so-called *typing constraint*, which rules out the possibility of using the same variable in incompatible positions (for example, in individual and in class position).

The issue of metamodeling in OWL has been investigated in several papers. It is known that the semantics of metamodeling of OWL 2 Full leads to undecidability of basic inference problems [7]. A possible solution to this problem is to enable metamodeling in OWL 2 DL by axiomatizing the higher order knowledge into first order assertions [4], but the process involves the use of complex expressions that are not supported by OWL 2 DL tractable profiles, and therefore seems inapplicable in OBDA. Another possible solution is the stratification of class constructors and axioms to describe metalevels of classes and properties [8], but such stratification poses challenges for the modeler, and rules out interesting ontology patterns. Relevant to our work are recent papers aiming at devising efficient techniques to answer SPARQL 1.1 queries posed to OWL 2 QL ontologies [2, 6, 5]. However, such papers concentrate on DSER, and therefore do not aim at the full power of metamodeling and metaquerying.

The goal of our work is to present a new higher-order semantics for OWL 2 QL, called *HOS* and inspired by [3], allowing us to effectively exploit the capabilities for metamodeling offered by punning, and to show that, based on such semantics, it is possible to define a new SPARQL entailment regime, called *HOSER*, for the class of (meta)queries obtained by relaxing the typing constraint of DSER.

Illustrating scenario. We refer to an OWL 2 QL ontology (see an excerpt in Table 1), whose central entity is :financial_instrument.Metamodeling comes into play in order to capture the notion of types of financial instruments, modeled by using the metaclass :type_of_f_i, whose instances are other classes which are intended to be subclasses of :financial_instrument (for example, :BTP and :commercial_paper, see axioms (3),(4),(6)), and by defining its properties. One notable example of such properties is :established_by (see axioms (9),(10),(11)), which associates to each type of financial instrument the law that formally established it (for example, the law :DR135bis established :BTP as a type of financial instrument, see axiom (5)).Observe that syntactically we are using punning. However, as we said before, the direct semantics of OWL 2 considers the individual named :BTP different from the class with the same name, and therefore is inadequate for metamodeling. This is evident if one considers the query

select $\$x\ \$y$ **where** $\{$:IT0005069395 rdf:type $\$x$. $\$x$:established_by $\$y\}$

asking for the law that established the type of financial instrument having a specific financial instrument (for example :IT0005069395) as instance. This query is not legal under DSER, because the variable $x appears both in individual and in class position. Note that if typing constraint is lifted, thus making the query legal, then it would become empty under DSER is trivially empty since DS cannot interpret any ontology element as both an individual and a class.

ClassAssertion(:BTP :IT0005069395)	(1)
ClassAssertion(:commercial_paper :ZAG000117292)	(2)
ClassAssertion(:type_of_Italian_f_i :BTP)	(3)
ClassAssertion(:type_of_foreign_f_i :commercial_paper)	(4)
ObjectPropertyAssertion(:established_by :BTP :DR135bis)	(5)
SubClassOf(:BTP :financial_instrument)	(6)
SubClassOf(:commercial_paper :financial_instrument)	(7)
SubClassOf(:financial_instrument DataSomeValuesFrom(:duration))	(8)
SubClassOf(:type_of_f_i ObjectSomeValuesFrom(:established_by :law))	(9)
SubClassOf(:type_of_Italian_f_i ObjectSomeValuesFrom(:established_by :Italian_law))	(10)
SubClassOf(ObjectSomeValuesFrom(ObjectInverseOf(:established_by)) :type_of_f_i)	(11)
SubClassOf(:type_of_Italian_f_i :type_of_f_i)	(12)
SubClassOf(:type_of_foreign_f_i :type_of_f_i)	(13)

Table 1: The ontology

Higher-Order Semantics for OWL 2 QL. A *vocabulary* V for an ontology is constituted by a tuple $V = (V_N, V_C, V_{OP}, V_{DP}, V_{DT}, \mathtt{L})$, where V_N is a set of IRIs that includes the reserved vocabulary R of OWL 2 QL, $V_C, V_{OP}, V_{DP}, V_{DT}$ are subsets of V_N, and L is the set of OWL 2 QL literals. The symbols in V_N are called *entity names*, since they are used to name all the entities of the ontology, while V_C (resp., V_{OP}, V_{DP}, V_{DT}) is the subset of V_N used to name those entities playing the role of class (resp., object properties, data properties, datatypes). Any symbol in $V_N \setminus \mathtt{R}$ may denote an entity that simultaneously plays the role of (*i*) an individual, (*ii*) either a class or a datatype, and (*iii*) either an object property or a data property. We denote by V_I the set $V_N \setminus (V_C \cup V_{OP} \cup V_{DP} \cup V_{DT})$, i.e., the set of entity names that can only play the role of individuals in the ontology. The symbols in V_N, also called *atomic expressions*, constitute the building blocks for denoting the entities of an ontology. Entities, however, can be denoted by using general *expressions*. The set Exp_V of expressions over V is the set $V_I \cup Exp_V^{OP} \cup Exp_V^C \cup V_{DP} \cup V_{DT}$, where $Exp_V^{OP} = V_{OP} \cup \{\mathtt{ObjectInverseOf}(e_1) \mid e_1 \in V_{OP}\}$, and $Exp_V^C = V_C \cup \{\mathtt{ObjectSomeValuesFrom}(e_1 e_2) \mid e_1 \in Exp_V^{OP}, e_2 \in V_C\} \cup \{\mathtt{DataSomeValuesFrom}(e_1 e_2) \mid e_1 \in V_{DP}, e_2 \in V_{DT}\}$. Expressions over V are then used in the axioms forming any ontology defined over the vocabulary V. HOS is based on the notion of interpretation structure, which plays the role of "interpretation domain" in classical logic. Given a vocabulary $V = (V_N, V_C, V_{OP}, V_{DP}, V_{DT})$, an *interpretation structure* for V is a tuple $\Sigma = (\Delta, \cdot^I, \cdot^E, \cdot^R, \cdot^A, \cdot^T)$

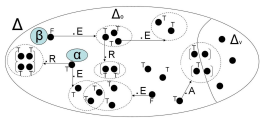


Fig. 1: Interpretation structure

Axiom a	$\mathcal{I} \models a$ **if**
SubClassOf($e_1 e_2$)	$\cdot^E$ is defined for both $e_1^{\mathcal{I}_o}$ and $e_2^{\mathcal{I}_o}$ and $(e_1^{\mathcal{I}_o})^E \subseteq (e_2^{\mathcal{I}_o})^E$
ObjectPropertyDomain($e_1 e_2$)	$\cdot^R$ is defined for $e_1^{\mathcal{I}_o}$, $\cdot^E$ is defined for $e_2^{\mathcal{I}_o}$ and $\forall \langle d, d' \rangle \in (e_1^{\mathcal{I}_o})^R : d \in (e_2^{\mathcal{I}_o})^E$
ClassAssertion($e_1 e_2$)	$\cdot^E$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = \mathtt{T}$ and $e_2^{\mathcal{I}_o} \in (e_1^{\mathcal{I}_o})^E$
ObjectPropertyAssertion($e_1 e_2 e_3$)	$\cdot^R$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = (e_3^{\mathcal{I}_o})^I = \mathtt{T}$ and $\langle e_2^{\mathcal{I}_o}, e_3^{\mathcal{I}_o} \rangle \in (e_1^{\mathcal{I}_o})^R$
DataPropertyAssertion($e_1 e_2 v$)	$\cdot^A$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = \mathtt{T}$ and $\langle e_2^{\mathcal{I}_o}, v^{\mathcal{I}_o} \rangle \in (e_1^{\mathcal{I}_o})^A$

Table 2: Satisfaction of OWL 2 QL axioms

where: (i) Δ is the disjoint union of two sets: Δ_o, the *object domain*, and Δ_v, the *value domain*; and (ii) $\cdot^E : \Delta_o \to \mathcal{P}(\Delta_o)$, $\cdot^R : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_o)$, $\cdot^A : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_v)$, and $\cdot^T : \Delta_o \to \mathcal{P}(\Delta_v)$ are partial functions, and $\cdot^I : \Delta_o \to \{\mathtt{T}, \mathtt{F}\}$ is a total function such that for each $d \in \Delta_o$, if $\cdot^E, \cdot^R, \cdot^A, \cdot^T$ are all undefined for d, then $d^I = \mathtt{T}$. Thus, the interpretation structure is a mathematical representation of a world made up by domain elements (the members of the set Δ) which can be either objects or values. Objects are polymorphic, in the sense that each of them can simultaneously be an individual(this is the case where function $\cdot^I$ is T), a set (this is the case where the partial function $\cdot^E$ is defined), a binary relation ($\cdot^R$ is defined), an attribute ($\cdot^A$ is defined), and a value type ($\cdot^T$ is defined). Also, an object that is a set (resp., relation, attribute, value type) is associated to its extension through $\cdot^E$ (resp., $\cdot^R$, $\cdot^A$, $\cdot^T$). Figure 1 shows a representation of an interpretation structure, where α is an individual, a set and a relation, whereas β is a set, but not an individual or a relation. An *interpretation* $\mathcal{I}$ for V is a pair, $\langle \Sigma, \mathcal{I}_o \rangle$, where Σ is an interpretation structure for V, and $\mathcal{I}_o$ is the *interpretation function* for $\mathcal{I}$, i.e., a function that maps every expression in Exp_V into an object in Δ_o, and every literal in L into a value in Δ_v, according to a set of suitable conditions. (e.g., $((\mathtt{ObjectInverseOf}(e_1))^{\mathcal{I}_o})^R = ((e_1^{\mathcal{I}_o})^R)^{-1}$). To define the semantics of axioms, we refer to the notion of satisfaction of an axiom with respect to an interpretation $\mathcal{I}$. Some of the rules defining such notion are specified in Table 2 (where e, e_1, e_2, e_3 and v are expressions).

SPARQL higher-order semantics entailment regime Defining a SPARQL entailment regime requires to specify (α) which is the semantics used to interpret the queried ontology, (β) which queries are legal, and (γ) a definition for the notion of answer to queries (called solution in SPARQL jargon). As for (α), we adopt HOS. As for (β), we extend the class of queries which are legal for DSER, by relaxing the typing constraint. As for (γ), we propose to interpret as pure existentials, variables that occur in the body of the query but not in the target, thus following the classical notion of existential variables in logic. Observe that the query mentioned in the scenario is legal for HOSER and that by evaluating it over the scenario ontology, it would return the answer $\langle$:BTP, :DR135bis$\rangle$. One of the results of our work is an algorithm, based on "blind metagrounding", showing that query answering over OWL 2 QL ontologies under HOSER has the same data complexity as under DSER. Nevertheless, such algorithm can be inefficient in practice. Hence, we devised a new algorithm that is polynomial with respect to extensional assertions and more efficient for a huge class of OWL 2 QL ontologies (so called *ISA-closed*) that are very common in practice.

References

1. D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier, 2011.
2. M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 14–26, 2014.
3. G. De Giacomo, M. Lenzerini, and R. Rosati. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*, 2011.
4. B. Glimm, S. Rudolph, and J. Völker. Integrated metamodeling and diagnosis in OWL 2. In *Proc. of ISWC 2010*, volume 6496 of *LNCS*, pages 257–272. Springer, 2010.
5. I. Kollia and B. Glimm. Optimizing SPARQL query answering over OWL ontologies. *J. Artif. Intell. Res. (JAIR)*, 48:253–303, 2013.
6. R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pages 552–567, 2014.
7. B. Motik. On the properties of metamodeling in OWL. *J. of Logic and Computation*, 17(4):617–637, 2007.
8. J. Z. Pan, I. Horrocks, and G. Schreiber. OWL FA: A metamodeling extension of OWL DL. In *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November 11-12, 2005*, 2005.

A Pragmatic Approach to Answering CQs over Fuzzy *DL-Lite*-ontologies–introducing FLite ⋆

Theofilos Mailis[1], Anni-Yasmin Turhan[2], and Erik Zenker[3]

[1] Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece
[2] Department of Computer Science, University of Oxford, UK
[3] Institute for Theoretical Computer Science,
Technische Universität Dresden, Germany

Abstract. Fuzzy Description Logics (FDLs) generalize crisp ones by providing membership degree semantics. To offer efficient query answering for FDLs it is desirable to extend the rewriting-based approach for *DL-Lite* to its fuzzy variants. For answering conjunctive queries over fuzzy *DL-Lite*$_R$ ontologies we present an approach, that employs the crisp rewriting as a black-box procedure and treats the degrees in a second rewriting step. This pragmatic approach yields a sound procedure for the Gödel based fuzzy semantics, which we have implemented in the FLite reasoner that employs the Ontop system. A first evaluation of FLite suggests that one pays only a linear overhead for fuzzy queries.

1 Introduction

Fuzzy variants of Description Logics (DLs) were introduced in order to describe concepts for which there exists no sharp, unambiguous distinction between members and nonmembers. For example, a natural way to model a component running at half of its capacity is to state that is overused to a degree of, say, 0.6.

In the last years, conjunctive query answering has been investigated for crisp and fuzzy DLs. The *DL-Lite* family of DLs [2, 6, 5] guarantees that query answering can be done efficiently —in the size of the data and in the overall size of the corresponding ontology. Though fuzzy *DL-Lite* variants have already been successfully investigated [18, 19, 10], their algorithms do not exploit the optimizations of query rewriting techniques that have been implemented in many systems for the crisp DLs, such as QuOnto2 [1, 12], Ontop [14], Owlgres [16], and IQAROS [22].

A pragmatic approach for answering conjunctive queries over crisp *DL-Lite*$_R$-TBoxes and fuzzy *DL-Lite*$_R$-ABoxes, that takes advantage of these optimizations, was presented in [9]. The combination of crisp TBoxes with fuzzy ABoxes is useful in applications, where only the data is imprecise, while the terminological knowledge is not, e.g., as in situation recognition applications that rely on sensor data.

⋆ Partially supported by DFG in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

In our pragmatic approach to answering of (fuzzy) conjunctive queries, a crisp *DL-Lite* reasoner is used as a black box to obtain an initial rewriting of the conjunctive query (without the fuzzy degrees). The obtained query gets extended in a second rewriting step by (1) fuzzy atoms, (2) degree variables that capture numerical membership degrees, and (3) numerical predicates that realize the fuzzy operators. The resulting query can then be evaluated by a SQL engine. This simple approach yields a sound and complete implementation only if fuzzy semantics with the min operator for conjunction is employed as, for instance, the popular Gödel semantics. For other semantics, at it will later be explained, our approach remains complete but not sound. We have implemented this approach in the FLite prototype, which uses the optimized Ontop reasoner to generate the first rewriting.

In this paper we describe the pragmatic approach, introduce some optimizations of it and study the scalability of FLite—in particular the overhead introduced by reasoning over fuzzy information. To this end we evaluate the FLite implementation against Ontop for different (fuzzy) conjunctive queries performed on a TBox and fuzzy ABoxes of different sizes. The benchmarks used for the evaluation are based on a situation recognition application for complex hard- and software systems.

The rest of the paper is structured as follows: Section 2 recalls *DL-Lite*$_R$ and its fuzzy variant. Section 3 presents a description of the two-step rewriting approach for answering fuzzy conjunctive queries and we discuss its limitations. In Section 4 we examine optimizations for this approach, which are implemented in FLite. Section 5 introduces our application of situation recognition that motivates the two-step rewriting approach for querying crisp TBoxes with fuzzy ABoxes and a benchmark from this application. The detailed evaluation of the FLite system against Ontop is given in Section 6 for different ontology sizes and different fuzzy queries. Conclusions and future work end the paper in Section 7.

2 Preliminaries

We start with *DL-Lite*$_R$ [2] and introduce its fuzzy variant [19, 10] afterwards. Let the following be countably infinite and pairwise disjoint sets: $\mathsf{N_C}$ of concept names, $\mathsf{N_R}$ of role names, $\mathsf{N_I}$ of individual names, and $\mathsf{N_V}$ of variable names. From these sets the complex *DL-Lite*$_R$-concepts, -roles and -queries are constructed. *DL-Lite*$_R$-concepts and -roles are defined according to the following grammar:

$$B \to A \mid \exists Q \qquad C \to \top \mid B \mid \neg B \qquad Q \to P \mid P^- \qquad R \to Q \mid \neg Q,$$

where $\top$ is the top concept, $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$. Based on these kinds of complex concepts and roles, a *DL-Lite*$_R$ TBox $\mathcal{T}$ is a finite set of axioms of the form: $B \sqsubseteq C$, $Q \sqsubseteq R$ or $funct(Q)$. Let $a, b \in \mathsf{N_I}$ and $d \in [0, 1]$ a *fuzzy degree*. A *fuzzy assertion* is of the forms: $\langle B(a), \geqslant d\rangle$ or $\langle P(a, b), \geqslant d\rangle$. An *ABox* $\mathcal{A}$ is a finite set of fuzzy assertions. A *fuzzy DL-Lite-ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. Please note that the TBoxes are always crisp in our setting.

Table 1. Families of fuzzy logic operators.

Family	t-norm $a \otimes b$	negation $\ominus a$	implication $\alpha \Rightarrow b$
Gödel	$\min(a, b)$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$	$\begin{cases} 1, & a \leqslant b \\ b, & a > b \end{cases}$
Łukasiewicz	$\max(a + b - 1, 0)$	$1 - a$	$\min(1 - a + b, 1)$
Product	$a \times b$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$	$\begin{cases} 1, & a \leqslant b \\ b/a, & a > b \end{cases}$

Crisp *DL-Lite*$_R$-ontologies are a special case of fuzzy ones, where only degrees 1 and 0 are admitted.

The reasoning problem we address is answering of (unions of) conjunctive queries. Let $t_1, t_2 \in \mathsf{N_I} \cup \mathsf{N_V}$ be terms, an *atom* is an expression of the form: $C(t_1)$ or $P(t_1, t_2)$. Let $\mathbf{x}$ and $\mathbf{y}$ be vectors over $\mathsf{N_V}$, then $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms of the forms $A(t_1)$ and $P(t_1, t_2)$. A *conjunctive query* (CQ) $q(\mathbf{x})$ over an ontology $\mathcal{O}$ is a first-order formula $\exists \mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ are the *answer variables*, $\mathbf{y}$ are *existentially quantified variables* and the concepts and roles in $\phi(\mathbf{x}, \mathbf{y})$ appear in $\mathcal{O}$. Observe, that the atoms in a CQ do not contain degrees. A *union of conjunctive queries* (UCQ) is a finite set of conjunctive queries that have the same number of answer variables.

The *semantics* of fuzzy *DL-Lite*$_R$ is provided via the different families of fuzzy logic operators depicted in Table 1 and interpretations. An *interpretation* for fuzzy *DL-Lite*$_R$ is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is as usual, but $\cdot^\mathcal{I}$ is an interpretation function mapping every

- $a \in \mathsf{N_I}$ to some element $a^\mathcal{I} \in \Delta^\mathcal{I}$,
- $A \in \mathsf{N_C}$ to a *concept membership function* $A^\mathcal{I} : \Delta^\mathcal{I} \to [0, 1]$,
- $P \in \mathsf{N_R}$ to a *role membership function* $P^\mathcal{I} : \Delta^\mathcal{I} \times \Delta^\mathcal{I} \to [0, 1]$.

Let δ, δ' denote elements of $\Delta^\mathcal{I}$ and $\ominus$ denote fuzzy negation (Table 1), then the semantics of concepts and roles are inductively defined as follows:

$$(\exists Q)^\mathcal{I}(\delta) = \sup_{\delta' \in \Delta^\mathcal{I}} Q^\mathcal{I}(\delta, \delta') \qquad (\neg B)^\mathcal{I}(\delta) = \ominus B^\mathcal{I}(\delta) \qquad \top^\mathcal{I}(\delta) = 1$$

$$P^{-\mathcal{I}}(\delta, \delta') = P^\mathcal{I}(\delta', \delta) \qquad (\neg Q)^\mathcal{I}(\delta, \delta') = \ominus Q^\mathcal{I}(\delta, \delta')$$

An interpretation $\mathcal{I}$ *satisfies* $B \sqsubseteq C$ iff $B^\mathcal{I}(\delta) \leqslant C^\mathcal{I}(\delta)$ for every $\delta \in \Delta^\mathcal{I}$, $Q \sqsubseteq R$ iff $Q^\mathcal{I}(\delta, \delta') \leqslant R^\mathcal{I}(\delta, \delta')$ for every $\delta, \delta' \in \Delta^\mathcal{I}$, and func($Q$) iff for every $\delta \in \Delta^\mathcal{I}$ there is a unique $\delta' \in \Delta^\mathcal{I}$ such that $Q^\mathcal{I}(\delta, \delta') > 0$. An interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$, i.e. $\mathcal{I} \models \mathcal{T}$, iff it satisfies all axioms in $\mathcal{T}$. $\mathcal{I}$ satisfies $\langle B(a), \geqslant d \rangle$ iff $B^\mathcal{I}(a^\mathcal{I}) \geqslant d$, and $\langle P(a, b), \geqslant d \rangle$ iff $P^\mathcal{I}(a^\mathcal{I}, b^\mathcal{I}) \geqslant d$. $\mathcal{I}$ is a *model of an ABox* $\mathcal{A}$, i.e. $\mathcal{I} \models \mathcal{A}$, iff it satisfies all assertions in $\mathcal{A}$. Finally an interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{A}$ and $\mathcal{T}$.

Given a CQ $q(\mathbf{x}) = \exists \mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$, an interpretation $\mathcal{I}$, a vector of individuals $\boldsymbol{\alpha}$ with the same arity as $\mathbf{x}$, we define the mapping π that maps: i) each individual

a to $a^{\mathcal{I}}$, ii) each variable in $\mathbf{x}$ to an element of $\boldsymbol{\alpha}^{\mathcal{I}}$, and iii) each variable in $\mathbf{y}$ to an element $\delta \in \Delta^{\mathcal{I}}$. Suppose that for an interpretation $\mathcal{I}$, Π is the *set of mappings* that comply to these three conditions. Computing the t-norm $\otimes$ of all atoms: $A^{\mathcal{I}}(\pi(t_1))$ and $P^{\mathcal{I}}(\pi(t_1), \pi(t_2))$ yields the degree of $\phi^{\mathcal{I}}(\boldsymbol{\alpha}^{\mathcal{I}}, \pi(\mathbf{y}))$. A tuple of individuals $\boldsymbol{\alpha}$ is a *certain answer* to $q(\mathbf{x})$, over $\mathcal{O}$, with a degree greater or equal than d (denoted $\mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d$), if for every model $\mathcal{I}$ of $\mathcal{O}$:

$$q^{\mathcal{I}}(\boldsymbol{\alpha}^{\mathcal{I}}) = \sup_{\pi \in \Pi}\{\phi^{\mathcal{I}}(\boldsymbol{\alpha}, \pi(\mathbf{y}))\} \geqslant d.$$

We denote the set of certain answers along with degrees, to a query $q(\mathbf{x})$ w.r.t. an ontology $\mathcal{O}$ with $ans(q(\mathbf{x}), \mathcal{O})$:

$$ans(q(\mathbf{x}), \mathcal{O}) = \{(\boldsymbol{\alpha}, d) \mid \mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d \wedge \nexists d'.d' > d \wedge \mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d'\}.$$

To illustrate the use of the fuzzy *DL-Lite*$_R$ language and queries, we provide an example from our application domain.

Example 1. The ontology $\mathcal{O}_{ex}$ for our running example consists of:

$$\begin{aligned}
\mathcal{T}_{ex} &:= \{\text{Server} \sqsubseteq \exists \text{hasCPU},\ \exists \text{hasCPU}^- \sqsubseteq \text{CPU},\ \text{func}(\text{hasCPU}^-)\} \\
\mathcal{A}_{ex} &:= \{\langle \text{Server}(\text{server}_1), \geqslant 1\rangle,\ \langle \text{hasCPU}(\text{server}_1, \text{cpu}_1), \geqslant 1\rangle, \\
&\quad \langle \text{OverUsed}(\text{cpu}_1), \geqslant 0.6\rangle,\ \langle \text{hasCPU}(\text{server}_1, \text{cpu}_2), \geqslant 1\rangle, \\
&\quad \langle \text{OverUsed}(\text{cpu}_2), \geqslant 0.8\rangle \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}
\end{aligned}$$

The first two axioms in $\mathcal{T}_{ex}$ state that each server has a part that is a CPU. The third one states that no CPU can belong to more than one server. $\mathcal{A}_{ex}$ provides information about the connections between servers and CPUs and each CPU's degree of overuse. To query the ontology $\mathcal{O}_{ex}$ we can formulate the queries:

$$\begin{aligned}
q_1(x, y) &= \text{hasCPU}(x, y) \wedge \text{OverUsed}(y) \\
q_2(x) &= \exists y\ \text{hasCPU}(x, y) \wedge \text{OverUsed}(y)
\end{aligned}$$

The query q_1 asks for pairs of Servers and CPUs with an overused CPU. The query q_2 asks for Servers, where the Server's CPU is overused. If conjunction and negation are interpreted as the Gödel family of operators, the certain answers w.r.t. $\mathcal{O}_{ex}$ are:

$$\begin{aligned}
ans(q_1(x, y), \mathcal{O}_{ex}) &= \{(\text{server}_1, \text{cpu}_1, 0.6),\ (\text{server}_2, \text{cpu}_2, 0.8)\} \\
ans(q_2(x), \mathcal{O}_{ex}) &= \{(\text{server}_1, 0.8)\}.
\end{aligned}$$

3 Fuzzy Query Answering by Extended Crisp Rewritings

Let CQ $q(\mathbf{x})$ be formulated over the vocabulary of the *DL-Lite*$_R$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. The main idea underlying the classic *DL-Lite*$_R$ query answering algorithm is to rewrite the query $q(\mathbf{x})$ with the information from the TBox $\mathcal{T}$ into a UCQ $q_{\mathcal{T}}(\mathbf{x})$ and then apply this UCQ to the ABox $\mathcal{A}$ alone [6, 2]. For fuzzy

DLs we extend this approach to handle degrees of ABox assertions. The main idea is depicted in Figure 1. To explain the algorithm we need the predicates A_f, P_f, and $\Phi_\otimes$. Intuitively, each binary predicate A_f is an extension of the unary predicate A such that the fuzzy concept assertion $\langle A(a), \geqslant d\rangle$ is equivalent to the predicate assertion $A_f(a, d)$ (similarly for P_f). The n-ary predicate $\Phi_\otimes$ is adopted in order to realize the semantics of the fuzzy conjunction (e.g. minimum for the Gödel norm) within a FOL query. Thus, for each tuple of degrees $d_1, \ldots, d_n \in [0, 1]$ such that $d_1 = \otimes(d_2, \ldots, d_n)$, we have that $(d_1, \ldots, d_n) \in \Phi_\otimes$. Suppose now that the CQ $q(\mathbf{x})$ is to be answered. The two-step rewriting algorithm proceeds as follows:

1. The crisp *DL-Lite*$_R$ algorithm rewrites $q(\mathbf{x})$ to $q_\mathcal{T}(\mathbf{x})$. (For ease of presentation we assume that $q_\mathcal{T}(\mathbf{x})$ is still a CQ.)
2. The fuzzy query $q_{\mathcal{T},f}(\mathbf{x}, x_d)$ is computed from $q_\mathcal{T}(\mathbf{x})$ by replacing atoms of the form $A(t_1)$ and $P(t_1, t_2)$ by $A_f(t_1, y_d)$ and $P_f(t_1, t_2, y_d)$, where the variable y_d is a *degree variable*. Its purpose is to retrieve the degree of an assertion. The degree value for fuzzy conjunction is retrieved by the predicate $\Phi_\otimes$ and (to be) stored in the additional degree variable x_d. Thus, the conjunction degree of a new atom $q_{\mathcal{T},f}(\mathbf{x}, x_d)$ is obtained by the predicate $\Phi_\otimes(x_d, y_1, \ldots, y_n)$, where y_i is a degree variable in the ith atom of the CQ.
3. The query is evaluated over the ABox and the actual computation of the degree values takes place. Now, for a tuple of individuals $\boldsymbol{\alpha}$ and degrees $d_1, d_2 \in [0, 1]$, if (α, d_1) and (α, d_2) are both answers to the query, only the answer with the higher degree is returned.

Note that this description abstracts from the fact that the ABox $\mathcal{A}$ is usually implemented by a relational database $\mathcal{D}$ and a mapping $\mathcal{M}$. We see in Section 3.1 how this mapping is extended to incorporate fuzzy information. For a more detailed presentation of the algorithms, the reader may refer to [9].

Example 2. We return to Example 1 and illustrate the application of the algorithm to the queries. Initially, q_1 and q_2 are rewritten to the following UCQs:

$$q_{1\mathcal{T}_{ex}}(x, y) = \{\text{hasCPU}(x, y) \wedge \text{OverUsed}(y)\}$$
$$q_{2\mathcal{T}_{ex}}(x) = \{\exists y.\text{hasCPU}(x, y) \wedge \text{OverUsed}(y)\}$$

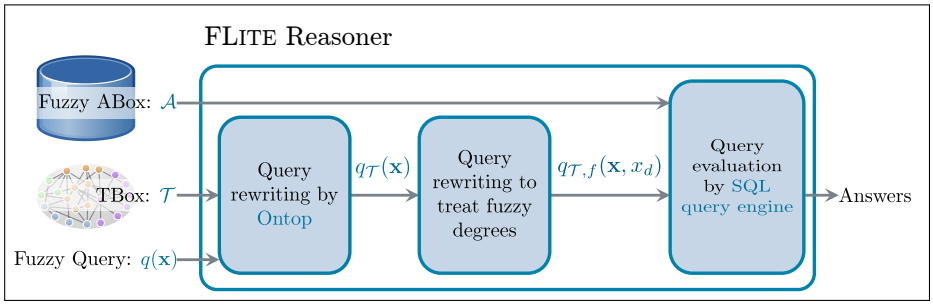


Fig. 1. The process flow diagram of the FLITE rewriting procedure.

In the next step, the algorithm extends the queries with degree variables and atoms, so that the corresponding degrees can be returned:

$$q_1{}^{f}_{\mathcal{T}_{ex}}(x, y, x_d) = \{\text{hasCPU}(x, y, y_{d_1}) \wedge \text{OverUsed}(y, y_{d_2}) \wedge \Phi_{\otimes}(x_d, y_{d_1}, y_{d_2})\}$$
$$q_2{}^{f}_{\mathcal{T}_{ex}}(x, x_d) = \{\exists y.\text{hasCPU}(x, y, y_{d_1}) \wedge \text{OverUsed}(y, y_{d_2}) \wedge \Phi_{\otimes}(x_d, y_{d_1}, y_{d_2})\}$$

For the ABox $\mathcal{A}_{ex}$ the following set of answers to each of the queries are returned:

$$ans(q_1{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) = \{(server_1, cpu_1, 0.6), (server_1, cpu_2, 0.8)\}$$
$$ans(q_2{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) = \{(server_1, 0.8)\}.$$

The limitations of our pragmatic approach are explained in [9]. To sum up, our approach yields sound and complete results for fuzzy semantics based on the min t-norm operator such as the Gödel family of operators. The correctness for this case can be derived from the crisp *DL-Lite*$_R$ proof along with the following points: (1) only crisp TBox axioms are allowed, (2) conjunctions only appear in conjunctive query expressions, (3) Ontop optimizations do not affect the correctness of the algorithm due to the properties of the min operator. To illustrate the last point a conjunctive query $q_{\mathcal{T}}(x) := A(x) \wedge A(x)$ is simplified by Ontop to $q_{\mathcal{T}}(x) := A(x)$. This simplification is correct for the min operator since $\min(A^{\mathcal{I}}(\delta), A^{\mathcal{I}}(\delta)) = A^{\mathcal{I}}(\delta)$ (for every interpretation $\mathcal{I}$ and every $\delta \in \Delta^{\mathcal{I}}$) . The latter does not apply for other t-norms, therefore the proposed methodology is complete but not sound for the Łukasiewicz and Gödel families of operators. Nevertheless, as described in [9], we devised a method by which each unsound answer can be identified and its correct degree estimated between two membership values. I.e. our algorithm asserts that for some $d \in [0.7, 0.8]$, $(server_1, d) \in ans(q_3{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex})$.

3.1 The FLite Reasoner Implementation

FLite[4] (Fuzzy *DL-Lite*$_R$ query engine) implements the presented query answering algorithm and builds on the Ontop framework [14]. On a technical level, the rewriting procedure becomes more involved when a reasoner such as Ontop is deployed, since such systems are build to operate on relational databases. Thus the queries $q_{\mathcal{T}}(\mathbf{x})$ and $q_{\mathcal{T},f}(\mathbf{x}, x_d)$ in Figure 1 are SQL queries, while the ABox $\mathcal{A}$ is derived from a partial mapping:

$$\mathcal{M} : \text{SQL SELECT Statements} \rightarrow \text{ABox assertions.}$$

In order to embed fuzzy information into mappings we adopt a reification approach sketched in the following example.

Example 3. We consider a fuzzy mapping described in the Quest syntax [13]. In this mapping, for the concept popularVideo each id of the table videos is

[4] The FLite reasoner is available from the following Git: https://iccl-share.inf.tu-dresden.de/flite-developer/flite.git

annotated with a popularity, i.e. a fuzzy degree:

```
target haec:video_{popularity_degree} a haec:PopularVideo.
source SELECT   fuzzy(id, popularity)
       AS       popularity_degree
       FROM     videos
```

Now, if a video with an id of 12 and a popularity of 0.8 appears in the database, then this corresponds to $\langle \text{PopularVideo}(\mathit{videos}12), \geqslant 0.8 \rangle$ stated in the ontology.

The function *fuzzy(column, degree)* is a marker for the FLite parser to recognize fuzzy statements. It indicates that every element of the particular *column* (or SQL expression) gets a fuzzy *degree* assigned. This degree can either be a column with values in $[0, 1]$, or an SQL expression corresponding to a fuzzy membership function. It should be noted that SQL expressions containing the fuzzy marker function appear in the initial mapping $\mathcal{M}$ and in $q_{\mathcal{T}}(\mathbf{x})$ queries while these markers are converted in $q_{\mathcal{T},f}(\mathbf{x}, x_d)$ queries to a SQL expressions that return the membership degree.

4 Optimizations for the FLite Implementation

Implementing the pragmatic approach naively would be very inefficient. First, in contrast to Ontop's rewritings, the "fuzzy" SQL query resulting from the FLite rewriting process is not optimized for fast execution. Thus, the query engine retrieves the same items multiple times. Second, the database contains numerical values, that are mapped to coarser categories with membership degrees at query execution time. These overheads can be reduced by optimizing the fuzzy rewritings and fuzzifying numerical values in the database on a preprocessing step. These optimizations are discussed in the following.

Self-join Removal (SR). Ontop performs a restructuring of query rewritings as outlined in [15, 7]. Due to the reification process for embedding fuzzy information, some of the optimizations performed by Ontop are obscured by the *fuzzy* pseudo-function. Currently, Ontop appears to omit the optimization step, if an unknown function is found in the query. Thus, its optimizer is not aware of the fact that the *fuzzy* function is exclusively used by FLite to tag fuzzy degrees in the query. Extending the process shown in Figure 1, the query optimizer should be applied to the query $q_{\mathcal{T},f}(\mathbf{x}, x_d)$, e.g. to remove self-joins. To illustrate the problem, consider a query $q_{\mathcal{T},f}$ that contains self-joins of the following form:

```
SELECT a.column_a, a.degree_a, b.column_b
FROM table_A as a,
INNER JOIN table_A as b ON a.column_a=b.column_a
```

It can easily be verified that this query contains redundant selection statements over the same database table and can be simplified to:

```
SELECT column_a, degree_a, column_b FROM table_A
```

The second query is performed in linear time w.r.t. the size of table_A, while the first one is performed in quadratic time. Indeed, we observed a huge improvement on query execution time, depending on the number of tables that are self-joined.

Pre-computation of Membership Degrees (PD). Calculation of membership degrees of numerical values during query execution can lead to significant run-time overhead depending on the complexity of the applied membership function. The following listing shows an SQL statement where fuzzy degrees are assigned to the values of a column by the membership degree function f:

```
SELECT fuzzy(column_a, f(column_b)) FROM crisp_table
```

The case where membership degrees are available directly , reduces the overhead of calling and executing such a function. In the following SQL statement, function f was replaced by a membership degree column.

```
SELECT fuzzy(column_a, membership_degree_column)
FROM fuzzy_table
```

For simple mappings of this form, database schema restructuring is unnecessary, since the observed raw data in our application database are often numerical values, an automatic mechanism that translates these values to membership degrees is necessary. For this purpose we introduce computed or virtual columns, which contain values that are computed by a user-defined membership function, which is triggered when a new row is added to the table. This avoids the overhead for the computation of membership function during the evaluation of the rewritten query at the cost of an increase in database size and membership function overhead during the database update process. In the end he FLite user has to assess on the basis of the mapping whether to spend more time for query execution or more disc space for additional fuzzy columns.

5 A Sample Application: Situation Recognition

The project "Highly Adaptive Energy-efficient Computing" (HAEC) investigates complex computing environments that are highly energy-efficient while compromising utility of services as little as possible. In order to be adaptive, the system needs to trigger adaptations (of hard- or software components), if the quality of the requested services or their number changes. To provide such a trigger mechanism we investigate an ontology-based situation recognition. The situations to be recognized are modelled as conjunctive queries. The background information on the system is captured in the TBox and the current system's state is captured by an ABox. Such ABoxes are automatically generated from sensor data and other systems information and the conjunctive queries for the situations are evaluated. In such a setting the numerical sensor data need to be mapped to coarser, symbolic categories and membership degrees. Similarly, the query needs to be able to retrieve individuals that fulfill the conditions of the query to a degree. We have built a TBox and a collection of ABoxes and queries for this application.

5.1 Towards a Benchmark

The hard- and software components of the HAEC system are modeled in a TBox which consists of 197 GCIs, 168 named classes and 38 roles (415 axioms total). Each state of the HAEC system is stored in tables of a relational database. This database stores the information on the soft- and hardware of the HAEC system. The tables contain numerical values for boards, processes, requests, etc.[5] Identifying the HAEC ontology as TBox and the HAEC database as ABox, this benchmark compares the run-time to answer (fuzzy) conjunctive queries over TBox and ABox by Ontop and FLite. In contrast to Ontop which requires a crisp mapping, FLite is using a partially fuzzy mapping.

6 Results of FLite on the Benchmark

FLite was evaluated over a series of variations of the HAEC benchmarks. The number of additional atoms in the conjunctive query n and the database scale factor k were increased in the series.

The initial database with a size of 768.0 KiB was scaled by k in the range of 10^0 to 10^5 to about 13 GiB by the Virtual Instance Generator (VIG) [8]. For each of these scaled databases, the number of atoms of the initial conjunctive query (2 concepts, 4 roles) was increased by n additional atoms from one atom to a maximum of seven (13 atoms total). The system on which the benchmark was performed on is powered by an Intel Core i7 2.6 GHz processor and was equipped with 8 GB DDR 1600 main memory. ABox information was stored in a MySQL 5.6.23 database. FLite and Ontop are executed on Oracle the JVM 1.7.

Figure 2 shows the query execution time of Ontop for a query with seven additional atoms and with a crisp mapping compared to the naive FLite implementation for a query with one to seven additional atoms with fuzzy mapping using Gödel t-norms for an increasing database size.
FLite needs longer run-time in the order of almost two magnitudes compared to the one of Ontop. Therefore, it is vital that the naive FLite implementation is enhanced by optimization techniques. Employing the optimizations introduced in Section 4 results in the query execution times displayed in Figure 3.

Optimization SR in Figure 3 shows run-time reduction up to a scale factor of about thousand, then it converges with the naive FLite implementation. Thus, an optimization that also effect larger databases is necessary. Optimization PD in Figure 3 shows opposite behavior, resulting in a run-time reduction from a scale factor of thousand. Finally, a combination of both reduces the run-time on all scale factors and is even on the same level with Ontop up to a scale factor of thousand. Thus, a query execution time with a flat linear run-time overhead with respect to Ontop is possible when the fuzzy rewriting is optimized and fuzzy translation functions are replaced by fuzzy computed columns.

[5] The files necessary to perform this benchmark can be found here: https://iccl-share.inf.tu-dresden.de/erikzenker/flite-benchmark.git

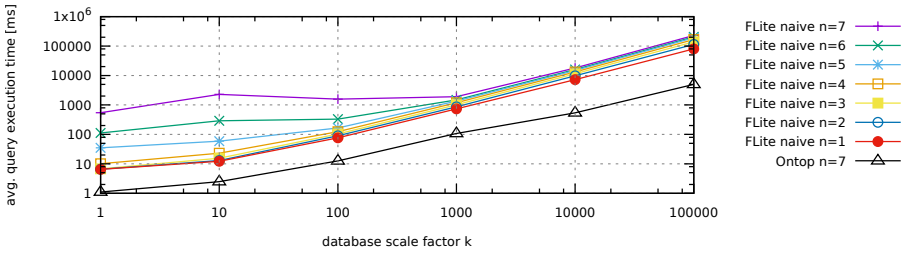


Fig. 2. Query execution time of Ontop with seven additional atoms with crisp mapping compared to the naive FLite implementation with one to seven additional atoms with fuzzy mapping.

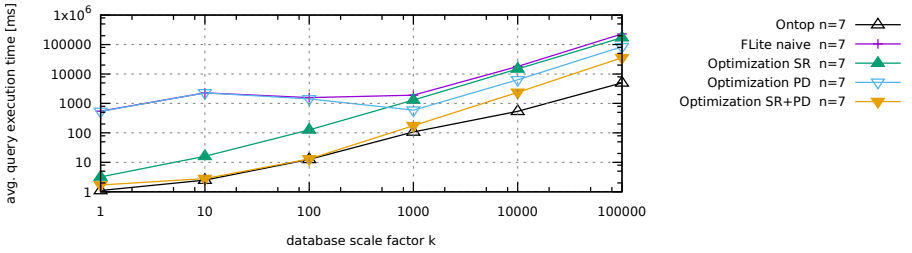


Fig. 3. Ontop is compared to optimized FLite versions SR, PD, and SR+PD. All setups were executed over a CQ with seven additional atoms.

Instead of comparing FLite with Ontop, a comparison with other fuzzy DL reasoners would have been desirable. However, reasoners such as LiFR [21], FuzzyDL [4], FiRE [17], and DeLorean [3] support only instance checking rather than conjunctive query answering (albeit for more expressive DLs than *DL-Lite*$_R$). Others such as the *DL-Lite* reasoner Ontosearch2 [11] or SoftFacts [20] could either not be obtained or installed. Thus we had to resort to Ontop for a comparison for query answering in *DL-Lite*$_R$.

7 Conclusions

We presented a pragmatic approach for answering fuzzy conjunctive queries over *DL-Lite*$_R$-ontologies with fuzzy ABoxes. Our approach uses rewritings obtained by the algorithm for answering crisp queries as an intermediate step and thus allows to make use of standard query rewriting engines. Although described here for *DL-Lite*$_R$, our approach can be extended to other DLs that enjoy FOL rewritability. Our algorithm is sound for those t-norms that have idempotent operators, such as the Gödel t-norm.

We implemented our approach in the FLite system and evaluated it against the Ontop reasoner for ABoxes of varying size. Our evaluation gave evidence that there is a substantial increase of run-time for large ABoxes, when fuzzy

information is queried. This increase can be reduced by basic optimizations. However, developing and extending FLite is on-going work.

References

1. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: Querying Ontologies. In *AAAI*, pages 1670–1671, 2005.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* Family and Relations. *Journal of artificial intelligence research*, 36(1):1–69, 2009.
3. F. Bobillo, M. Delgado, and J. Gómez-Romero. Reasoning in Fuzzy OWL 2 with DeLorean. In *Uncertainty Reasoning for the Semantic Web II*, pages 119–138. 2013.
4. F. Bobillo and U. Straccia. fuzzyDL: An Expressive Fuzzy Description logic Reasoner. In *FUZZ-IEEE*, pages 923–930, 2008.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, and R. Rosati. *Ontologies and Databases: The DL-Lite Approach*. 2009.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *Journal of Automated reasoning*, 39, 2007.
7. R. Kontchakov, M. Rezk, M. Rodríguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL Queries over Databases under OWL 2 QL Entailment Regime. In *The Semantic Web–ISWC 2014*, pages 552–567. Springer, 2014.
8. D. Lanti, M. Rezk, M. Slusnys, G. Xiao, and D. Calvanese. The NPD Benchmark for OBDA Systems. In *10th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2014)*, page 3, 2014.
9. T. Mailis and A.-Y. Turhan. Employing DL-Lite$_R$-Reasoners for Fuzzy Query Answering. In *Proceedings of the 4th Joint International Semantic Technology Conference (JIST2014)*, LNCS, 2014.
10. J. Z. Pan, G. B. Stamou, G. Stoilos, and E. Thomas. Expressive Querying over Fuzzy DL-Lite Ontologies. In *Description Logics*, 2007.
11. J. Z. Pan, E. Thomas, and D. Sleeman. Ontosearch2: Searching and querying web ontologies. *Proc. of WWW/Internet*, 2006:211–218, 2006.
12. A. Poggi, M. Rodriguez, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In *Proc. of OWLED*, 2008.
13. M. Rodriguez-Muro, J. Hardi, and D. Calvanese. Quest: Efficient SPARQL-to-SQL for RDF and OWL. In *11th International Semantic Web Conference ISWC 2012*, page 53, 2012.
14. M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-Based Data Access: Ontop of Databases. In *International Semantic Web Conference (1)*, volume 8218 of *LNCS*, pages 558–573. Springer, 2013.
15. M. Rodriguez-Muro, M. Rezk, J. Hardi, M. Slusnys, T. Bagosi, and D. Calvanese. Evaluating SPARQL-to-SQL Translation in Ontop. In *Informal Proceedings of ORE'13*, pages 94–100, 2013.
16. M. Stocker and M. Smith. Owlgres: A Scalable OWL Reasoner. In *OWLED*, volume 432, 2008.
17. G. Stoilos, N. Simou, G. Stamou, and S. Kollias. Uncertainty and the Semantic Web. *Intelligent Systems*, 21(5):84–87, 2006.
18. U. Straccia. Answering Vague Queries in Fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems,(IPMU-06)*, pages 2238–2245, 2006.

19. U. Straccia. Towards Top-k Query Answering in Description Logics: The Case of DL-Lite. In *Logics in Artificial Intelligence*, pages 439–451. Springer, 2006.
20. U. Straccia. SoftFacts: A Top-k Retrieval Engine for Ontology Mediated Access to Relational Databases. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 4115–4122. IEEE, 2010.
21. D. Tsatsou, S. Dasiopoulou, I. Kompatsiaris, and V. Mezaris. LiFR: A Lightweight Fuzzy DL Reasoner. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 263–267. 2014.
22. T. Venetis, G. Stoilos, and G. Stamou. Query Extensions and Incremental Query Rewriting for OWL 2 QL Ontologies. *Journal on Data Semantics*, pages 1–23, 2014.

Empirical Investigation of Subsumption Test Hardness in Description Logic Classification

Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{matentzn,bparsia,sattler}@cs.manchester.ac.uk

Abstract. Recently, modular techniques have been employed for optimising Description Logic reasoning, specifically to enable incremental reasoning and improve overall classification time. Classifying a module of an ontology should be significantly easier than reasoning in the whole ontology. However, we observed in previous work that neither it is generally true that modular reasoning techniques have a reliable positive effect, nor even that the classification time of a module is less than or equal to the classification time of the whole ontology. One possible explanation for the latter could be that counter-productive optimisations are triggered within the reasoner when dealing with the sub-module, and thus individual subsumption tests get harder when parts of the ontology are missing. The goal of this paper is to understand the contribution of subsumption tests to the hardness of classification. The contribution is twofold: (1) We analyse the impact of subsumption test hardness on DL classification by analysing a well known corpus of ontologies, and (2) we present a novel approach based on modularity to robustly detecting subsumption tests that are *too hard*.

Keywords: classification,ontologies,benchmarking,modular reasoning

1 Introduction

Reasoning in popular, very expressive Description Logics (DL) is very difficult (e.g., $\mathcal{SROIQ}$ is N2Exptime-complete) [12]. Perhaps surprisingly, modern reasoning systems suitable for the entirety of OWL 2 DL (essentially a notational variant of $\mathcal{SROIQ}$) such as FaCT++ [20], Pellet [18], HermiT [5] and recently Konclude [19] generally perform well against real ontologies. However, due to the poor performance in some (often important) cases, the quest for optimisations is ongoing. The need to empirically validate such optimisations stems from the sheer complexity of reasoner architectures. Worst case complexity analysis and its variants do not account for the high variability of classification times of real ontologies. Modern reasoning systems have to accommodate multiple reasoning services and also tend to implement a wide range of optimisation techniques that might affect each other. Various sources of non-determinism, mainly traversal (subsumption test order) and or-branch exploration of a tableau further complicate the situation. Statistical methods such as linear regression [16] tend to

be only precise for trivial cases, and are limited in their explanatory richness. Currently, principled benchmarking provides the only way to creating detailed characterisations of DL reasoning performance.

Using locality-based modules to optimise Description Logic classification experienced a resurgence in recent years [15, 21]. Intuitively, breaking the input problem into smaller pieces, reasoning over those pieces separately, then recombining the results is appealing. Furthermore, if there are especially difficult parts of the ontology, perhaps they can be isolated to reduce their impact. In practice however, modular reasoning techniques do not always improve the performance of classification [6]. In fact, they can drastically impair performance, making it a hit and miss game to chose between a modular reasoner (e.g. MORe-HermiT, Chainsaw-JFact) and its monolithic counterpart (e.g. HermiT, JFact). These cases can often be due to various kinds of overhead induced by modular reasoners (module extraction) or redundancy introduced by the mostly unavoidable and often significant overlap between the various modules extracted. In a preliminary set of experiments [13] we observed that not only are there cases where there are individual subsumption tests that can be, often significantly, harder in a module extracted by a modular reasoner than in the whole ontology, but also that there are occasionally modules whose classification time exceeds that of the entire ontology $\mathcal{O}$ it was extracted from.

The **goal** of this paper is to understanding the contribution of subsumption tests (ST) to the hardness of classification. The **contribution** is twofold: (1) We analyse the impact of ST hardness on DL classification by characterising a well known corpus of ontologies, and (2) we present a novel approach based on modularity to robustly detecting subsumption tests that are potentially *too hard*. As a result, we re-confirm the almost 20 years old results by Horrocks [11] that ST's are generally rather easy. We also isolate counter-intuitive instances that, however, are often likely to be the consequence of the surprising degree of observed stochasticity in the classification process.

2 Background

Understanding the experimental design and methodology presented here does not require more than a cursory understanding of the syntax, semantics, and proof theories implemented. The most prominent families of reasoning algorithms for description logics are tableau (incl. hyper-tableau) and consequence-based. In our work, we are mainly concerned with tableau-based algorithms. The reasoners under investigations in this paper are designed to implement key reasoning services for the Web Ontology Language (OWL), most importantly classification and consistency checking. Given an ontology (a set of axioms) $\mathcal{O}$, the signature of an ontology $\widetilde{O}$ is the set names appearing in the axioms in $\mathcal{O}$. We use $CT(\mathcal{O})$ (classification time) and $CT(\mathcal{M})$ respectively to denote the time of computing the set of atomic subsumptions (i.e., statements of the form $A \sqsubseteq B$ where A and B are properties or classes in the signature) or *classification* of $\mathcal{O}$. For brevity, we refer to overall classification time as OCT and subsumption test time as STT.

While subsumption testing, and therefore classification, is in theory intractable, highly optimised reasoners do fairly well in practice. The observed efficiency despite the worst case complexity is in principle down to four factors. (1) Real ontologies are bounded in size. That means that even an exponential algorithm might fully classify an ontology, from a user perspective, in an acceptable amount of time. (2) Many ontologies fall into tractable fragments of OWL, and can be classified using efficient polynomial algorithms such as the ones from the family of consequence-based algorithms. (3) The last 20 years brought a plethora of different optimisations to make *satisfiability checks easier* [11, 3]. (4) Very efficient algorithms were developed to avoid the vast majority of subsumption tests altogether [1, 4, 17].

Current modular classification approaches use so-called syntactic locality-based $\bot$-modules [10] which have a number of desirable properties: (1) They are relatively cheap to extract and are reasonably compact and exact, (2) If $\mathcal{O} \models A \sqsubseteq C$ then for any given $\bot$-module $\mathcal{M}_\bot$, of $\mathcal{O}$ where $A \in \widetilde{\mathcal{M}_\bot}$, $\mathcal{M}_\bot \models A \sqsubseteq C$ (were C is an arbitrary expression over the signature of $\mathcal{O}$). Thus, $\bot$-modules are classification complete for their signature with respect to their parent ontology. Hereafter, we will use $\mathcal{M}$ to refer to a syntactic locality based $\bot$-module.

Very recently, reasoner developers have started to **utilise modularity for classification**. They either are (1) using modules for incremental reasoning [9] or (2) using modules to improve classification time [15, 21].

3 Related Work

Attempts to understand DL reasoning performance are, up until today, rarely systematic or comprehensive. Recently, the ORE reasoner competition tries to establish the methodological foundations for more reliable comparisons [6] between different reasoners and across a range of different reasoning services. OWL Reasoner benchmarks have been conducted for varying purposes, for example (and most prominently) guiding end-users for selecting appropriate reasoners for their problem [2, 6] or understanding reasoning or the state of reasoning in general [7]. Dentler et al. [2] conduct a principled investigation to identify suitable criteria for choosing an appropriate reasoner for EL ontologies. In our work, we are interested in mapping out subsumption test hardness during full classification across reasoner-ontology pairs (phenomenological characterisation) and the potential of modularity to pinpoint counter-intuitive cases (i.e. harder tests in a sub-module). Most benchmarks conduct an only semi-principled dataset selection: Even carefully executed benchmarks such as Dentler et al. [2] usually cherry pick a set of *somehow* relevant ontologies. Few works sample from existing corpora or the web, and only Gonçalves et al. [7], to the best of our knowledge, deal with corpora larger than 500 ontologies. In practice, the current de facto gold-standard corpus for ontology experimentation is BioPortal [14], which also provides a well designed infrastructure to obtain an interesting range of biomedical ontologies programatically. We are using a snapshot of BioPortal in this work. As far as we know, no benchmark to date has investigated *subsumption testing*

during classification across reasoners in a principled manner. However, various benchmarks have investigated the effect of certain optimisations on subsumption test avoidance [4]. While the literature on classification optimisation and reasoning is vast, little progress has been made in understanding classification hardness of real ontologies, both empirically and formally.

4 Subsumption Test Hardness

The phenomenon under investigation is **subsumption test hardness in the context of classification**. A **subsumption test** is a question asked by the reasoner to determine whether $A \sqsubseteq B$. The **subsumption test hardness** is the time it takes to compute the answer, operationalised as wall-clock time. In this work the answer to a test is either yes or no. Note however, that for any implementation (1) more than just a binary answer will be provided (i.e., cached models, derived subsumptions) and (2) no guarantee is given that the answer is correct (bugs in the reasoner). "In the context of classification" means that we are not exploring individual "cold" tests, i.e. letting the reasoner compute whether $A \sqsubseteq B$ for any A, B from outside the classification process, because we want to understand the contribution of subsumption testing to classification as a whole, with all the optimisations involved.

Our **model of subsumption test hardness** with respect to sub-modules is based on the following intuition: Given a *positive* ST $\mathcal{ST}$, it should be the case that for every two modules $\mathcal{M}_1$,$\mathcal{M}_2$ with $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \mathcal{O}$ in which the ST is triggered, the hardness of $\mathcal{ST}$ always stays the same. The reason for that are module properties: every justification for an entailment is part of every module that entails it. Thus, every way that the entailment holds is contained in the module, no "new" information about the entailment exists in the rest of the ontology. Intuitively, additional "stuff" can make it *harder* to figure out the entailment, but not make it easier. This makes this metric a possible indicator of counter-productive optimisations: If we find that $\mathcal{ST}_{\mathcal{M}_2}$ is harder than $\mathcal{ST}_{\mathcal{M}_1}$, we might conclude that the reasoner is doing some unnecessary extra work in $\mathcal{M}_2$ (case 1); if $\mathcal{ST}_{\mathcal{M}_1}$ is harder than $\mathcal{ST}_{\mathcal{M}_2}$, there is a possibility that a counter-productive optimisation may have been triggered (case 2). Only the second case is truly pathological: A test should never get harder when *irrelevant* axioms are removed from the ontology. The first case might simply occur because if $\mathcal{M}$ grows, it gets harder to identify the irrelevant axioms. One possibility to explain both cases may be the inherent stochasticity of classification as implemented by current OWL Reasoners. For example, a random factor might (for example by changing the test order) simply shift the load of $\mathcal{ST}$ in $\mathcal{M}_1$ to another ST $\mathcal{ST}_2$ that consecutively makes $\mathcal{ST}$ easier. Another reason for a test becoming easier in a sub-module might be the exploitation of partial results from negative tests (e.g. caching).

Our empirical investigation of subsumption tests has two parts: (A) a broad characterisation of the landscape of subsumption testing and (B) an in-depth characterisation of non-easy subsumption tests. We treat a test as non-easy if it

takes longer than 100 ms . The first **part A** will attempt to answer the following questions: What is the impact of subsumption testing on reasoning performance in general (RQ1)? How many tests are positive or negative and how do they differ in hardness (RQ2)? How hard are real tests actually (RQ3)?

Part B serves as an in-depth characterisation that attempts to address questions related to the general stability of the measurements (**intra-module**) and the effect of modularity (**inter-module**). Is ST hardness a stable phenomenon (RQ4)? This is important in order to judge how reliably we can trace a single subsumption test through different sub-modules of an ontology, and may also give a warning sign for triggered non-determinism, for example in the case that a test appears or disappears given a particular ontology-reasoner pair across runs. We will address this problem mainly by looking at the *coefficient of variation* (COV) of subsumption test hardness. The COV is a statistical, standardised measure of dispersion of a distribution (for example the distribution of test hardness) defined as the ratio of the standard deviation to the mean and is used to compare the variation of one data series to another, even if they are on a different scale. What are the reasons for instability (RQ5)? We will not conclusively try to answer this problem, but we will collect some evidence for stochasicity by looking at intra-module variation of test counts, a strong indicator of non-determinism.

Does modularity change the hardness of tests (RQ6)? In order to answer this question, we will classify tests by analysing how modularity affects their hardness. This happens as follows: We identify all super and submodule combinations $\mathcal{M}_1, \mathcal{M}_2$ as described earlier. For each test triggered in both $\mathcal{M}_1$ and $\mathcal{M}_2$, we determine: (1) was the effect positive on average (across runs), (2) what was the magnitude of the effect and (3) was the effect stable? We define **stability of an effect** as follows: given a subsumption test $\mathcal{ST}$ that occurs in two modules $\mathcal{M}_1, \mathcal{M}_2$ with $\mathcal{M}_1 \subset \mathcal{M}_2$, and two sets of measurements $X(\mathcal{ST}_{\mathcal{M}_1})$ and $X(\mathcal{ST}_{\mathcal{M}_2})$ (a) measurements $ME \in X(\mathcal{ST}_{\mathcal{M}_1})$ are either all harder or all easier than measurements $ME \in X(\mathcal{ST}_{\mathcal{M}_2})$ (strong stability) or (b) the overlap of the ranges of $X(\mathcal{ST}_{\mathcal{M}_1})$ and $X(\mathcal{ST}_{\mathcal{M}_2})$ is less than 10% of the range of $X(\mathcal{ST}_{\mathcal{M}_1}) \sqcap X(\mathcal{ST}_{\mathcal{M}_1})$.

We group ST hardness into the following bins: Very Hard (more than 100 seconds), Hard (>10 sec), Medium Hard (>1 sec), Medium (>100 ms), Medium Easy (>10 ms), Easy (>1 ms), Very Easy (>100 μs.), Trivial (<100 μs). The upper bound of each bin corresponds to the lower bound of the previous one.

5 Experimental Design

We conducted our study on a corpus of 339 OWL API (3.5.0)-parsable BioPortal ontologies, obtained through the BioPortal REST Services[1] (January 2015 snaphshot). All ontologies were serialised into OWL/XML, with merged imports closure. A minimum amount of repair (injecting missing declarations, dropping empty n-ary axioms, etc.) was applied to ensure that trivial violations do not impair DLness.

[1] `http://data.bioontology.org/documentation`

For all our experiments, we use four OWL reasoners that implement the OWL API interface: HermiT 1.3.8, Pellet 2.3.1, JFact 1.2.3 and FaCT++ 1.6.3. All four are among the most heavily used reasoners for OWL 2 DL. The reasoners have been modified for the benchmark: When a subsumption test is conducted, the start and end timestamps, the sub and super class under consideration and the result of the test are recorded. While we can use this approach to compare results for each reasoner, interpretation of comparisons between reasoners might be misleading due to implementational details. For example, methods that test for subsumption and ultimately satisfiability may be nested. See companion website for more information (Section 6). Because we are interested in real life behaviour, we allowed the reasoner to fall into states like the deterministic part of HermiT for Horn-SHIQ or Pellets internal EL-Reasoner. That said, we cannot claim to measure `all` subsumption tests a reasoner does. We can, however, establish a lower bound and are confident that we capture the vast majority of the hard tests, because the sum of test times occasionally account for almost 100% of the OCT for all reasoners.

A set of four equal-spec Mac Minis with Mac OS X Lion 10 (64 bit), 16 GB RAM and 2.7 GHz Intel Core i764-bit was used for the benchmarking. Every single classification was done in a separate isolated virtual machine (Java 7, -Xms2G, -Xmx12G). In order to reduce potential bias induced by run order (unaccounted for background processes kicking in, runtime optimisations), we fully randomise the run order and evenly distribute the experiment run jobs across the four machines.

Experimental Pipeline: For the first experiment we execute a single run of all reasoners across the entire corpus, with a timeout of 60 minutes per run. Due to technical details, the timeout is a lower bound and might not be triggered until some minutes later. Note that we include every ontology in the corpus, including the ones not strictly in OWL DL (53). The reason for that is that these ontologies do form part of the landscape, and reasoners are used on them. The main sources of violations are uses of reserved vocabulary (37% of all violations across the corpus), illegal punning (32%) and uses of datatypes not on the OWL 2 datatype map (11%).

For the second experiment, we select a set of reasoner-ontology pairs for which, according to the results of experiment 1, at least one test was measured that was harder than 100 milliseconds. Because of the various claims we have with respect to modules, we also excluded ontologies that do not fall under OWL 2 DL. *Runtime limitations* forced us to exclude the NCIt from the sample, due to the extreme number of measured subsumption tests (JFact 751,907 tests, Pellet 461,831, FaCT++ 605,481). For this experiment, we first obtain random cumulative subsets from the ontologies in our narrowed down sample, similar to Gonçalves et al. [8], with 16 slices. In a nutshell, given the set of logical axioms the ontologies are comprised of, we obtain a random $\frac{1}{16}$th of the axioms, serialise this subset, add another randomly drawn $\frac{1}{16}$th from the remaining axioms to the first, serialise them together, and then iteratively grow each consecutive subset until the final set is the whole ontology. From the signature of each subset

sampled, we obtain the $\bot$-locality module using the OWL API module extractor. Module properties ensure that, given subset $S1 \subset S2$, $\mathcal{M}_{\widetilde{S1}} \subset \mathcal{M}_{\widetilde{S2}}$. The module of $\frac{16}{16}$th, $\mathcal{M}_{\widetilde{\mathcal{O}}}$, corresponds to the whole ontology. We call this nested set of modules a **path**. Note that the modules are usually considerably larger than their respective subsets, which will give us a good sample of relatively large modules with hopefully hard subsumption tests. Each of the modules obtained is classified three times (i.e., three independent runs) by each reasoner. Given a path $\mathcal{M}_1 \subset \mathcal{M}_2 ... \subset \mathcal{M}_n$, we call $\mathfrak{P}$ the set of all pairs $\mathcal{M}_i, \mathcal{M}_j$ with i<j.

6 Results

Supporting materials, datasets and scripts can be found online[2]. Percentages in this section are subject to appropriate rounding.

6.1 Subsumption Test Landscape

Out of the 1356 attempted classification runs (4 reasoners and 339 ontologies), 1136 (85%) completed successfully. 322 ontologies were dealt with by at least one reasoner (95%) within the 60 minute timeout. Reasons for failure include hitting the timeout, unsupported datatypes (FaCT++), and lack of DLness (mainly HermiT). From the 322 ontologies successfully processed, 186 did not have any subsumption tests measured by any of the three reasoners. By reasoner, FaCT++ did not test in 177 cases, HermiT in 189, JFact in 191 and Pellet did not fire a ST during 218 successful classifications. For the remaining 136, at least one reasoner conducted a ST as described in Section 5. One interesting observation

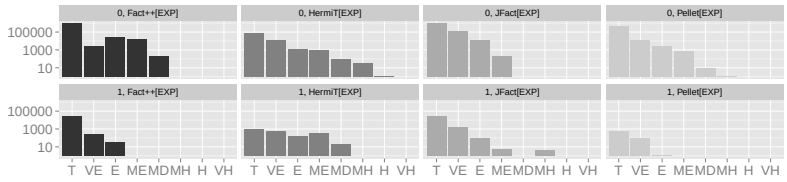


Fig. 1. Counts of ST's for each hardness bin by reasoner (log scale), distinguished by positive (1) and negative (0) tests.

is that most positive tests are of only trivial hardness, while negative tests are generally harder. While subsumption testing dominates the OCT only in a few cases, it occasionally accounts for more than 80%. Very rarely we can observe a single test accounting for more than 10% of the OCT. The maximum impact for a single test by Pellet is 11.3%, HermiT 23.1%, JFact 24.8% and FaCT++ 9.2%. The distribution of subsumption test hardness across all runs according to our hardness scale (Sec. 4) is shown in Figure 1.

[2] `http://bit.ly/1bIqdNX`

6.2 In-depth Characterisation

From the previous experiment, according to the process detailed in Section 5, 3 ontologies were selected for FaCT++, 13 for HermiT, 5 for JFact and 4 for Pellet. The full ontologies have OCT's ranging from 7.31 seconds to 1211.00 seconds (median: 103.20, mean: 210.70). Out of the 1200 (16 modules per ontology, 3 runs per module, 25 ontology reasoner pairs) attempted classifications (timeout 60 minutes), 1093 (91%) successfully terminated. Out of the possible 400 modules (16 modules, 25 ontology-reasoner pairs) across the entire set, we obtain 371 records from the **intra-module analysis**, 358 out of which were obtained from three distinct measurements, 6 are comprised of two distinct measurements and 7 by only one. Since we are interested in observing variability, we discard the latter 7 and stick with 364 partially or fully complete records. Variability is determined using the coefficient of variation (COV). From the module perspective, we look at three distinct sources of variation: overall classification time (OCT), sum of all subsumption test times (SUMST) and the total number of tests conducted (CTT). Across modules, only 3 module OCT varies by more than 30%, 12 by more than 20% and 19 by more than 10%. The module with the worst variation corresponds to a module taken from a $\frac{2}{16}$th of the Biotop ontology, classified by JFact (min=38.49 sec, max=194.22 sec). A more detailed picture of the overall variation can be taken from Figure 2. In terms of test count, the variation is surprisingly large. 246 out of 364 cases (68%) show differences in the number of test measured across runs. Only 118 (32%) do not vary at all. 20 modules vary by more than 10% in the number of subsumption tests.

Across all 371 modules, we measured the hardness of 2,536,339 distinct ST's. Only 89% of the tests are measured more than once and we discard the rest. As can be seen in Figure 2, the coefficient of variation is generally log10-normally distributed (here reported in percent rather than in proportions of 1), but varies considerably across reasoners. On average, measurements deviate as much as 13.22% and 13.96% for Pellet and HermiT respectively, while measurements for and JFact deviate by 3.5%, and FaCT++ only 2.83%. The maximum variation for any test measurement for Pellet is 172.65%, for HermiT 172.50%, for FaCT++ 167.87% and for JFact 169.07%.

For the **inter-module analysis**, we sampled 30 sub-module super-module pairs from $\mathfrak{P}$ from the 120 possible combinations as described in Section 5. Taking into account the successful classification we obtained data from 703 out of 750 possible comparisons. For result stability, we excluded a further 14 pairs that had only a single measurement for either the sub or the super-module, and continued with 689. Figure 3 shows the overall changes in measurement times across pairs by reasoner. Bin membership is determined as follows. Given a pair $<\mathcal{M}_1, \mathcal{M}_2> \in \mathfrak{P}$ we look at the change from either the $CT(\mathcal{M}_1)$ to $CT(\mathcal{M}_2)$ or the change from a subsumption test $\mathcal{ST}_{\mathcal{M}_1}$ to $\mathcal{ST}_{\mathcal{M}_2}$. Every pair of measurements has a tendency, a magnitude and a degree of stability. The tendency *easier* (mean hardness change less than -5%) denotes that a test is easier in the super-module (potentially pathological), *harder* (mean hardness change more than 5%) the reverse, and neutral means the mean measurement difference does

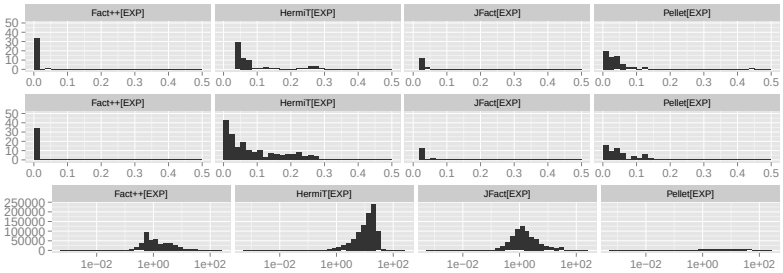


Fig. 2. Top 2 rows: Histogram of variation (COV) by reasoner. Top: OCT, bottom: SUMST. Bottom: Histogram of variation (COV) of ST measurements by reasoner. Mind the log scale.

not change by more than 5%. High magnitudes are changes above 50%, medium magnitudes are changes between 5% and 50% and low changes are below 5%. An effect can be of three degrees of stability: *clear cut*, *high* or *low*, see Section 4. Neutral cases have high stability if both sets of measurements have a variation coefficient less than 5%. From the module perspective, the main observation to be made here is that there are 39 cases in the set where the sub-module is harder than the super-module and 173 where there is no significant change in hardness (less than 5% change). Test time stability varies a lot across reasoners. While FaCT++ measurements are mostly stable, Pellet and JFact measurements vary a lot across almost all potential categories. The pathological cases as described in Section 4, EHC and EHH, occur rarely. Out of the 1,507,654 tests that got easier overall, only 399,644 (26%) are of a high magnitude. Out of those, 376,078 are clear cut, and 8,850 of high stability. From the clear cut cases, 59,656 are potentially unaffected by non-determinism, out of which 204 are harder than 100 ms. Out of the highly stable cases, only 302 are potentially unaffected by non-determinism, out of which only 10 are harder than 100ms. None of the tests in both groups are harder than a second.

7 Discussion

We quantify the impact of subsumption test hardness (RQ1) on classification time in two ways: (1) Contribution of test times measured to OCT and (2) ratio of number of ontologies with tests to those without. Only few of the 136 ontologies with tests were dominated by test hardness: Only 1 ontology had more than a 50% contribution of total SST for Hermit, 7 for Pellet, 19 for FaCT++ and 23 for JFact. However, there are cases where the contribution is very high. The ratio of ontologies entirely without tests is very high: FaCT++: 52%-71%, HermiT 55%-80%, JFact in 56%-76% and Pellet 64%-84%. We have established only the lower bound. The upper bound covers the very unlikely possibility that the failed classifications might be all without tests. Additionally, 36 of the

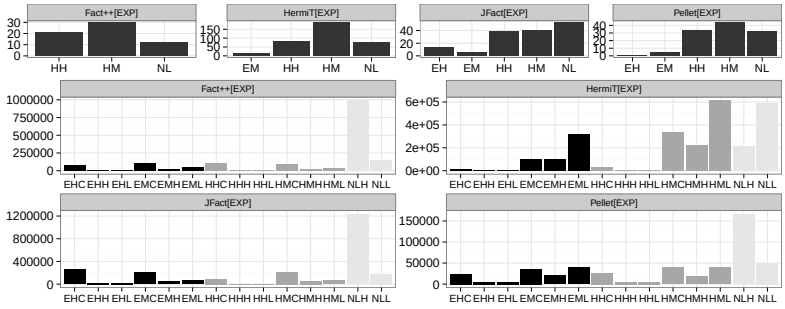


Fig. 3. Hardness classes by reasoner. Top row: OCT, bottom 4: SST. Bin labels x-axis: 1st letter: tendency (easier, neutral, harder), 2nd: magnitude (low, medium, high), 3rd: stability: (clearcut, high, low). Y-axis: number of comparisons.

ontologies entirely without tests have less than 100 TBox axioms (12 less than 10).

RQ2 is quantified by ratio of positive to negative tests. Positive tests account to between 0.12% (Pellet) and 2.61% (FaCT++) of the overall number of tests (JFact 2.49%, HermiT 2.12%). This low ratio is not surprising, given that the worst case N^2 is dominated by far by non-subsumptions. As a side observation, current traversal algorithms appear highly efficient. Only 3 ontology-reasoner pairs (two distinct ontologies, small TBoxes) trigger more than 10% of the worst case N^2 number of subsumption tests, and 50 pairs (30 unique ontologies) trigger more than 1% of the worst case. This result however is only indicative of the efficiency, as we do not guarantee to measure all tests.

The distribution of test hardness as shown in Figures 1 tends towards easy tests (RQ3). Figure 1 show that the number of really hard tests are in the minority: only 346 out of 2,671,896 tests measured overall are harder than a second. This result may emphasise the importance of test avoidance over further optimising individual subsumption tests. However, as there are individual tests that can make up to 25% of the overall reasoning time, it cannot be disregarded.

The *variation of the measurements*, both for individual tests and overall times, is, at least in its magnitude, surprising (RQ4). While the variation of test times could be so high merely due to the low number of measurements that are very vulnerable to experimental error (for example an unaccounted for system background kicking in, stochasticity in the garbage collection, room temperature), we cannot claim the same for the variation in the numbers of triggered tests. That 68% of the modules in the sample vary in the number of tests is a very strong indicator for the stochasticity of the classification process (at least in this particular sample), be it due to random effects in the programming language or deliberate randomness induced by the implementation. This poses a serious threat for single-run benchmarking, as it is still general practice in the DL community. A small indication of the potential impact of a particular programming environment is the very low average variation in test times collected

for FaCT++, which is the only reasoner in the set implemented in C++. In the *inter-module comparison* we learned that our pathological cases rarely happen and if so, the effect they might have on overall classification time is negligible, due to the potential degree of the effect and the rarity in which they occur. Furthermore, the strong evidence of stochasticity of the classification process makes it unclear whether the effect might not simply be due to non-determinism. Despite having detected some cases that are clearly counter-intuitive (in the sense of getting easier when irrelevant stuff is added in), we cannot be sure whether modularity is the cause, due of the small effect size (RQ6). On top of that, easier and harder tests almost balance each other out. Given our sample bias, our results are not conclusive.

8 Conclusions and Future Work

In this paper we have presented a procedure for reliable and reproducible isolation of counter-intuitive reasoning behaviour on subsumption tests during classification and presented some such isolated cases. Future work includes completing the full characterisation of the corpus with respect to the pathological cases and then investigating the causal basis of those cases. The most likely explanation is that the additional axioms trigger a cheaper choice in the complex non-determinism algorithms. The big challenge is whether any progress can be made in a fairly reasoner independent way. One idea is to extract the justifications for a given entailment and see whether they are disproportionally difficult individually. This would suggest that the additional information is directing the algorithm toward "easier" reasoners.

References

1. F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. *Appl. Intell.*, 4(2):109–132, 1994.
2. K. Dentler, R. Cornet, A. t. Teije, and N. d. Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web*, 2(2):71–87, 2011.
3. B. Glimm, I. Horrocks, and B. Motik. Optimized Description Logic Reasoning via Core Blocking. In *IJCAR 2010*, pages 457–471, 2010.
4. B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos. A novel approach to ontology classification. *J. Web Sem.*, 14:84–101, 2012.
5. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 Reasoner. *J. Autom. Reasoning*, 53(3):245–269, 2014.
6. R. S. Gonçalves, S. Bail, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, B. Glimm, and Y. Kazakov. ORE Reasoner Evaluation (ORE) Workshop 2013 Results: Short Report. In *ORE 2013*, pages 1–18, 2013.
7. R. S. Gonçalves, N. Matentzoglu, B. Parsia, and U. Sattler. The Empirical Robustness of Description Logic Classification. In *ISWC 2013*, pages 277–280, 2013.
8. R. S. Gonçalves, B. Parsia, and U. Sattler. Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies. In *ISWC 2012*, pages 82–98, 2012.

9. B. C. Grau, C. Halaschek-Wiener, and Y. Kazakov. History Matters: Incremental Ontology Reasoning Using Modules. In *ISWC 2007*, pages 183–196, 2007.
10. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular Reuse of Ontologies: Theory and Practice. *J. Artif. Intell. Res. (JAIR)*, 31:273–318, 2008.
11. I. R. Horrocks. *Optimising tableaux decision procedures for description logics*. PhD thesis, Citeseer, 1997.
12. Y. Kazakov. RIQ and SROIQ Are Harder than SHOIQ. In *KR 2008*, pages 274–284, 2008.
13. N. Matentzoglu, B. Parsia, and U. Sattler. An Empirical Investigation of Difficulty of Subsets of Description Logic Ontologies. In *DL 2014.*, pages 659–670, 2014.
14. N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. D. Storey, C. G. Chute, and M. A. Musen. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(Web-Server-Issue):170–173, 2009.
15. A. A. Romero, B. C. Grau, and I. Horrocks. MORe: Modular Combination of OWL Reasoners for Ontology Classification. In *ISWC 2012*, pages 1–16, 2012.
16. V. Sazonau, U. Sattler, and G. Brown. Predicting Performance of OWL Reasoners: Locally or Globally? In *KR 2014*, 2014.
17. R. Shearer and I. Horrocks. Exploiting Partial Information in Taxonomy Construction. In *ISWC 2009*, pages 569–584, 2009.
18. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.
19. A. Steigmiller, T. Liebig, and B. Glimm. Konclude: System description. *J. Web Sem.*, 27:78–85, 2014.
20. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *IJCAR 2006*, pages 292–297, 2006.
21. D. Tsarkov and I. Palmisano. Chainsaw: a Metareasoner for Large Ontologies. In *ORE 2012*, 2012.

Heuristics for Applying Cached OBDA Rewritings

Andreas Nakkerud and Evgenij Thorstensen

Department of Informatics, University of Oslo

Abstract. In OBDA systems, cached rewritings can be used to significantly shorten the process of finding rewritings of bigger queries. The main challenge in using cached rewritings is finding the optimal combination to apply, a problem which is NP-complete. In this paper, we present a way to calculate the value of a cached rewriting, and propose using this value to decide which cached rewritings to apply. The idea behind our technique is to estimate how much computation, in particular optimization, has gone into the rewriting. This can tell us something about which cached rewritings we should prioritise, and also when to cache a rewriting. In order to quantify optimization, we define a measure of UCQs, and calculate this measure at different stages of the rewriting.

1 Introduction

Ontology-based data access (OBDA) [11] is a recent paradigm for accessing *data sources* through an *ontology* that acts as a conceptual, integrated view of the data. The data sources are connected to the ontology via *mappings* that specify how to retrieve the appropriate data from the sources. The framework of OBDA has received a lot of attention in the last years: many theoretical studies have paved the way for the construction of OBDA systems (e.g., [3, 5, 13]) and the development of OBDA projects for enterprise data management in various domains [2].

The usual way of answering a query over the ontology in the OBDA framework [11] is to transform it into a set of queries to be executed over the data sources. This process is computationally expensive, as it requires logical reasoning over knowledge represented in the ontology (rewriting) followed by the application of mappings to the ontological query (unfolding). Furthermore, as the resulting set of queries is likely to contain redundancies, optimization techniques are typically applied along the way [10]. It therefore makes sense to *cache* the results of a query rewriting and unfolding, in the hope of using them to speed up the answering of future queries [10]. However, the problem of finding the cached rewritings to use is itself hard, as it is a variant of the well-known problem of query answering using views [8]. In particular, checking whether a cached rewriting can be applied to a query is in general NP-complete.

In this paper, we therefore study the following problem: Given a query Q to answer and a set of cached rewritings of previous queries, how useful is each

cached rewriting likely to be, if it is applicable to Q? Knowing this, the system can make sensible decisions as to which cached rewritings to try and apply, and in what order.

As a measure of the usefulness of an existing cached rewriting, previous work by Di Pinto et al. [10] use the number of atoms in the queries. This approach does not take into account the details of the rewriting algorithm used, nor the specification of the OBDA system the queries are answered over. In order to get a finer picture of the usefulness of a cached rewriting, we define a measure for UCQs, and apply it to an intermediate stage of the rewriting. We then use this measure, as well as measures of the cached rewriting, to define a heuristic. In addition to analysing the queries of a cached rewriting directly, this heuristic also takes into account how ontology queries are rewritten by the OBDA system.

2 Preliminaries

In this section, we define basic notions related to OBDA systems and their components: databases, ontologies, and mappings. We then give a definition of query rewriting and unfolding, followed by a formal definition of cached query rewritings as mappings that possess specific properties.

2.1 Databases

In this paper, we assume a fixed relational schema $\mathcal{S}$, and adopt the standard notions for conjunctive queries (CQs) over $\mathcal{S}$ [1]. To every conjunctive query CQ we associate a measure of its size, denoted by $\mathsf{size}(CQ)$. There are several ways to measure the size of a conjunctive query; we will discuss using the number of atoms or the number of variables in the query. However, it is possible to also use more advanced measures of query size, such as tree and hypertree width [6, 7].

Given a conjunctive query Q and a tuple of constants $\boldsymbol{t}$, we write $Q[\boldsymbol{t}]$ for the query obtained by replacing the free variables of Q with the constants in $\boldsymbol{t}$. Given a database instance $\mathcal{D}$ and a query Q, we write $\mathsf{ans}(Q, \mathcal{D})$ for the set of answers to Q over $\mathcal{D}$, defined in the standard way.

2.2 Ontologies

An ontology is a description of a domain of interest in some formal language. Here, we consider the languages of Description Logics (DLs). In general, an ontology expressed in a description logic is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, where the TBox $\mathcal{T}$ contains axioms specifying universal properties of the concepts and roles in the domain, while the ABox $\mathcal{A}$ specifies instances of concepts and roles. In the OBDA setting, the ABox is given by mappings rather than explicitly, and so the TBox is the only relevant component. We therefore set $\mathcal{O} = \mathcal{T}$.

In the examples discussed in this paper, we will use the description logic *DL-Lite*$_A$ [4, 12]. The syntax of *DL-Lite*$_A$ is based on concepts, value-domains,

roles, and attributes, and can be defined using the following grammar [10]:

$$\begin{array}{ll} B \to A \mid \exists Q \mid \delta(U_C) & E \to \rho(U) \\ C \to B \mid \neg B & F \to T_1 \mid \ldots \mid T_n \\ Q \to P \mid P^- & V \to U \mid \neg U \\ R \to Q \mid \neg Q, & \end{array}$$

where A is a *concept name*, P is a *role name*, U is an *attribute name*, and $T_1, \ldots, T_n$ are *value-domains*. We let Σ_O be a set of ontology predicates, and Γ_C a set of constants. The semantics of *DL-Lite*$_A$ is defined in terms of first-order interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over $\Sigma_O \cup \Gamma_C$. The non-empty domain $\Delta^{\mathcal{I}}$ is the union of the disjoint sets Δ_V and $\Delta_O^{\mathcal{I}}$, where Δ_V is the domain for interpreting data values, and $\Delta_O^{\mathcal{I}}$ is the domain for interpreting object constants. The interpretation function $\cdot^{\mathcal{I}}$ is defined as follows:

$$\begin{array}{ll} A^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} & (\neg U)^{\mathcal{I}} = (\Delta_O^{\mathcal{I}} \times \Delta_V) \setminus U^{\mathcal{I}} \\ P^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}} & (P^-)^{\mathcal{I}} = \{(o, o') \mid \exists v.[(o', o) \in P^{\mathcal{I}}]\} \\ U^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V & (\exists Q)^{\mathcal{I}} = \{o \mid \exists o'.[(o, o') \in Q^{\mathcal{I}}]\} \\ (\neg B)^{\mathcal{I}} = \Delta_O^{\mathcal{I}} \setminus B^{\mathcal{I}} & (\delta(U))^{\mathcal{I}} = \{o \mid \exists v.[(o, v) \in U^{\mathcal{I}}]\} \\ (\neg Q)^{\mathcal{I}} = (\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}} & (\rho(U))^{\mathcal{I}} = \{v \mid \exists o.[(o, v) \in U^{\mathcal{I}}]\}. \end{array}$$

A *DL-Lite*$_A$ TBox $\mathcal{T}$ is a finite set of axioms of the form

$$B \sqsubseteq C \qquad Q \sqsubseteq R \qquad E \sqsubseteq F \qquad U \sqsubseteq V \qquad (\mathsf{funct}\ Q) \qquad (\mathsf{funct}\ U).$$

The interpretation $\mathcal{I}$ satisfies an axiom $X \sqsubseteq Y$ if $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$. $\mathcal{I}$ satisfies ($\mathsf{funct}\ Z$) if for every $o, o', o'' \in \Delta_O^{\mathcal{I}}$, if $(o, o') \in Z^{\mathcal{I}}$ and $(o, o'') \in Z^{\mathcal{I}}$, then $o' = o''$.

2.3 Mappings

In the general case, a mapping assertion m between a database schema $\mathcal{S}$ and an ontology $\mathcal{O}$ has the form $Q \rightsquigarrow q$, where the body Q is a CQ over $\mathcal{S}$, and the head q a CQ over the vocabulary of $\mathcal{O}$ [11], possibly with shared free variables. To define the semantics of mapping assertions, we first need to define the notion of an OBDA system.

2.4 OBDA systems

An OBDA system specification is a triple $\mathcal{B} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ where $\mathcal{S}$ is a database schema, $\mathcal{M}$ a set of mapping assertions, and $\mathcal{O}$ an ontology. The semantics of query answering in OBDA systems are usually defined using first-order interpretations.

Definition 1 (OBDA semantics). *Let $\mathcal{B} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA system specification, and $\mathcal{D}$ a database for $\mathcal{S}$. A first order interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{D}$ is a model for $\mathcal{B}$ if*

– $\mathcal{I} \models \mathcal{O}$, *and*
– *for every tuple of constants* $\boldsymbol{t}$ *from* $\mathcal{D}$*, and every mapping assertion* $Q \rightsquigarrow q \in \mathcal{M}$*, we have that* $\mathcal{I} \models q[\boldsymbol{t}]$ *whenever* $\mathcal{I} \models Q[\boldsymbol{t}]$.

The set of answers to a query q *over* $\mathcal{B}$ *and* $\mathcal{D}$ *is the set of tuples* $\boldsymbol{t}$ *from* $\mathcal{D}$ *such that* $q[\boldsymbol{t}]$ *is true in every model of* $\mathcal{B}$ *and* $\mathcal{D}$*. We write* $\mathsf{ans}(q, \mathcal{B}, \mathcal{D})$ *for the answers to* q *over* $\mathcal{B}$ *and* $\mathcal{D}$.

As mentioned in the introduction, answering a query in an OBDA system is usually done by rewriting and unfolding the query into the correct database queries to execute.

Definition 2 (Rewriting and unfolding). *Let* $\mathcal{B} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ *be an OBDA system specification, and* q *a CQ over the vocabulary of* $\mathcal{O}$*. A* rewriting *of* q *under* $\mathcal{B}$ *is a query* q' *over the same vocabulary such that* $\mathsf{ans}(q', \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle, \mathcal{D}) = \mathsf{ans}(q, \mathcal{B}, \mathcal{D})$ *for every database instance* $\mathcal{D}$.

An unfolding *of the rewriting* q' *of* q *is a query* Q *over* $\mathcal{S}$ *such that* $\mathsf{ans}(Q, \mathcal{D}) = \mathsf{ans}(q, \mathcal{B}, \mathcal{D})$ *for every database instance* $\mathcal{D}$.

We can now define the notion of a perfect mapping assertion, which captures the idea of caching the rewriting and unfolding of a query.

Definition 3 (Perfect mapping assertion [10]). *Let* $\mathcal{B} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ *be an OBDA system specification. A mapping assertion* $Q \rightsquigarrow q$ *is a* perfect mapping assertion *if for every database instance* $\mathcal{D}$*, we have* $\mathsf{ans}(q, \mathcal{B}, \mathcal{D}) = \mathsf{ans}(Q, \mathcal{D})$.

Each cached rewriting and unfolding is a perfect mapping assertion. In fact, caching is one of the primary methods for obtaining perfect mapping assertions [10].

3 Applying Perfect Mapping Assertions

As part of their full OBDA rewriting algorithm, Di Pinto et al. [10] present the algorithm *ReplaceSubqueryR* for applying a perfect mapping assertion before the regular rewriting procedure starts. Using the notion of *restricted homomorphisms,* they give an exact definition of when a perfect mapping assertion can be applied.

Given a conjunctive query q, *ReplaceSubqueryR* goes through the perfect mapping assertions in the order specified by some heuristic, modifying q whenever the perfect mapping assertion is applicable. The order of application is significant, since the application of one perfect mapping assertion can prohibit the subsequent application of another. Finding the optimal combination of perfect mapping assertions to apply is NP-hard [10]. Di Pinto et al. use a greedy strategy. The heuristic for this strategy is the number of atoms in the heads of the perfect mapping assertions.

Our goal is to define an improved heuristic for the order of application of perfect mapping assertions.

The following example, based on one of the challenges faced by the Optique project[1], shows how we create perfect mapping assertions.

Example 4. Let $company(name, owner, manager, accountant)$ be a database table. We define the concepts $Company$ and $Owner$, $Manager$ and $Accountant$, and the roles $hasOwner$, $hasManager$, and $hasAccountant$. We define the TBox,

$$\mathcal{T} = \left\{ \begin{array}{r} \exists hasOwner \sqsubseteq Company, \\ \exists hasManager \sqsubseteq Company, \\ \exists hasAccountant \sqsubseteq Company \end{array} \right\},$$

and mapping assertions

$$\mathcal{M} = \left\{ \begin{array}{l} \exists y, z, w.company(x, y, z, w) \rightsquigarrow Company(x) \\ \exists z, w.company(x, y, z, w) \rightsquigarrow hasOwner(x, y) \\ \exists y, w.company(x, y, z, w) \rightsquigarrow hasManager(x, z) \\ \exists y, z.company(x, y, z, w) \rightsquigarrow hasAccountant(x, w) \end{array} \right\}.$$

We now look at the query $q(x) = Company(x)$. The ontology rewriting of $Company(x)$ is

$$\begin{aligned} q'(x) = Company(x) &\vee \exists v.hasOwner(x, y) \\ &\vee \exists v.hasManager(x, y) \vee \exists v.hasAccountant(x, y). \end{aligned}$$

When unfolding $q'(x)$ we get the UCQ

$$\begin{aligned} &\exists y, z, w.company(x, y, z, w) \\ &\quad \vee \exists v, z, w.company(x, v, z, w) \\ &\quad \vee \exists y, v, w.company(x, y, v, w) \\ &\quad \vee \exists y, z, v.company(x, y, z, v), \end{aligned}$$

which is obviously equivalent to

$$\exists y, z, w.company(x, y, z, w).$$

We cache this rewriting by saving the perfect mapping assertion

$$\exists y, z, w.company(x, y, z, w) \rightsquigarrow Company(x).$$

Example 4 illustrates one situation where perfect mapping assertions are useful. An alternative way of dealing with his particular example is by modifying the set $\mathcal{M}$ of mapping assertions according to the TBox [13], and then to optimise it [9, 13]. This approach is not as general as the perfect mapping assertion approach, because it only works when there are redundancies in the mapping assertions. The perfect mapping assertions can represent optimisations that are only valid in special cases.

[1] http://optique-project.eu/

4 Query Measure

In order to evaluate the quality of a perfect mapping assertion, we need some way of measuring a UCQ. We will use this measure to quantify the amount of optimization that has gone into creating a perfect mapping assertion.

There are several ways to measure the size of a conjunctive query. For ease of exposition, we will use the number of atoms in the query in our examples, although the number of variables is usually more relevant to the exact cost of optimising a query, since query optimisation amounts to finding homomorphisms between queries. If the number of variables is approximately linear in the number of atoms, the results of using either will be similar.

Furthermore, the number of atoms in each conjunction of the unfolding of a conjunctive query can be approximated from the number of mapping assertions that mentions conjunct. On the other hand, finding the number of variables requires an analysis of each mapping assertion.

The size of a UCQ is harder to describe with a single number, because UCQs have two dimensions: the conjunctions and the conjuncts in them.

Definition 5 (Measure of UCQs). *Let $Q = CQ_1 \vee \ldots \vee CQ_k$ be a UCQ, $\mathsf{size}(CQ_i)$ the size (e.g. number of atoms) of CQ_i, and f a weighting function. Define $g(x) = x$ if f is at most linear, and $g = f^{-1}$ otherwise. The* measure of *Q is*

$$S_Q = g\left(\sum_{i=1}^{k} f(\mathsf{size}(CQ_i))\right).$$

The function f defines how S_Q depends on the size of the conjunctive queries, the function g makes S_Q at most linear in the sum of the sizes. Our choice of g will make S_Q of the order $O(k \cdot \max_i[\mathsf{size}(CQ_i)])$, where, and k is the number of conjunctive queries in the UCQ. We choose f according to the following observations.

- f **constant:** We disregard the size of disjuncts. The cost of performing optimizations is assumed to be insignificant compared to the cost of rewriting, unfolding and storing the query.
- f **linear:** The cost of optimization is assumed to be on the same order as the cost of rewriting, unfolding and storing the query.
- f **polynomial or exponential:** The cost of optimization is assumed to be more important than the cost of rewriting, unfolding and storing the query.

The above list is explains how different definitions of f affects the measure S_Q. It gives a strong indication of how we should define f, depending on what we want to use the measure S_Q for. In Sections 5 and 6, we will use linear f. We get back to the choice of f briefly in Section 7.

5 Maximal expansion

Having defined a measure for UCQs, we are now in a position to assign a value to the body Q of a perfect mapping assertion $Q \rightsquigarrow q$. Such an analysis gives us some information about how much we stand to gain by applying this perfect mapping assertion. We can, however, get an even better picture by looking at the process of rewriting q. Doing this, we can get a measure of how complex q really is in the relevant OBDA system.

Definition 6 (Maximal expansion). *Let q be a query, $\mathcal{B} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ an OBDA system specification, and $\mathfrak{R}$ an ontology rewriting algorithm. The* maximal expansion of q over $\mathcal{B}$ and $\mathfrak{R}$, *denoted* $\mathsf{me}(q, \mathcal{B}, \mathfrak{R})$, *is the unoptimised ontology rewriting and unfolding of q over $\mathcal{B}$ using $\mathfrak{R}$.*

We write $\mathsf{me}(q, \mathcal{B})$ when the choice of $\mathfrak{R}$ is clear from the context. When the OBDA system is also understood from the context, we let $S_{\mathsf{me}(q)}$ denote the measure of $\mathsf{me}(q, \mathcal{B})$. The following example shows how we calculate $S_{\mathsf{me}(q)}$.

Example 7. We look at an OBDA system specification with TBox axioms $A \sqsubseteq \exists R$ and $S \sqsubseteq T$, and the mapping

$$\mathcal{M} = \left\{ \begin{array}{l} Q_{A_1} \rightsquigarrow A, Q_{A_2} \rightsquigarrow A, Q_{A_3} \rightsquigarrow A, \\ Q_{R_1} \rightsquigarrow R, Q_{R_2} \rightsquigarrow R, Q_{R_3} \rightsquigarrow R, Q_{R_4} \rightsquigarrow R, \\ Q_{S_1} \rightsquigarrow S, Q_{S_2} \rightsquigarrow S, \\ Q_{T_1} \rightsquigarrow T, Q_{T_2} \rightsquigarrow T, Q_{T_3} \rightsquigarrow T \end{array} \right\}$$

and rewrite the query $q(x, y) = \exists z.R(x, z) \wedge T(x, y)$. The ontology rewriting of q according to rewriting algorithm $\mathfrak{R}$ is

$$\begin{aligned} q'(x, y) = [\exists z.R(x, z) \wedge T(x, y)] \vee [\exists z.R(x, z) \wedge S(x, y)] \\ \vee [A(x) \wedge T(x, y)] \vee [A(x) \wedge S(x, y)] \end{aligned}$$

We choose $f(x) = g(x) = x$, let $\mathsf{size}(CQ)$ be the number of atoms in CQ, and calculate $S_{\mathsf{me}(q)}$. If each Q_{X_n} is a query without joins, then the resulting maximal expansion is a UCQ with joins of size 2. The first disjunct in q' is unfolded into 12 conjunctive queries of size 2, since there are four queries mapped to R and 3 to T. We do similar calculations with each disjunct of q', and end up with a total of 35 conjunctive queries, each of size 2. Therefore $S_{\mathsf{me}(q)} = 70$.

In Example 7, we assumed that there were no conflicts between mapping assertions during the unfolding. Such conflicts can arise when there the mapping assertions contain constants in place of some variables. In this case, only those mapping assertions that have constants replacing the same variables can create conflict, and this will usually only be the case for a very few combinations of assertions. If constants are common in the used mapping assertions, then greater care must be taken when using approximations of $S_{\mathsf{me}(q)}$.

6 Heuristic

With a measure for UCQs and the notion of maximal expansions, we are now in a position to expand on the heuristic suggested by Di Pinto et al. [10]. We design a heuristic where large values are better. In the following we assume a fixed OBDA system and ontology rewriting algorithm.

For a perfect mapping assertion $Q \rightsquigarrow q$, we calculate the measures S_q, S_Q and $S_{\mathsf{me}(q)}$. Note that $S_q = \mathsf{size}(q)$, since q is a conjunctive query.During both ontology rewriting and unfolding, the size of a conjunctive query can grow exponentially. In both cases there is generally multiple options for dealing with every conjunct, and the result is a very large UCQ. For this reason, we will use $\log S_Q$ and $\log S_{\mathsf{me}(q)}$, since they will be approximately proportional to S_q. We also calculate the ratio $\log S_{\mathsf{me}(q)} / \log S_Q$, which tells us how much optimisation has gone into the creation of the perfect mapping assertion. We use this optimization ratio to adjust other measures of the value of the perfect mapping assertion.

If S_q is large for a perfect mapping assertion $Q \rightsquigarrow q$, that is, the size of q is large, then applying it cuts a large portion of the original query. There is, however, a risk that this portion could be rewritten directly without much difficulty. In order to compensate for this effect, we scale S_q by the optimisation ratio $\log S_{\mathsf{me}(q)} / \log S_Q$.

A perfect mapping assertion is also valuable if the corresponding $S_{\mathsf{me}(q)}$ is large, no matter the size of S_q. Again, we scale $\log S_{\mathsf{me}(q)}$ by the optimisation ratio $\log S_{\mathsf{me}(q)} / \log S_Q$.

Assigning a weight parameter to each of these compound measures, we get the heuristic

$$a\frac{(\log S_{\mathsf{me}(q)})^2}{\log S_Q} + b\frac{S_q \cdot \log S_{\mathsf{me}(q)}}{\log S_Q} + cS_q$$

We assume S_q, $\log S_Q$ and $\log S_{\mathsf{me}(q)}$ have the same units, since they are all approximately linear in the size of the head q of the perfect mapping assertion $Q \rightsquigarrow q$. Then, the unit of each term in the above sum is the same, and also approximately linear in the $\mathsf{size}(q)$. This means that the ratio between the terms will be fairly stable in respect to typical query size, and as such, the tuning of the parameters a, b, and c will not be very sensitive to the typical query size.

Example 8. We look at an ontology with an empty TBox, and the four roles R_1, R_2, R_3, and R_4. We define the set of mapping assertions

$$\mathcal{M} = \{Q_i^j \rightsquigarrow R_i \mid j = 1, \ldots, i^2\},$$

where $i = 1, \ldots, 4$. We let $\mathsf{size}(CQ)$ be the number of atoms in CQ, and assume that each Q_i^j is atomic. We let $f(x) = x$, and calculate S_q and $S_{\mathsf{me}(q)}$ for different conjunctions of R_i, with each role occurring at most once in each query. The results are shown in Table 1. We see that $S_{\mathsf{me}(q)}$ gives us a much finer division than S_q, which only divides the queries into 3 groups. Given the query

$$R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_4) \wedge R_4(x_4, x_5),$$

and the perfect mapping assertions

$$Q_A(y_1, y_2, y_3, y_4) \rightsquigarrow R_1(y_1, y_2) \wedge R_2(y_2, y_3) \wedge R_3(y_3, y_4)$$
$$Q_B(z_1, z_2, z_3) \rightsquigarrow R_3(z_1, z_2) \wedge R_4(z_3, z_4),$$

there is a good chance that our best option is to apply the second, shorter perfect mapping assertion, because the maximal expansion of $R_3(x_3, x_4) \wedge R_4(x_4, x_5)$ is so large. In order to be more precise, we would need to obtain the measures S_{Q_A} and S_{Q_B}.

q	R_1	R_2	R_3	R_4	R_1 R_2	R_1 R_3	R_1 R_4	R_2 R_3	R_2 R_4	R_3 R_4	R_1 R_2 R_3	R_1 R_2 R_4	R_1 R_3 R_4	R_2 R_3 R_4
S_q	1	1	1	1	2	2	2	2	2	2	3	3	3	3
$S_{\mathsf{me}(q)}$	1	4	9	16	8	18	32	72	128	288	108	192	432	1728

Table 1. The measure $S_{\mathsf{me}(q)}$ can be used to fine order the groups defined by S_q, but there are also pairs of queries where S_q and $S_{\mathsf{me}(q)}$ disagree on ordering.

7 Discussion and Conclusion

Di Pinto et al. [10] have found that using cached rewritings can significantly reduce the cost of answering queries. Their well performing algorithm, *ReplaceSubqueryR*, relies on a heuristic for determining the order in which to apply cached rewritings. Di Pinto et al. have chosen a simple heuristic based on the sizes of the heads of the cached rewritings. This is a good choice if all subqueries have approximately the same complexity when seen together with the TBox and the mapping. If some ontology predicates are easier to rewrite and unfold than others, then the measures of the maximal expansion $S_{\mathsf{me}(q)}$ and the optimised rewriting S_Q become relevant to the choice of cached rewriting.

We suggest the heuristic

$$a\frac{(\log S_{\mathsf{me}(q)})^2}{\log S_Q} + b\frac{S_q \cdot \log S_{\mathsf{me}(q)}}{\log S_Q} + cS_q$$

If we let $a = b = 0$, $c \neq 0$, and define $\mathsf{size}(CQ)$ to be the number of atoms in CQ, then our heuristic becomes the same as the one used in [10]. By tweaking f in Definition 5, and the parameters a, b and c, we can shift importance away from S_q, and towards maximal expansion $S_{\mathsf{me}(q)}$ and optimisation S_Q. The ideal setup will depend on the OBDA system specification, and should be decided experimentally. For the heuristic presented here to be more accurate than the one in [10], the OBDA system specification needs to be uneven in terms of how

each ontology predicate is rewritten. If the mapping has very many assertions for some ontology predicates and few for others, or if some ontology predicates occur often in the TBox while others don't, then the count of predicates in a query need not reflect how it behaves during rewriting. Also, if the mapping is very large, the optimisation ratios are more likely to be significant, since the unfolding and subsequent optimisation will be a large part of query rewriting.

Since S_q, $S_{\mathsf{me}(q)}$, and S_Q are relatively cheap to calculate or approximate during rewriting, and cheap to store in a cache, we claim our suggested heuristic, in many settings, will outperform the simpler heuristic provided by [10]. Even when the simple heuristic performs very well, we can let $a = 0$ and $b \ll c$, so that $S_{\mathsf{me}(q)}$ is used for a fine splitting as illustrated by Example 8.

8 Future Work

We plan to continue this work by experimentally verifying our results. In particular, we would like to compare the different weighting and scaling functions discussed here on real-world datasets. Another line of enquiry would be to see how well these heuristics can substitute for e.g. techniques to eliminate mapping redundancy, as discussed in Section 3.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. Natalia Antonioli, Francesco Castanò, Cristina Civili, Spartaco Coletta, Stefano Grossi, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Domenico Fabio Savo, and Emanuela Virardi. Ontology-based data access: the experience at the Italian Department of Treasury. volume 1017, pages 9–16, 2013.
3. D. Calvanese, M. Giese, P. Haase, I. Horrocks, T. Hubauer, Y. Ioannidis, E. Jiménez-Ruiz, E. Kharlamov, H. Kllapi, J. Klüwer, M. Koubarakis, S. Lamparter, R. Möller, C. Neuenstadt, T. Nordtveit, Ö. Özcep, M. Rodriguez-Muro, M. Roshchin, F. Savo, M. Schmidt, A. Soylu, A. Waaler, and D. Zheleznyakov. Optique: OBDA solution for big data. In *Revised Selected Papers of ESWC 2013 Satellite Events*, volume 7955 of *LNCS*, pages 293–295. Springer, 2013.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
5. Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo. MASTRO STUDIO: Managing ontology-based data access applications. *PVLDB*, 6:1314–1317, 2013.
6. Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

7. Georg Gottlob, Reinhard Pichler, and Fang Wei. Tractable database design through bounded treewidth. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'06)*, pages 124–133. ACM, 2006.
8. Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.
9. Domenico Lembo, José Mora, Riccardo Rosati, Domenico Fabio Savo, and Evgenij Thorstensen. Towards mapping analysis in ontology-based data access. In Roman Kontchakov and Marie-Laure Mugnier, editors, *Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*, volume 8741 of *Lecture Notes in Computer Science*, pages 108–123. Springer, 2014.
10. Floriana Di Pinto, Domenico Lembo, Maurizio Lenzerini, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Optimizing query rewriting in ontology-based data access. In Giovanna Guerrini and Norman W. Paton, editors, *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 561–572. ACM, 2013.
11. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. In Stefano Spaccapietra, editor, *Journal on Data Semantics X*, volume 4900 of *Lecture Notes in Computer Science*, pages 133–173. Springer, 2008.
12. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.
13. Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 558–573. Springer, 2013.

Managing QoS Acceptability for Service Selection : A Probabilistic Description Logics Based Approach

Salima Benbernou[1], Allel Hadjali [2], Naouel Karam[3], Mourad Ouziri[1]

[1]Université Paris Descartes, Sorbonnes Paris Cité, France
{mourad.ouziri,salima.benbernou}@parisdescartes.fr
[2] LIAS/ENSMA, Poitiers, France
allel.hadjali@ensma.fr ,
[3]Department of Mathematics and Computer Science, Freie Universität Berlin, Germany
naouel.karam@fu-berlin.de

Abstract. Quality of Service (QoS) guarantees are usually regulated in a Service Level Agreement (SLA) between provider and consumer of services. Such guarantees are often violated, it may however be the case where the available services do not match exactly all required QoS, leading the system to grind to a halt. It would be better to look for an approximation for acceptable QoS and avoid the complete stopping of the running services. This paper aims at making dynamic QoS acceptability easy for service selection. The proposed model is based on an extension of existing probabilistic description logics reacting to QoS variations. The contributions made are twofold (1) a query description language to express the required QoS by means of a probabilistic description logic (2) a reasoning algorithm for decision making about the acceptability of QoS w.r.t the probabilistic description.

1 Introduction

In a Service Oriented Architecture, service providers need to characterize their services defining both the offered functionalities and the offered quality. Then, the quality of service (QoS) properties can be critical elements for achieving the business goals of service providers. Establishing QoS contracts, described in a SLA, that can be monitored at runtime, is therefore of paramount importance. The SLA must stipulate the values of QoS attributes that a service provider is expected to deliver to a client.

Moreover, a successful execution of a service composition implies continuous monitoring of QoS at runtime. However, at this step, variations could occur in the QoS and most likely induce violations of the agreement. For that, the service composition should support a dynamic QoS-driven adaptation. A static adaptation is not adequate due to variations of the QoS. Existing approaches deal with static QoS, they do not address the attribute variations issue. A *formal and declarative approach* to achieve a dynamic adaptation is then highly desired.

In this paper, we propose a probabilistic description logic based approach to describe the QoS attributes and handle their variations through the use of linguistic concepts. The proposed approach helps to select services in order to build approximate compositions which may not satisfy all the requirements, hence avoiding to completely stop running the processes. It provides the basis allowing the representation and reasoning about the introduced symbolic concepts, where the variation of QoS is modeled in a probabilistic way, hence managing the QoS acceptability. Our main contributions are summarized in the following:

1. We introduce a query description language to express the required QoS using both linguistic concepts and probabilistic representations. To this end, we extend existing probabilistic description logics to handle QoS variations.
2. We develop a suitable reasoning algorithm for computing the QoS acceptability of the selected service w.r.t the probabilistic description. The algorithm achieves an approximate reasoning using inference rules involving probabilistic statements.

The remainder of the paper is structured as follows. In section 2, we provide first, a discussion about existing works dealing with QoS and, second a review of the approaches related to probabilistic description logics. Section 3 describes our proposal to manage the QoS acceptability. In section 4, we present the syntax and semantics of the proposed language. Section 5 details the reasoning algorithm and section 6 concludes the paper.

2 Related Work

Most of existing approaches focused on the contract definition and on mechanisms for contract enactment. [1] describes a matchmaking algorithm for ranking functionally equivalent services. In [8], a fuzzy service adaptation approach that leverages the degrees of QoS satisfaction, is discussed. [11] proposes a soft constraint-based framework to seamlessly express QoS properties reflecting both customer preferences and penalties applied to unfitting situations. The work in [2] discusses a constraint based approach for quality assurance in a choreography system. In [10], a new trend called service skyline computation for handling multiple quality criteria including user preferences, is proposed.

On the other hand, semantic technologies have been applied in recent works for reasoning about QoS of different systems during service composition. [7] proposes a meta-model for non functional property descriptions targeted to support the selection of Web services. The contribution in [3] focuses on the analysis of the requirements for a semantically rich QoS-based Web Service Description Model and an accurate, effective QoS-based WS Discovery (WSDi) process. [9] presents an overview of prominent research works related to developing QoS ontologies.

It is worthy to note that the above approaches do not handle the variations of QoS at runtime. An effective service composition management requires a support for the acceptance of QoS values, and should also allow for a flexible

acceptability for autonomous corrective actions. We provide here a probabilistic description logic based approach allowing self-healing in reaction to QoS variations by expressing their semantics.

In the literature, a number of works on probabilistic description logics exists [6, 5, 4]. In cite[4], the authors proposed a probabilistic extension of the expressive description logics P-SHIF(D) and P-SHOIN (D) that encompasses both statistical and subjective features, and also addresses the non-monotonic aspects of probabilistic knowledge using a semantics based on Lehmanns lexicographic entailment. In the cited approaches, the terminological and assertional knowledge is extended with conditional constraints and refered to as probabilistic knowledge. Those constraints express probabilities relating concepts or individual assertions. Our approach describes probabilities in a higher level of abstraction which allows to define probabilistic concepts that can be used in complex descriptions. We provide a reasoning algorithm dealing with such concepts.

3 Dynamic QoS Acceptability Framework

As depicted in Figure 1, our work can be integrated into a service composition system as a real-time QoS acceptability process. The inputs are the QoS required by the service selection process and the events captured by the monitoring process. The output is a set of acceptable services regarding their effective QoS. Events dealing with real-time execution of services are captured by the mon-

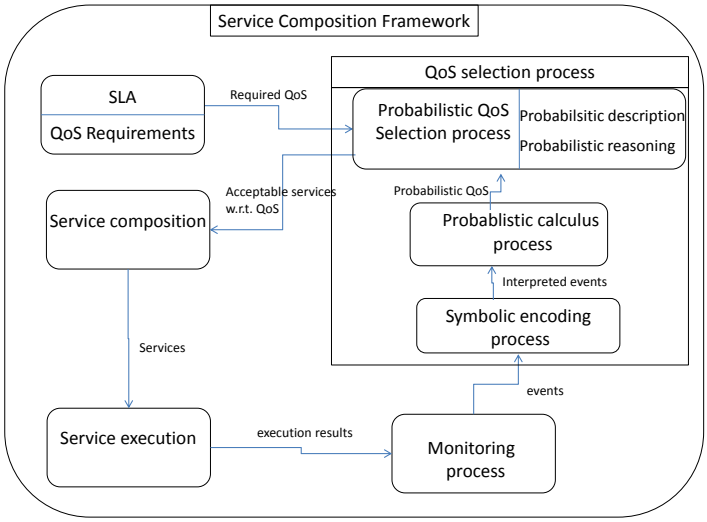


Fig. 1. QoS variations framework for service selection

itoring process. From these events, we extract real-time measures of the QoS provided by services. To efficiently handle such measures for the purpose of decision making, they are expressed thanks to symbolic concepts while taking into account their variations in a probabilistic way. We illustrate this idea through the following example.

Let us consider two services S_1 and S_2 involved in a web services composition. Assume that the monitoring process has captured the events given in Table 1.

Event	Execution time	Service ID	Response time (TR)	RAM Consuming (RC)
1	10:15:45	S_1	12ms	3MB
2	10:55:05	S_1	3ms	11MB
3	21:30:00	S_1	2ms	4MB
4	10:15:45	S_1	4ms	13MB
5	21:35:55	S_2	1ms	80MB
6	23:10:09	S_2	2ms	75MB

Table 1. Example of events captured by the monitoring process

3.1 Symbolic encoding of the events

Each quantitative measure captured by the monitoring process is interpreted by symbolic value using business rules. Business rules may be provided by the application domain experts in a Service Level Agreement (SLA). In what follow are examples of business rules:

- Rule 1 (for the interpretation of Time Response measures): A response time is said to be good (GoodTR) if it is less than 3ms, medium (MediumTR) if it is between 4ms and 6ms and bad (BadTR) otherwise.
- Rule 2 (for the interpretation of RAM consuming measures): A RAM consuming is said to be good (GoodRC) if it is less than 5Mb, bad (BadRC) otherwise.

Given the above two rules, the measures of Time Response and RAM are shown in Table 1 can be expressed in the following qualitative/symbolic descriptions:

- Event 1: S_1 : BadTR, S_1 : GoodRC.
- Event 2: S_1 : GoodTR, S_1 : BadRC.
- Event 3: S_1 : GoodTR, S_1 : GoodRC.
- Event 4: S_1 : MiddleTR, S_1 : BadRC.
- Event 5: S_2 : GoodTR, S_2 : BadRC.
- Event 6: S_2 : GoodTR, S_2 : BadRC.

3.2 Probabilistic QoS

The real-time QoS of a given service is not static but varies over different executions of the service. The QoS of each service has to be represented by a unique description in terms of a probability distribution [1]. Hence, the multiple values about a symbolic QoS are described using the Bayesian probability defined as follows:

$$P(SymbQoS|S_i) = \frac{P(SymbQoS \cap S_i)}{P(S_i)} = \frac{|SymbQoS \cap S_i|}{|S_i|} = \frac{number\ of\ appearences\ of\ SymbQoS\ in\ the\ events\ of\ S_i}{Number\ of\ executions\ of\ S_i}$$

where
$SymbQoS \in \{BadTR, GoodTR, MiddleTR, BadRC, GoodRC, MiddleRC\}$.

Back to the example introduced in section 3.1, the services S_1 and S_2 have been executed four times and twice, respectively. Hence, the probability values associated with their QoS can be computed as follows:

- S_1 received once BadTR. So, S_1 is BadTR with a probability of 0.25.
- S_1 received three times GoodTR. So, S_1 is GoodTR with a probability of 0.75.
- S_1 received twice GoodRC. So, S_1 is GoodRC with a probability of 0.5.
- S_1 received twice BadRC. So, S_1 is BadRC with a probability of 0.5.
- S_2 received twice GoodTR. So, S_2 is GoodTR with a probability of 1.
- S_2 received twice BadRC. So, S_2 is BadRC with a probability of 1.

3.3 Probabilistic QoS selection process

Given a required QoS, a traditional selection process decides whether a known service is acceptable or not. In our case, the selection process operates in an uncertain (probabilistic) environment since each QoS is described thanks to a probability distribution.

To this end, we propose a formal model based on probabilistic DLs to process acceptability of QoS. In this model, the required and provided QoS are described using a formal description language. A reasoning algorithm to decide the acceptability, is proposed as well.

As example for a required QoS, let's consider the following query: Retrieve the services that should provide a Good Response Time with a probability greater than 0.7 and Bad RAM Consuming with a probability less than 0.55. Intuitively, Only service S_1 is acceptable with respect to this required QoS, the Service S_2 is not acceptable because it does not provide Bad RAM consuming with probability less than 0.55.

[1] The interpretation of probability considered here is the so-called "the a-priori interpretation" which is the oldest and simplest one. The probability of an event is the number of favorable cases, where this event occurs, divided by the total numbers of possibles cases. So, w.r.t the same service and the same attribute, the required axiom that probabilities add up to 1 holds

In what follows, we present the proposed formal model for QoS acceptability decision making.

4 Probabilistic DL for Handling QoS Acceptability

The model relies on a probabilistic extension of DL defined with the following components:

- An expression language, including a set of symbols that are used to express knowledge about QoS.
- A semantic of symbols of the expression language, based on an interpretation domain and an interpretation function.
- A reasoning algorithm based on a set of inference rules to make a decision about the QoS acceptability.

Syntax of the description language
The proposed syntax is summarized in Table 2 (with $\circ \in \{<, >, \leq, \geq\}$). The introduced description language allows expressing two types of information related to QoS:

1. The knowledge about dynamic QoS, and
2. The information related to the required QoS.

Syntax	Interpretation
$Q_{\circ p}$	Refers to services that provide the QoS Q with a probability $\circ\ p$ ($Q_{\circ 1}$ is noted Q)
$Q1_{\circ p} \sqcap Q2_{\circ q}$	Refers to services that provide the QoS Q1 and Q2 with probability, respectively, $\circ\ p$ and $\circ\ q$
$Q1_{\circ p} \sqcup Q2_{\circ q}$	Refers to services that provide the QoS Q1 or Q2 with probability, $\circ\ p$ and $\circ\ q$, respectively
$\neg Q_{\circ p}$	Refers to services that do not have the QoS Q with a probability $\circ\ p$
$Q1 \sqsubseteq Q2$	Means: Any service that provides the QoS $Q1$, then it provides automatically the QoS $Q2$.

Table 2. Syntax of the proposed description language

For instance, the dynamic QoS of the example provided in the previous section can be written as follows:
$BadTR_{0.25}(S_1)$, $GoodTR_{0.75}(S_1)$, $GoodRC_{0.5}(S_1)$, $BadRC_{0.5}(S_1)$,

$$GoodTR_1(S_2),\ BadRC_1(S_2).$$

Similarly, the description of the required QoS can also be expressed using this language. For example, the following formulas are two different descriptions of possible required QoS:
$GoodTR_{\geq 0.7} \sqcap GoodRC_{\geq 0.4}$
$\neg BadTR_{\geq 0.1} \sqcap GoodRC_{\geq 0.9}$

In the first, we require a good time response with a minimum rate of 70% and good RAM Consuming with the minimum rate of 40%. While in the second one, we require services that provide good RAM Consuming with probability greater or equal than 90% and, at the same time, do not provide bad time response with probability greater than 10%.

Formal semantics of the description language
The formal semantics of terms and constructors of the description language is defined by the pair $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ where:

- $\Delta^{\mathcal{I}}$ is an interpretation domain. It is composed of individuals that represent the services managed in the BPM (Business Process Management) system.
- $.^{\mathcal{I}}$ is an interpretation function that assigns terms Q of the language to individuals of $\Delta^{\mathcal{I}}$ as shown in Table 4 (where $\bar{\circ}p$ stands for the complementary of $\circ p$, for instance $\bar{>}p = \leq p$):

Syntax	Formal interpretation
$Q_{\circ p}$	$Q^{\mathcal{I}}_{\circ p} = \{s \in \Delta^{\mathcal{I}} \mid Q^{\mathcal{I}}(s) \circ p\}$
$Q1_{\circ p} \sqcap Q2_{\circ q}$	$Q1^{\mathcal{I}}_{\circ p} \sqcap Q2^{\mathcal{I}}_{\circ q} = \{s \in \Delta^{\mathcal{I}} \mid Q1^{\mathcal{I}}(s) \circ p \wedge Q2^{\mathcal{I}}(s) \circ q\}$
$Q1_{\circ p} \sqcup Q2_{\circ q}$	$Q1^{\mathcal{I}}_{\circ p} \sqcup Q2^{\mathcal{I}}_{\circ q} = \{s \in \Delta^{\mathcal{I}} \mid Q1^{\mathcal{I}}(s) \circ p \vee Q2^{\mathcal{I}}(s) \circ q\}$
$\neg Q_{\circ p}$	$(\neg Q)^{\mathcal{I}}_{\circ p} = Q^{\mathcal{I}}_{\bar{\circ} p} = \{s \in \Delta^{\mathcal{I}} \mid Q^{\mathcal{I}}(s)\ \bar{\circ}\ p\}$
$Q1 \sqsubseteq Q2$	$\forall s \in \Delta^{\mathcal{I}}, Q1^{\mathcal{I}}(s) \Rightarrow Q2^{\mathcal{I}}(s)$

Table 3. Formal semantics of the proposed probabilistic DL

For instance, the probabilistic concept $Q_{\geq 0.5}$ is interpreted as the set of individuals with a probability degree greater than 0.5. Interpretation of complex and composed descriptions is also allowed as stated in Table 3.

5 Reasoning Algorithm for the Dynamic QoS Acceptability Decision

Based on the syntax and semantics of the description language presented in the previous section, we develop a reasoning algorithm that decides whether a provided dynamic QoS is acceptable with respect to a required one.

The decision problem is formulated as follows:
"Is the dynamic QoS provided by the service S_i acceptable with respect to a given required QoS RQ ?"

This acceptability decision problem can be formalized by the following logical deduction: $\langle (TBox, ABox) \rangle \models RQ(S_i)$
where $(TBox, ABox)$ is the knowledge base with two components:

– A TBox: containing knowledges about the considered domain application. For example, it may contain the following axioms:

  $\{GoodTR_1 \sqsubseteq BadTR_0,\ BadRC_1 \sqsubseteq GoodRC_0\}$
– An ABox: containing knowledges about dynamic QoS provided by services. For example, the ABox may contain the previous QoS of services S_1 and S_2:

  $\{BadTR_{0.25}(S_1), GoodTR_{0.75}(S_1), GoodRC_{0.5}(S_1), BadRC_{0.5}(S_1),$
  $GoodTR_1(S_2), BadRC_1(S_2)\}$

The logical deduction is achieved by checking the inconsistency of the following knowledge base:

$$\langle TBox, ABox \cup \{\neg RQ(S_i)\}\rangle$$

The inference machinery calls on the propagation rules introduced in the following sub-section.

5.1 Algorithm for QoS acceptability

A sample of the propagation rules that constitute the foundations of our probabilistic DL is depicted in Figure 2. Those rules are defined with respect to the formal semantics given in section 4. For sake of clarity, the $R_{reduction}$ rules are given for $\leq$ and $\geq$, one can easily deduce the corresponding rules for $<$ and $>$. The algorithm checks the consistency of the knowledge base:

$$\langle TBox, ABox \cup \neg RQ(S_i)\rangle$$

by applying the propagation rules until termination. Each application of a propagation rule generates a new inference system IS_i. The algorithm terminates if:

1. There exists a clash in the current inference system. In this case, the knowledge base is not consistent which means that the checked QoS is acceptable.
2. No more rule can be applied to generate a new inference system. In this case, the knowledge base is consistent, which means that the checked QoS is not acceptable.

An inference system IS_i contains clash if:

$$IS_i = \{Q_{<p}(s), Q_{>p'}(s), p \leq p'\}, \text{ or}$$
$$IS_i = \{Q_{\leq p}(s), Q_{\geq p'}(s), p < p'\}, \text{ or}$$
$$IS_i = \{Q_{<p}(s), Q_{\geq p'}(s), p < p'\}, \text{ or}$$
$$IS_i = \{Q_{\leq p}(s), Q_{>p'}(s), p < p'\}, \text{ or}$$
$$IS_i = \{Q_{=p}(s), Q_{=p'}(s), p \neq p'\}$$

The algorithm is sound because for any satisfiable concept of the proposed desciption logic, the application of propagation rules given in Fig. 2 terminates on at least one state free of clash. By the same way, starting from an inconsistant concept, all terminal states of the algorithm after applying propagation rules contain clash.

Rule R_1 :
$IS_i = \{Q(s)\} \longrightarrow IS_{i+1} = IS_i \cup \{Q_{\geq 1}(s)\}$

Rule R_0 :
$IS_i = \{\neg Q(s)\} \longrightarrow IS_{i+1} = IS_i \cup \{Q_{\leq 0}(s)\}$

Rule $R_\neg$:
$IS_i = \{\neg Q_{o\ p}(s)\} \longrightarrow S_{i+1} = IS_i \cup \{Q_{\bar{o}\ p}(s)\}$ (Note that: $\bar{<} =\geq, \bar{\leq} =>, \bar{>} =\leq, \bar{\geq} =<$)
Rule $R_\sqcap$:
$IS_i = \{(Q1_{\circ p} \sqcap Q2_{\circ q})(s)\} \longrightarrow IS_{i+1} = IS_i \cup \{Q1_{\circ p}(s), Q2_{\circ q}(s)\}$

Rule $R_\sqcup$:
$IS_i = \{(Q1_{\circ p} \sqcup Q2_{\circ q})(s)\} \longrightarrow IS'_{i+1} = IS_i \cup \{Q1_{\circ p}(s)\}, IS''_{i+1} = IS_i \cup \{Q2_{\circ q}(s)\}$

Rule $R_\sqsubseteq$:
$IS_i = \{(Q1_{\circ p} \sqsubseteq Q2_{\circ q}), Q1_{\circ p}(s)\} \longrightarrow IS_{i+1} = IS_i \cup \{Q2_{\circ q}(s)\}$

Rule $R'_\sqsubseteq$:
$IS_i = \{(Q1_{\geq p} \sqsubseteq Q2_{\circ q}), Q1_{=r}(s), p \leq r\} \longrightarrow IS_{i+1} = IS_i \cup \{Q2_{\circ q}(s)\}$

Rule $R_{reduction}$:
$IS_i = \{Q_{\geq p}(s), Q_{\leq p}(s)\} \longrightarrow IS_{i+1} = IS_i - \{Q_{\geq p}(s), Q_{\leq p}(s)\} + \{Q_{=p}(s)\}$
$IS_i = \{Q_{\geq p}(s), Q_{\geq q}(s), p \geq q\} \longrightarrow IS_{i+1} = IS_i - \{Q_{\geq q}(s)\}$
$IS_i = \{Q_{\geq p}(s), Q_{=q}(s), q \geq p\} \longrightarrow IS_{i+1} = IS_i - \{Q_{\geq p}(s)\}$
$IS_i = \{Q_{\leq p}(s), Q_{\leq q}(s), p \leq q\} \longrightarrow IS_{i+1} = IS_i - \{Q_{\leq q}(s)\}$
$IS_i = \{Q_{\leq p}(s), Q_{=q}(s), p \geq q\} \longrightarrow IS_{i+1} = IS_i - \{Q_{\leq p}(s)\}$

Fig. 2. A sample of the propagation rules

5.2 An illustrative example

We illustrate the use of the proposed algorithm to decide whether the dymanic QoS of services S_1 and S_2 is acceptable with respect to the following required QoS:

$$RQ \ = \ (GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8}) \sqcup GoodRC_{\geq 0.9}.$$

Acceptance of the QoS of S_2 The QoS of the service S_2 is acceptable w.r.t RQ if:

$\langle TBox_{exple}, ABox_{exple}\rangle \models RQ(S_2)$.

That is, $\langle TBox_{exple}, ABox_{exple} \cup \{\neg RQ(S_2)\}\rangle$ is inconsistent
where,
$TBox_{exple} = \{BadRC_1 \sqsubseteq GoodRC_0\}$
$ABox_{exple} = \{GoodTR_1(S_2), BadRC_1(S_2)\}$

The reasoning algorithm generates inference systems IS_i by applying the propagation rules as follows:

$IS_0 = \{TBox_{exple}, ABox_{exple}, \neg RQ(S_2)\}$
$= \{BadRC_1 \sqsubseteq GoodRC_0, GoodTR_1(S_2), BadRC_1(S_2), \neg[(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8}) \sqcup GoodRC_{\geq 0.9}](S_2)\}$
$IS_1 = IS_0 \cup \{GoodRC_0(S_2), [\neg(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8}) \sqcap \neg GoodRC_{\geq 0.9}](S_2)\}$ (generated by $R_{\sqsubseteq}$ and pushing the negation)
$IS_2 = IS_1 \cup \{\neg(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8})(S_2), \neg GoodRC_{\geq 0.9}(S_2)\}$ (generated by $R_{\sqcap}$)
$IS_3 = IS_2 \cup \{(GoodRC_{<0.4} \sqcup BadTR_{\geq 0.8})(S_2), GoodRC_{<0.9}(S_2)\}$ (generated by $R_{\neg}$)
$IS'_4 = IS_2 \cup \{(GoodRC_{<0.4}(S_2), GoodRC_{<0.9}(S_2)\}$ or $IS''_4 = IS_2 \cup \{(BadTR_{\geq 0.8})(S_2), GoodRC_{<0.9}(S_2)\}$ (IS'_4, IS''_4 generated by $R_{\sqcup}$)
$IS'_5 = IS'_4 \cup \{(GoodRC_{<0.4}(S_2)\}$ (generated by $R_{reduction}$ on IS'_4)
$IS'_6 = IS'_5 \cup \{(GoodRC_0(S_2)\}$ (generated by $R_{reduction}$ with $GoodRC_0(S_2)$)
The algorithm stops at the inference system IS'_6 because no more rule can be applied.

As there is no clash in IS'_6, the system is not inconsistent (without performing the inference system IS''_4). That is, it exists an interpretation such that:

$$\langle TBox_{exple}, ABox_{exple}\rangle \models \neg RQ(S_2)$$

We can deduce that the service S_2 is not acceptable w.r.t. the required QoS RQ.

Acceptance of the QoS of S_1 The QoS of the service S_1 is acceptable w.r.t RQ if:

$\langle TBox_{exple}, ABox_{exple}\rangle \models RQ(S_1)$.

That is,

$\langle TBox_{exple}, ABox_{exple} \cup \{\neg RQ(S_1)\}\rangle$ is inconsistant

where,

$TBox_{exple} = \{\Phi\}$
$ABox_{exple} = \{BadTR_{0.25}(S_1), GoodTR_{0.75}(S_1), GoodRC_{0.5}(S_1), BadRC_{0.5}(S_1)\}$

The reasoning algorithm generates inference systems IS_i by applying the propagation rules as follows:

$IS_0 = \{TBox_{exple}, ABox_{exple}, \neg RQ(S_1)\}$
$= \{BadTR_{0.25}(S_1), GoodTR_{0.75}(S_1), GoodRC_{0.5}(S_1), BadRC_{0.5}(S_1), \neg[(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8}) \sqcup GoodRC_{\geq 0.9}](S_1)\}$
$IS_1 = IS_0 \cup \{GoodRC_0(S_2), [\neg(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8}) \sqcap \neg GoodRC_{\geq 0.9}](S_1)\}$ (generated by pushing the negation)
$IS_2 = IS_1 \cup \{\neg(GoodRC_{\geq 0.4} \sqcap \neg BadTR_{\geq 0.8})(S_1), \neg GoodRC_{\geq 0.9}(S_1)\}$ (generated by $R_{\sqcap}$)
$IS_3 = IS_2 \cup \{(GoodRC_{<0.4} \sqcup BadTR_{\geq 0.8})(S_1), GoodRC_{<0.9}(S_1)\}$ (generated by $R_{\neg}$)
$IS'_4 = IS_3 \cup \{GoodRC_{<0.4}(S_1), GoodRC_{<0.9}(S_1)\}$ or $IS''_4 = IS_3 \cup \{BadTR_{\geq 0.8}(S_1), GoodRC_{<0.9}(S_1)\}$ (IS'_4, IS''_4 are generated by $R_{\sqcup}$)
IS''_4 contains the clash $\{BadTR_{\geq 0.8}(S_1), BadTR_{0.25}(S_1)\}$
$IS'_5 = IS'_4 \cup \{(GoodRC_{<0.4}(S_1)\}$ (generated by $R_{reduction}$ on IS'_4)
IS'_5 contains the clash : $GoodRC_{<0.4}(S_1), GoodRC_{0.5}(S_1)$

The algorithm stops because all the inference systems contain a clash. Therefore, the system is inconsistent.

This means that the service S_1 is acceptable w.r.t. the required QoS RQ.

6 Conclusion and Future Work

In this paper, we proposed a high level support for describing the semantic QoS variations in order to react to the service behavior changes. It provides an appropriate adaptation and avoid the process to grind to a halt. The key element of our model is the probabilistic extension of DL proposed. This extension allows to express the QoS requirements and performs an efficient reasoning mechanism to let the system self healing. Our variant of description logic has a minimal set of logical constructors. It contains only concept terms and two constructors which are conjunction and disjunction. As future work, we plan to extend the proposed approach by providing more constructors.

References

1. Marco Comuzzi and Barbara Pernici. A framework for qos-based web service contracting. *TWEB*, 3(3), 2009.
2. Dragan Ivanovic, Manuel Carro, and Manuel V. Hermenegildo. A constraint-based approach to quality assurance in service choreographies. In *ICSOC*, pages 252–267, 2012.
3. Kyriakos Kritikos and Dimitris Plexousakis. Requirements for qos-based web service description and discovery. *IEEE T. Services Computing*, 2(4):320–337, 2009.
4. Thomas Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6-7):852–883, 2008.
5. Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semant.*, 6(4):291–308, November 2008.
6. Carsten Lutz and Lutz Schröder. Probabilistic description logics for subjective uncertainty. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 393–403. AAAI Press; Menlo Park, CA, 2010.
7. Flavio De Paoli, Matteo Palmonari, Marco Comerio, and Andrea Maurino. A meta-model for non-functional property descriptions of web services. In *ICWS*, pages 393–400, 2008.
8. Barbara Pernici and Seyed Hossein Siadat. A fuzzy service adaptation based on qos satisfaction. In *CAiSE*, pages 48–61, 2011.
9. Vuong Xuan Tran and Hidekazu Tsuji. A survey and analysis on semantics in qos for web services. In *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications*, pages 379–385, Washington, DC, USA, 2009. IEEE Computer Society.
10. Qi Yu and Athman Bouguettaya. Multi-attribute optimization in service selection. *World Wide Web*, 15(1):1–31, 2012.
11. Mohamed Anis Zemni, Salima Benbernou, and Manuel Carro. A soft constraint-based approach to qos-aware service selection. In *ICSOC*, pages 596–602, 2010.

Abductive Reasoning with Description Logics: Use Case in Medical Diagnosis

Júlia Pukancová and Martin Homola

Comenius University in Bratislava,
Mlynská dolina, 84248 Bratislava, Slovakia
{pukancova,homola}@fmph.uniba.sk

Abstract. Ontologies have been increasingly used as a core representation formalism in medical information systems. Diagnosis is one of the highly relevant reasoning problems in this domain. In recent years this problem has captured attention also in the description logics community and various proposals on formalising abductive reasoning problems and their computational support appeared. In this paper, we focus on a practical diagnostic problem from a medical domain – the diagnosis of diabetes mellitus – and we try to formalize it in DL in such a way that the expected diagnoses are abductively derived. Our aim in this work is to analyze abductive reasoning in DL from a practical perspective, considering more complex cases than trivial examples typically considered by the theory- or algorithm-centered literature, and to evaluate the expressivity as well as the particular formulation of the abductive reasoning problem needed to capture medical diagnosis.

Keywords: Diagnosis, abduction, use case.

1 Introduction

Abduction, originally introduced by Peirce [15], is a form of backward reasoning, typically with a diagnostic rationale. We have a knowledge base $\mathcal{K}$ that is supposed to model some problem, and we have an observation O which is supposed to follow in situations captured by $\mathcal{K}$, but we are not able to explain O deductively, i.e., $\mathcal{K} \not\models O$. In abductive reasoning we ask the question – why is it that O does not follow from $\mathcal{K}$, and we look for a hypothesis (or, explanation) H such that, if added to $\mathcal{K}$, then O will follow from the resulting knowledge base. Moreover, we most typically look for explanations consisting of extensional rather than intensional knowledge, i.e., some set of ground facts that will, together with $\mathcal{K}$, explain O.

Abduction only recently captured the researchers' interest also in the area of ontologies and DL [7], where it also has some interesting applications, including possible explanations of incomplete modelling or incomplete matching [4], monitoring malfunctions in complex systems [11], and multimedia interpretation [16], among others.

The problem of diagnosis, often shown as a classic example of abductive reasoning [7,9], is highly relevant in the medical domain. Not only for primary diagnosis of a certain disease (as the model example in this use case), but also in emerging applications such as telemedical monitoring systems and ambient assisted living, where the patient's

condition is continually monitored and diagnosed for anomalies, therapy adherence, etc. Most of these applications nowadays heavily rely on ontologies (i.e., DL-based knowledge bases) that have been increasingly used as core representation formalism for clinical knowledge. While abduction over DL has been studied especially from the theoretical and from the algorithmic perspective, we are not aware of any case studies focusing on the practical aspects of modelling problems for abductive reasoning.

In this paper, we focus on a practical diagnostic problem from a medical domain: the diagnosis of diabetes mellitus. Based on information from clinical guidelines and other relevant sources (e.g., [1,2]) we formalize it in DL in such a way that the expected diagnoses are abductively derived. While we simplify the problem for reasons of conciseness, we do abstract a number of distinct, less or more problematic cases that need to be addressed, including: (a) dealing with the hierarchy of symptoms and possible diagnoses, (b) differential and elimination diagnosis, (c) associated conditions with similar symptoms, (d) distinguishing and reporting complications, and some more.

In the analysis that follows, we evaluate the modelled examples from the perspective of which particular variant of abduction is being addressed, what DL expressivity is needed, and we highlight the most important modelling issues that we run into.

In the end, we learned the following lessons: medical diagnosis especially requires ABox abduction, as hypothesizing the intensional knowledge in this domain is typically not desired – that is the area of domain experts. We were mostly able to model our simplified examples with the rather less expressive DL $\mathcal{ALC}$ for which abductive reasoning is available [9,12,13]. Though, examples requiring more complex constructs can also be found. Finally, modelling diagnostic knowledge bases with DL is different from modelling typical ontologies. In order to get the desired explanations the statements often need to be formulated more strongly, so that the desired observations follow. Also, to compare the generated hypotheses, at least part of the knowledge is used also deductively. Combining abductive and deductive reasoning within one knowledge base poses some difficulties, even on simplistic examples.

2 Abductive Reasoning with DL

The basic abductive framework for DL was introduced by Elsenbroich et al. [7] who proposed formulations for a number of distinct abductive problems. The main three types are summarized below. We will assume that the reader is already familiar with DL, the split of the knowledge base into the TBox (intensional knowledge) and the ABox (extensional knowledge), basic syntax, and semantics [3].

Definition 1 (Abduction problems in DL [7]). *An abduction problem is a pair $\mathcal{P} = (\mathcal{K}, O)$ such that $\mathcal{K}$ is a knowledge base in DL, and O a TBox or ABox assertion. A solution of $\mathcal{P}$ is any finite set H of TBox and ABox assertions such that $\mathcal{K} \cup H$ is consistent and $\mathcal{K} \cup H \models O$. In addition, $\mathcal{P}$ is called*

- ***TBox abduction problem**: if H is a set of TBox assertions and O is a TBox assertion.*
- ***ABox abduction problem**: if H is a set of ABox assertions and O is an ABox assertion.*

– ***Knowledge base abduction problem****: the general problem, i.e., if there are no restrictions on H and O.*

The definition gives a generic framework for abduction, but it is not very useful without further constraining the possible hypotheses. The number of possible explanations is very high, even infinite, therefore we need to be able to compare them and select the most preferred ones. The commonly used restrictions include [7]:

Definition 2. *Given an abduction problem $\mathcal{P} = (\mathcal{K}, O)$ and hypotheses H, H', we say that:*

1. *H is **consistent** if $H \cup \mathcal{K} \not\models \bot$, i.e. H is consistent w.r.t. $\mathcal{K}$*
2. *H is **relevant** if $H \not\models O$, i.e. H does not entail O*
3. *H is **explanatory** if $\mathcal{K} \not\models O$, i.e. $\mathcal{K}$ does not entail O*
4. *H is **stronger** than H' ($H \prec_{\mathcal{K}} H'$) if $\mathcal{K} \cup H$ entails H' (and vice-versa H is **weaker** than H' if $\mathcal{K} \cup H'$ entails H)*
5. ***minimal** if for every H', $H' \prec_{\mathcal{K}} H$, i.e., H is weaker than any other H'*

In general, H is a preferred solution if it is consistent, relevant, explanatory and there is no strictly weaker solution H' (i.e., such that $H \prec_{\mathcal{K}} H'$ and $H' \not\prec_{\mathcal{K}} H$). If there is single such solution, it is called the most preferred. Such hypotheses are (semantically) minimal. Minimality is important, because if there is a (strictly) weaker hypothesis than H it means that H hypothesizes too much. As abduction amounts to guessing, in a sense we do not want to guess more than necessary in order to derive the observation.

Consistency is required, because from inconsistency ($\mathcal{K} \cup H \models \bot$) one is able to derive everything. Such hypotheses would explain every observation and so it is not meaningful to find solutions not consistent with the knowledge base $\mathcal{K}$. The knowledge base $\mathcal{K}$ represents the background theory from which we are interested to derive the hypotheses. Therefore we should not be able to explain the observation without it. Such hypotheses (i.e., when $H \models O$) are therefore irrelevant. Similarly, an abduction problem only needs explaining the observation does not already follow from $\mathcal{K}$.

The diagnostic problems in the medical domain, which is our interest in this paper, will most often call for ABox abduction. This is because our aim is not to enrich the knowledge base with new axioms; these are typically sufficiently described by domain experts. Therefore our purpose is to build a knowledge base formalizing the available expert's knowledge (TBox) and use it to find explanations in form of facts (ABox assertions) to any observation, which is typically also a fact (ABox assertion).

A number of researchers addressed the computational solution for abduction problems. We will focus on those who addressed ABox abduction. Klarman et al. [12] proposed an algorithm for ABox abduction on top of $\mathcal{ALC}$ based on resolution. They show that it is sound and complete. Halland and Britz [9] developed, also for $\mathcal{ALC}$, a method based on the DL tableau algorithm. Ma et al. [13] also rely on th DL tableau algorithm, but extend the approach towards $\mathcal{ALCI}$. Completeness was not shown by Ma et al., while Halland and Britz explicitly note that their approach is incomplete.

Du et al. [5] solve abduction reasoning in DL via a reduction to logic programming, where abduction has been extensively studied [6]. In logic programming, abductive explanations are not arbitrary, but are typically drawn from a set of distinctive literals

called abducibles. This is because the user is often able to charaterzie in which part of the knowledge the hypothesis is expected, and thus to reduce the search space. To transfer this notion to the area of DL, Du et al. introduced a new variant of the ABox abduction problem, which we will call *simple*, in the form of $\mathcal{P}(\mathcal{K}, A, O)$. Besides for the knowledge base $\mathcal{K}$ and the observation O (set of concept assertions or atomic roles assertions), it adds the abducibles A – a set of atomic concepts and atomic roles. An abductive solution H for $\mathcal{P} = (\mathcal{K}, A, O)$ is a minimal set of ABox axioms composed of individuals of $\mathcal{K}$ and concepts or roles of A, such that $\mathcal{K} \cup H \models O$. The solution should be relevant, and consistent. The simple ABox abduction problem may be solved by reduction to logic programming, and consecutive evalaution by a readymade reasoning engine for logic programming.

3 Diabetes Mellitus Use Case

of the assumed simplification

Diabetes mellitus (DM) is a group of metabolic diseases characterized by hyperglycemia resulting from defects in insulin secretion, insulin action, or both. The chronic hyperglycemia of diabetes is associated with long-term damage, dysfunction, and failure of various organs, especially the eyes, kidneys, nerves, heart, and blood vessels [2]. We chose DM for our use case, as its diagnosis is a complex problem, with need to distinguish between particular subtypes and associated conditions, identification of possible complications, etc.

Our aim is to conceptualize a KB in DL that can be used for diagnosis of DM relying on abductive reasoning. As the problem is rather complex, we will concentrate on selected specific subproblems, and we will also abstract from some details which can be implemented analogously.

3.1 Hierarchy of Symptoms

Typical symptoms of diabetes mellitus include: frequent urination, excessive thirst, hyperglycemia, blurred vision, anorexia, weight loss, fatigue, and weakness. There are some more, but they can be added into the formalization analogously. In the following, diabetes mellitus is represented by the concept symbol DM, and the symptoms by $\mathsf{S}_1, \ldots, \mathsf{S}_8$, respectively. In abductive reasoning, we observe some set of symptoms and try to hypothesize the most relevant diagnosis. Therefore we record the relation between the diagnosis and its symptoms by the axiom:

$$\exists \mathsf{hasDiag}.\mathsf{DM} \sqsubseteq \exists \mathsf{hasSymp}.(\mathsf{S}_1 \sqcap \cdots \sqcap \mathsf{S}_8) \tag{1}$$

While literally (more precisely: deductively) this axiom means, that whoever has DM must simultaneously manifest all eight symptoms $\mathsf{S}_1, \ldots \mathsf{S}_8$, which is not always true; it allows to generate relevant abductive hypotheses: if the patient p is observed to have any subset of the symptoms $\mathsf{S}_1, \ldots, \mathsf{S}_8$ (e.g., we have an observation $O_1 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_2) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_8)$) then $H_1 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM}\}$ is among the generated diagnoses. Note that this kind of modelling is simplified, as it ignores uncertainties, often present in medical knowledge. In this paper we take this simplification as

we want to explore the possibilities of abduction with regular DL. Introducing uncertainties is left for future work.

There are, possibly, some relations between symptoms, like one symptom may be a more specific or a synonymous name for another, for instance, polydipsia (S_9) is a synonym for excessive thirst. This is modelled by adding:

$$\mathsf{S}_2 \equiv \mathsf{S}_9 \tag{2}$$

Now, whenever we observe S_9 instead of S_2 among some other symptoms of DM we derive equal hypotheses (e.g., if $O_2 = \{p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_9) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_7)\}$, H_1 is still abductively derived).

A more complex relationship among symptoms may occur in cases when some symptoms are conditions which may themselves be manifested by some other symptoms. For instance anorexia (S_5) is a condition associated with weight loss, fatigue, and weakness ($\mathsf{S}_6, \ldots, \mathsf{S}_8$). This may be modelled in two ways, adding either (3–4) or (5):

$$\mathsf{S}_5 \sqsubseteq \exists \mathsf{hasSymp}.(\mathsf{S}_6 \sqcap \cdots \sqcap \mathsf{S}_8) \tag{3}$$

$$\mathsf{hasSymp} \cdot \mathsf{hasSymp} \sqsubseteq \mathsf{hasSymp} \tag{4}$$

$$\mathsf{S}_5 \sqsubseteq \mathsf{S}_6 \sqcap \cdots \sqcap \mathsf{S}_8 \tag{5}$$

This then allows us to keep the axioms that relate diagnoses to symptoms more concise, e.g., we may now replace (1) by:

$$\exists \mathsf{hasDiag}.\mathsf{DM} \sqsubseteq \exists \mathsf{hasSymp}.(\mathsf{S}_1 \sqcap \cdots \sqcap \mathsf{S}_5) \tag{6}$$

Using either (2–4), (6) or (2), (5–6) as the KB, we equally derive the hypothesis H_1 for both observations O_1 and O_2, exactly as before.

As both of the above outlined solutions allow to derive the desired hypothesis, we will favour the solution on the right hand side, as it requires a less expressive DL (i.e., role transitivity axioms of the form (4) are not needed).

3.2 Hierarchy of Diagnoses

Potential diagnoses of our interest are listed in Table 1. These concepts can be readily taken from a number of medical ontologies. We chose to root them in SNOMED CT, where they are present as disorders. As in many cases in the medical domain, the main diagnosis (DM), has some subtypes which we want to distinguish. There are some alternate diagnoses (e.g., diabetes insipidus) which need to be rooted out or confirmed during the diagnostic process. And there are some related diagnoses (e.g., obesity, ketoacidosis) which may take part in the diagnosis of DM as relevant symptoms. The diagnoses thus form a hierarchy.

We will narrow down our focus on the diagnoses listed in Table 1. The respective part of the hierarchy is formalized using the following DL axioms:

$$\mathsf{DM1} \sqcup \mathsf{DM2} \sqcup \mathsf{GD} \sqsubseteq \mathsf{DM} \tag{7}$$

$$\mathsf{LADA} \sqsubseteq \mathsf{DM1} \tag{8}$$

The hierarchy of diagnoses has significant influence on the generated hypotheses. For instance, considering the axioms (5–6) formalizing the symptoms of DM, together

Table 1. Relevant diagnoses in the use case

Disorder (SNOMED)	DL
Diabetes mellitus	DM
Diabetes insipidus	DI
Diabetes mellitus type 1	DM1
Diabetes mellitus type 2	DM2
Latent autoimmune diabetes mellitus in adult	LADA
Gestational diabetes	GD
Obesity	Ob
Ketoacidosis	KA

with (7–8), if some of the symptoms of DM are observed (e.g., O_1 or O_2), the abductive reasoner will generate a number of diagnoses: $H_1 = \{p\colon \exists \mathsf{hasDiag.DM}\}$ as before, but in addition also $H_2 = \{p\colon \exists \mathsf{hasDiag.DM1}\}$, $H_3 = \{p\colon \exists \mathsf{hasDiag.DM2}\}$, $H_4 = \{p\colon (\exists \mathsf{hasDiag.DM1}) \sqcap (\exists \mathsf{hasDiag.DM}_2)\}$, and similar hypotheses for all (asserted or derived) subconcepts of DM. This is because they now all allow to derive the given observations. On the other hand, if we compare these hypotheses semantically, we see that $H_i \prec_{\mathcal{K}} H_1$ for $i > 1$ in the hypotheses above, as $\mathcal{K} \cup H_i \models H_1$, and never vice versa. Hence only H_1 will be preferred.

3.3 Distinguishing Between Two Diagnoses

A typical task in medical diagnostics is to distinguish one diagnosis from another, often similar one. This is called differential diagnosis. The two diagnoses may be distinguished by considering some symptoms relevant to one of them, but not the other.

We will consider DM1 and DM2; as differentiating between them is a relevant medical problem. DM1 and DM2 have some symptoms in common, but they also have some different symptoms. In this case, the common symptoms are those of DM. When the patient has some of these symptoms we say that she has DM – this case is covered by axioms (5–6).

Specific symptoms of DM1 include: belly pain, vomiting, fruity breath odor, drowsiness, and coma ($\mathsf{S}_{10}, \ldots, \mathsf{S}_{14}$). This is formalized as follows:

$$\exists \mathsf{hasDiag.DM1} \sqsubseteq \exists \mathsf{hasSymp.}(\mathsf{S}_{10} \sqcap \ldots \sqcap \mathsf{S}_{14}) \tag{9}$$

Analogously, the symptoms of DM2 include: skin problems, slow healing, tingling, numbness, and high BMI. We will name these as $\mathsf{S}_{15}, \ldots, \mathsf{S}_{19}$, which gives us the axiom:

$$\exists \mathsf{hasDiag.DM2} \sqsubseteq \exists \mathsf{hasSymp.}(\mathsf{S}_{15} \sqcap \ldots \sqcap \mathsf{S}_{19}) \tag{10}$$

Together the three diagnoses are now covered with (5–7) and (9–10). Let us consider some observations and the respective hypotheses:

- If we observe some set of symptoms that are common to both DM1 and DM2, e.g., having again the observation $O_1 = p\colon (\exists \mathsf{hasSymp.S}_2) \sqcap (\exists \mathsf{hasSymp.S}_8)$, then the most preferred diagnosis will be $H_1 = \{p\colon \exists \mathsf{hasDiag.DM}\}$. However, also $H_2 = \{p\colon \exists \mathsf{hasDiag.DM1}\}$, $H_3 = \{p\colon \exists \mathsf{hasDiag.DM2}\}$, $H_4 = \{p\colon (\exists \mathsf{hasDiag.DM1}) \sqcap$

$(\exists \mathsf{hasDiag}.\mathsf{DM}_2)\}$, will be valid abductive explanations (among others), but as we already discussed in Sect. 3.2 they are all stronger than H_1 and hence H_1 will be the most preferred.

- If we observe at least one symptom specific to DM1, e.g., having the observation $O_3 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_2) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_{10})$, this is no longer abductively explained by H_1, nor H_3. The most preferred hypothesis will be H_2. H_4 is an abductive explanation as well, but it is stronger than H_2, and hence H_2 is preferred.
- The case when we observe some specific symptoms of DM2 (but none of DM1) is exactly analogous.
- Finally, if we observe specific symptoms of both DM1 and DM2, e.g., we have $O_4 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_{10}) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_{15})$, the most preferred hypothesis that abductively explains this observation is H_4. This is because in case of H_1–H_3 there is always some symptom which is not explained. There are other explanations (e.g., $H_5 = \{p\colon \exists \mathsf{hasDiag}.(\mathsf{DM1} \sqcap \mathsf{DM2})\}$), but they are all stronger than H_4.

3.4 Case of Secondary Explanation of the Observation

During differential diagnosis it is also important to recognize cases when the given set of observed symptoms may have other possible explanations than the disease in question. For instance, one of the major symptoms of DM (hyperglycemia named as S_3) may be caused as a side effect of some medication.

As the axiom (6) includes S_3 as one of the DM symptoms, we are already able to answer on the observation of having S_3. Another possible explanation is taking the medications (name them M_1). So we add a new axiom:

$$\exists \mathsf{hasMedication}.\mathsf{M1} \sqsubseteq \exists \mathsf{hasSymp}.\mathsf{S}_3 \tag{11}$$

As this is a very simple conceptualization, there is only one observation by which this axiom play role. This observation is $O_5 = p\colon \exists \mathsf{hasSymp}.\mathsf{S}_3$. As well we have exactly two hypotheses $H_6 = \{p\colon \exists \mathsf{hasMedication}.\mathsf{M1}\}$ and $H_7 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM}\}$. Neither H_6 nor H_7 is stronger then the other one so both are preferred.

This result means, that we cannot be sure, which explanation is correct and we have to continue in diagnosting. We are satisfied with this answer because also in medical domain information about hyperglycemia presence is insufficient condition to decide between taking medications and having diagnosis DM.

3.5 Associated Conditions

In the medical domain, relations between diagnoses may come into play. Some associated conditions may have similar symptoms, or subset of symptoms as other diagnoses. Thus, in fact, in Axiom (9) the symptoms $\mathsf{S}_{10}, \ldots, \mathsf{S}_{14}$ are symptoms of an associated diagnosis called ketoacidosis. Similarly the symptom S_{19} is a symptom of obesity, an associated diagnosis of $\mathsf{DM2}$.

To take this into the account, we may add to (9) a new axiom, Axiom (12), and analogously to (10) a new axiom, Axiom (13), as follows:

$$\exists \mathsf{hasDiag}.\mathsf{KA} \sqsubseteq \exists \mathsf{hasSymp}.(\mathsf{S}_{10} \sqcap \ldots \sqcap \mathsf{S}_{14}) \tag{12}$$

$$\exists \mathsf{hasDiag}.\mathsf{Ob} \sqsubseteq \exists \mathsf{hasSymp}.\mathsf{S}_{19} \tag{13}$$

When we take the observations O_1 and O_3 and try to explain them using axioms (5–7) and (9–13), we see some changes:

- The case of observation $O_1 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_2) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_8)$ is not affected by the newly added axioms, as the symptom S_2 is not explained by any of them. So, the most preferred hypothesis is again H_1.
- If we observe at least one symptom specific to ketoacidosis together with some symptoms of DM (which are all symptoms of DM1 as we already know), e.g., having the observation $O_3 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_2) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_{10})$, then besides for $H_2 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM1}\}$ we have to consider also hypothesis $H_6 = \{p\colon (\exists \mathsf{hasDiag}.\mathsf{DM}) \sqcap (\exists \mathsf{hasDiag}.\mathsf{KA})\}$: they both explain O_3 and both are preferred to any other but they are mutually incomparable. But H_6 is certainly unexpected as so far we modelled the axioms in such a way that abductively the symptoms DM and ketoacidosis explain the diagnosis of DM1. To solve this we have to add yet another axiom:

$$\exists \mathsf{hasDiag}.\mathsf{DM} \sqcap \exists \mathsf{hasDiag}.\mathsf{KA} \sqsubseteq \exists \mathsf{hasDiag}.\mathsf{DM1} \tag{14}$$

  The axiom enables to compare the two hypotheses also deductively, i.e., any patient with both diagnoses DM and ketoacidosis is inferred to have also DM1. Hence H_6 is now stronger than H_2, and so H_2 is the single most preferred hypothesis.
- The case when we observe some specific symptoms of obesity together with symptoms of DM is analogous. We get the expected hypothesis $H_3 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM2}\}$, but to suppress $H_7 = \{p\colon (\exists \mathsf{hasDiag}.\mathsf{DM}) \sqcap (\exists \mathsf{hasDiag}.\mathsf{Ob})\}$ as less preferred, we need to add:

$$\exists \mathsf{hasDiag}.\mathsf{DM} \sqcap \exists \mathsf{hasDiag}.\mathsf{Ob} \sqsubseteq \exists \mathsf{hasDiag}.\mathsf{DM2} \tag{15}$$

- If we observe only the symptoms shared by obesity and DM2, in our simplified example, only $O_5 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_{18})$, two preferred and incomparable hypotheses are $H_8 = \{p\colon (\exists \mathsf{hasDiag}.\mathsf{Ob})\}$ and $H_3 = \{p\colon (\exists \mathsf{hasDiag}.\mathsf{DM2})\}$. In this case, this is in accord with the medical knowledge: this symptom can either be caused by one condition or by the other.

3.6 Case of Complications

However, in certain circumstances, one may wish to model the associated diagnoses differently, to capture a closer relation between them. This is, for instance, the case of DM1 and ketoacidosis. When we consider $O_3 = p\colon (\exists \mathsf{hasSymp}.\mathsf{S}_2) \sqcap (\exists \mathsf{hasSymp}.\mathsf{S}_{10})$ the modelling from the previous section gives the single most preferred diagnosis $H_2 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM1}\}$. While ketoacidosis is also indicated by symptom S_{10}, the hypothesis $H_9 = \{p\colon \exists \mathsf{hasDiag}.\mathsf{KA}\}$ is not an abductive explanation as it does not explain S_{10}.

This solution does not correctly capture the importance of ketoacidosis presence in diabetic patients. Ketoacidosis does not merely share symptoms with DM1, but it is an acute complication thereof which rarely occurs in non-diabetic individuals.

Therefore, we would intuitively want the explanation $H_{10} = \{p\colon \exists \mathsf{hasDiag}.\mathsf{DM1}) \sqcap (\exists \mathsf{hasDiag}.\mathsf{KA})\}$ in this case. In fact, H_{10} also explains O_3 in this case but is is stronger than H_2 and hence also less preferred.

To achieve this, we have to remodel the knowledge, so that, given some symptoms that are shared by both DM1 and ketoacidosis, only the conjunction of these diagnoses explains them: we exchange (9) together with (12) with a new axiom:

$$(\exists \mathsf{hasDiag.DM1}) \sqcap (\exists \mathsf{hasDiag.KA}) \sqsubseteq \exists \mathsf{hasSymptom.}(\mathsf{S}_{10} \sqcap \ldots \sqcap \mathsf{S}_{14}) \tag{16}$$

This however still does not force H_{10} as single most preferred, as due to the presence (7) and (14) we now have $H_{10} \prec_{\mathcal{K}} H_6$ and $H_6 \prec_{\mathcal{K}} H_{10}$, hence both H_{10} and H_6 are equally preferred. However, recall that we have asserted (14) only to support a slightly different relation between DM1 and ketoacedosis in the previous section, so when we drop this axiom then the single, most preferred diagnosis remains to be H_{10}.

This last move reflects the fact that in order to model different relationships between preferred explanations we need to manipulate also the deductive part of the KB (used during deductive comparisons of hypotheses) and, what is more, these manipulation is not just additive, it is selective, which may cause problems in more complex cases with higher number of variously interrelated diagnoses.

3.7 Case of Further Examination Needed

Consider again the case of one hypothesis being a more specific case of another. Typically, we have two diagnoses which have some common symptoms, while the more specific one likely has some additional symptoms (like with DM1 and DM, or DM2 and DM above). If we observe only some of the common symptoms, our previous modelling derives the less specific hypothesis. The more specific hypothesis also explains the observations, but there is no additional evidence for it, so we prefer the less specific one.

Most of the time this is expected, but not all the time. It may be necessary to differentiate between such hypotheses by all means, and so, if perhaps some evidence is lacking it should be obtained by additional examination if possible, by a laboratory test for instance. Hence the outcome from the diagnosis procedure should not be just the less specific hypothesis, but instead it should indicate also that additional tests are needed.

From a medical perspective this might be illustrated on the following problem: some patients who fit a certain profile (they are older, and not obese) and typically show symptoms common to DM1, may in fact have a specific type of DM1 called LADA (cf. Table 1 and axiom (8)). To differentiate between these two diagnoses, medical practitioners are advised to test antibodies (e.g., GADA) in a blood sample.

As this case leads to some complex modelling, we will explain it on simplified examples. Assume that LADA is a specific form of DM1, and that DM1 has some symptom (S'_1) and LADA has an additional specific one (S'_2). That is, we start from:

$$\mathsf{LADA} \sqsubseteq \mathsf{DM1} \tag{17}$$

$$\exists \mathsf{hasDiag.DM1} \sqsubseteq \exists \mathsf{hasSymp.}\mathsf{S}'_1 \tag{18}$$

$$\exists \mathsf{hasDiag.LADA} \sqsubseteq \exists \mathsf{hasSymp.}\mathsf{S}'_2 \tag{19}$$

Now we end up exactly in the situation described above. If only DM1 symptoms are observed (i.e., in our simplified case $O_6 = p\colon \exists \mathsf{hasSymp.}\mathsf{S}'_1$) then the preferred hypothesis is $H_2 = \{p\colon \exists \mathsf{hasDiag.DM1}\}$. In order to resolve this we may try to use (20) instead of (18):

$$(\exists \mathsf{hasDiag.DM1}) \sqcap (\exists \mathsf{needS\_LT}) \sqsubseteq \exists \mathsf{hasSymp.}\mathsf{S}'_1 \tag{20}$$

We now get the expected most preferred hypothesis $H_{11} = \{p\colon (\exists \mathsf{hasDiag.DM1}) \sqcap (\exists \mathsf{needS\_LT})\}$ for O_6 and so much for $O_7 = p\colon \exists \mathsf{hasSymp.S}'_2$ we will get $H_{12} = \{p\colon \exists \mathsf{hasDiag.LADA}\}$ as most preferred , however for $O_8 = p\colon (\exists \mathsf{hasSymp.S}'_1) \sqcap (\exists \mathsf{hasSymp.S}'_2)$ we now get $H_{12} = \{p\colon (\exists \mathsf{hasDiag.LADA}) \sqcap (\exists \mathsf{needS\_LT})\}$ as most preferred, as the need of the lab test is now necessary to explain S'_1. This is unintuitive as indeed once we observe S'_2 we know that the patient has LADA, no more tests are needed. It is easy to verify that it does not help to alter (19) to (21):

$$\exists \mathsf{hasDiag.LADA} \sqsubseteq \exists \mathsf{hasSymp.}(S'_1 \sqcap S'_2) \tag{21}$$

This is due to (17), (20), and (21) now for O_6 give both H_{11} and H_{12} as preferred, and they are incomparable. But clearly H_{12} is exactly the wrong hypothesis here. We are getting into some vicious circle.

The only solution we were able to come up with is to start treating the symptoms as completely specified. That is, either S'_1 or $\neg\mathsf{S}'_1$ is always part of the observation, and same for S'_2 or other symptoms possibly involved in this part of the derivation. Using (17), (22), and (23), we always get the expected results:

$$(\exists \mathsf{hasDiag.DM1}) \sqcap (\exists \mathsf{needS\_LT}) \sqsubseteq \exists \mathsf{hasSymp.}(\mathsf{S}'_1 \sqcap \neg\mathsf{S}'_2) \tag{22}$$

$$\exists \mathsf{hasDiag.LADA} \sqsubseteq \exists \mathsf{hasSymp.}(\mathsf{S}'_1 \sqcap \mathsf{S}'_2) \sqcap \exists \mathsf{hasSymp.}(\neg\mathsf{S}'_1 \sqcap \mathsf{S}'_2) \tag{23}$$

However, we now have to ask queries differently. For $O'_6 = p\colon \exists \mathsf{hasSymp.}(\mathsf{S}'_1 \sqcap \neg\mathsf{S}'_2)$ the most preferred hypothesis is H_{11}, and for both for $O'_7 = p\colon \exists \mathsf{hasSymp.}(\neg\mathsf{S}_1 \sqcap \mathsf{S}'_2)$ and for O_8 (no need to change here) we will get H_{12}.

So, we were able to get the expected results, but for a considerable price. Treating symptoms as completely specified is not in line with the usual intuitions behind abduction, where it is normal to assume that the observations are incomplete, and we are tasked to give the most appropriate explanation for any such given observation. In addition, it leads to a considerable blow up in the axioms, where all possible combinations of positive and negative symptoms need to be enumerated.

4 Discussion

All formalizations in our use case are instances of the ABox abduction problem as defined in Definition 1, based on Elsenbroich et al. [7]. The explanations that we seek are typically of a specific form of a single assertion:

$$p\colon (\exists R_1.C_1) \sqcap \cdots \sqcap (\exists R_n.C_n) \tag{24}$$

involving single patient p, where $R_1, \ldots, R_n$ most typically will come from some a priori known set of roles which are relevant based on the domain knowledge. A similar assumption may often hold also for the concepts $C_1, \ldots, C_n$ (e.g., most often these will be atomic concepts for various diagnoses and conditions relevant to the patients state).

While exceptions to these constraints may certainly be found (e.g., a second person involved from which the patient contracted the disease), in many cases the form of expected explanations will be reducible into atomic diagnoses by adding "interface" axioms of the form:

$$\mathsf{Diag1} \equiv (\exists R_1.C_1) \sqcap \cdots \sqcap (\exists R_n.C_n) \tag{25}$$

The hypothesis of the complex form (24) now reduces into the atomic form p: Diag1 which makes it possible to postulate the problem as simple ABox abduction. This is an important observation, as Du et al. [5] showed that the simple abduction problem can be effectively solved even for DLs up to $\mathcal{SHIQ}$.

Most of the examples we have shown rely in a fairly basic DL $\mathcal{ALC}$, abduction support for which is known [12,13,5,9]. We used complex role inclusions to capture dependencies between symptoms, but we also showed a simpler modelling which does not require it. Different complex DL constructs that might possibly be needed in more realistic situations certainly include number restrictions, and also restriction over concrete domains (known, e.g., in OWL [14]), that would enable to support some more elaborate statements about symptoms (the patient has at least some number of symptoms, or has a numeric value of some symptom from a specific range). Extensions of abductive reasoning for more expressive DLs that include such constructs are therefore desirable.

From a modelling perspective, an interesting lesson learned from the use case is that modelling a knowledge base to be used in a classical, deductive way, and modelling it to support abductive reasoning pose different, and sometimes conflicting requirements. As we noted in Sect. 3.1, it is often the case that certain explanation is expected to be valid for a number of symptoms, including any subsets thereof. Hence the typical approach that we relied upon is to formulate the axioms in a stronger fashion – the explanation implies all the symptoms, hence any subset follows. This kind of modelling is also demonstrated in the literature [7,5].

Firstly, we note that this requires the knowledge to be used in abductive application to be modelled differently than it is typically usual for ontologies, which normally are subject to deductive reasoning. Secondly, as we have repeatedly observed in this paper, even the process of abduction has an important subtask when hypotheses are compared and strictly deductive reasoning is used for this. This results in complex inter-relations between the "abductive" and the "deductive" part of the knowledge base and may lead to rather complicated modelling. A possible way how to deal with this is to treat the abductive and the deductive knowledge separately [16], or to used a more refined formulation of the abduction problem [10]. We plan to try this in the future.

Finally, we note that abduction has also a number of advantages, e.g., when compared to deductive diagnostic reasoning. In number of works [8,17,18] relying upon the latter, where deductive rules of the form $S_1, \cdots, S_n \rightarrow D$ are used, where S_i are symptoms and D is the diagnosis, the authors point out the problem occurring in case of two diagnoses D_1, D_2 with the former having a subset of symptoms of the latter. Using deductive inference, and observing all symptoms of D_2, both D_1 and D_2 are derived. This is counterintuitive as specific symptoms of D_2 were observed which are not symptoms of D_1. This problem has to be addressed by some suitable workarounds. In comparison, as we have demonstrated in the use case, abductive reasoning naturally eliminates the hypotheses which do not explain all of the observed symptoms.

Acknowledgments. This work was supported by VEGA project no. 1/1333/12. Júlia Pukancová is also supported by grant GUK no. UK/426/2015.

References

1. American Association of Clinical Endocrinologists: Diabetes care plan guidelines. Endocrine Practice 17 (2011)
2. American diabetes association: Diagnosis and classification of diabetes mellitus. Diabetes Care 31 (2008)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In: Proceedings of the 2003 International Workshop on Description Logics (DL2003), Rome, Italy September 5-7, 2003 (2003)
5. Du, J., Qi, G., Shen, Y., Pan, J.Z.: Towards practical ABox abduction in large description logic ontologies. Int. J. Semantic Web Inf. Syst. 8(2), 1–33 (2012)
6. Eiter, T., Gottlob, G., Leone, N.: Abduction from logic programs: Semantics and complexity. Theor. Comput. Sci. 189(1-2), 129–177 (1997)
7. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006 (2006)
8. García-Crespo, Á., Rodríguez, A., Mencke, M., Gómez-Berbís, J.M., Colomo-Palacios, R.: Oddin: Ontology-driven differential diagnosis based on logical inference and probabilistic refinements. Expert Systems with Applications 37(3), 2621–2628 (2010)
9. Halland, K., Britz, K.: Naïve ABox abduction in $\mathcal{ALC}$ using a DL tableau. In: Proceedings of the 2012 International Workshop on Description Logics, DL-2012, Rome, Italy, June 7-10, 2012. Sun SITE Central Europe (CEUR) (2012)
10. Hubauer, T., Lamparter, S., Pirker, M.: Relaxed abduction: Robust information interpretation for incomplete models. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011 (2011)
11. Hubauer, T., Legat, C., Seitz, C.: Empowering adaptive manufacturing with interactive diagnostics: A multi-agent approach. In: Advances on Practical Applications of Agents and Multiagent Systems - 9th International Conference on Practical Applications of Agents and Multiagent Systems, PAAMS 2011, Salamanca, Spain, 6-8 April 2011. pp. 47–56 (2011)
12. Klarman, S., Endriss, U., Schlobach, S.: ABox abduction in the description logic $\mathcal{ALC}$. Journal of Automated Reasoning 46(1), 43–80 (2011)
13. Ma, Y., Gu, T., Xu, B., Chang, L.: An ABox abduction algorithm for the description logic $\mathcal{ALCI}$. In: Intelligent Information Processing VI - 7th IFIP TC 12 International Conference. pp. 125–130 (2012)
14. OWL Working Group (ed.): OWL 2 Web Ontology Language Document Overview. Recommendation, W3C (27 October 2009)
15. Peirce, C.S.: Deduction, induction, and hypothesis. Popular science monthly 13, 470–482 (1878)
16. Petasis, G., Möller, R., Karkaletsis, V.: BOEMIE: Reasoning-based information extraction. In: Proceedings of the 1st Workshop on Natural Language Processing and Automated Reasoning co-located with 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), A Corunna, Spain, September 15th, 2013. pp. 60–75 (2013)
17. Rodríguez, A., Labra, J., Alor-Hernandez, G., Gómez, J.M., Posada-Gomez, R.: ADONIS: Automated diagnosis system based on sound and precise logical descriptions. In: Proc. of 22nd IEEE International Symposium (2009)

18. Rodríguez-González, A., Labra-Gayo, J.E., Colomo-Palacios, R., Mayer, M.A., Gómez-Berbís, J.M., García-Crespo, A.: SeDeLo: Using semantics and description logics to support aided clinical diagnosis. Journal of Medical Systems 36(4), 2471–2481 (2012)

TBox Reasoning in the Probabilistic Description Logic $\mathcal{SHIQ}_\mathcal{P}$

Viachaslau Sazonau and Uli Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{sazonauv,sattler}@cs.manchester.ac.uk

Abstract. One shortcoming of classic Descriptions Logics, DLs, is their inability to encode probabilistic knowledge and reason over it. This is, however, a strong demand of some modern applications, e.g. in biology and healthcare. Therefore, probabilistic extensions of DLs are attracting attention nowadays. We introduce the probabilistic DL $\mathcal{SHIQ}_\mathcal{P}$ which extends a known probabilistic DL. We investigate two reasoning problems for TBoxes: deciding consistency and computing tight probability bounds. It turns out that both problems are not harder than reasoning in the classic counterpart $\mathcal{SHIQ}$. We gain insight into complexity sources.

1 Introduction

Descriptions Logics [1], DLs, are a family of knowledge representation formalisms that form the basis for popular knowledge representation languages. In particular, they underly the Web Ontology Language [5], OWL, which is a W3C standard. Logical theories that encode a domain of interest in such languages are called ontologies. The last decade has witnessed a rapid growth in the number and size of ontologies which have become the common way to encode and share information in application areas such as medicine, biology, astronomy, defence and others.[1] Since DLs are essentially decidable fragments of first-order logic, FOL, ontologies are only capable to encode certain knowledge. Although some means of uncertainty, in fact, can be encoded, e.g. "a parent is a human who has some children" and "sky is sunny or cloudy", there are no built-in ways to represent probabilistic knowledge.

Successful application areas of DLs, however, often require modelling probabilistic knowledge which DLs are not able to deal with. The examples showing this appeal are evident in medicine. For instance, the medical ontology SNOMED CT [17] contains concepts mentioning "Probable cause", "Probable diagnosis", etc. This shortcoming in expressive power of classic DLs has recently caused non-classic proposals and various extensions. A recent survey is given in [12].

Two main approaches to probabilistic extensions of DLs differ in the view of probability. Halpern [4] makes a distinction between *statistical* and *subjective* probabilities. He formalizes statistical probabilities in "Type 1" probabilistic FOL and subjective probabilities in "Type 2" probabilistic FOL. The statistical view considers a probability

[1] http://bioportal.bioontology.org/

distribution over a *domain* that specifies the probability for an individual in the domain to be randomly picked. The subjective view is based on so-called *possible worlds* and specifies the probability distribution over a set of possible worlds [4]. We call the semantics of a probabilistic DL *statistical* if it uses the statistical view (Type 1 extension) and *subjective* if it is based on possible worlds (Type 2 extension).

The contributions of this work are as follows. Firstly, we introduce a new probabilistic extension of the classic DL $\mathcal{SHIQ}$, called $\mathcal{SHIQ}_\mathcal{P}$, with the statistical semantics inherited from the Type 1 probabilistic FOL that distinguishes it from many existing extensions whose semantics is based on possible worlds. We discuss relations to other extensions and some important features of the statistical semantics of $\mathcal{SHIQ}_\mathcal{P}$. Secondly, we study two reasoning problems for TBoxes in this extension: deciding consistency and computing tight probability bounds. We show that both problems are in ExpTime in the size of a TBox which implies that they are not harder than reasoning in the classic DL $\mathcal{SHIQ}$. The algorithm for solving each problem consists of two parts: detecting satisfiable types and solving a linear program on those types. We show that, in fact, the linear program can be built on the types over the probabilistic part only. Some examples and proofs are moved to Appendix.[2]

2 Syntax and Semantics of $\mathcal{SHIQ}_\mathcal{P}$

The syntax of $\mathcal{SHIQ}_\mathcal{P}$ extends classical $\mathcal{SHIQ}$ axioms with probabilistic statements over concepts. We assume the reader to be familiar with the DL $\mathcal{SHIQ}$ [6]. As usual, a classic $\mathcal{SHIQ}$ TBox $\mathcal{T}_c$ is a finite set of general concept inclusions and role inclusions.

Definition 1. *(TBox syntax) A $\mathcal{SHIQ}_\mathcal{P}$ TBox $\mathcal{T}$ is a set $\mathcal{T}_c \cup \mathcal{T}_p$, where $\mathcal{T}_c$ is a classic $\mathcal{SHIQ}$ TBox and $\mathcal{T}_p$ is a set of probabilistic statements. A probabilistic statement is a statement in one of the following forms:*

(i) $\sum_{j=1}^{m'} a'_j \cdot \mathbb{P}(C_j) \bowtie r'$ *(unconditional form);*
(ii) $\sum_{j=1}^{m''} a''_j \cdot \mathbb{P}(C_j|D) \bowtie r''$ *(conditional form);*

where $\bowtie \in \{<, \leq, \geq, >\}$, $a'_j, a''_j, r', r'' \in \mathbf{R}$, C_j, D *are possibly complex $\mathcal{SHIQ}$ concepts. We call concept inclusions, role inclusions, and probabilistic statements* axioms *of a TBox.*

We use the abbreviation $C \equiv D$ for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$ and $\sum_{j=1}^{m'} a'_j \cdot \mathbb{P}(C_j) = r'$ for $\{\sum_{j=1}^{m'} a'_j \cdot \mathbb{P}(C_j) \leq r',\ \sum_{j=1}^{m'} a'_j \cdot \mathbb{P}(C_j) \geq r'\}$ in (i) and the analogous one in (ii). Before defining the semantics we give an illustrative example, see Example 1.

Example 1. ($\mathcal{SHIQ}_\mathcal{P}$ TBox) According to statistics on smoking in England in 2010,[3] 47% of all adults in the sample are men (1); 20% and 25% of all adults (2), 20% and 29% of all men (3), 19% and 22% of all women (4) are current and former smokers, respectively. Current and former smokers are 17% and 28% of all married adults (5),

[2] www.cs.man.ac.uk/~sazonauv/SazonauDL15.pdf
[3] http://www.hscic.gov.uk/catalogue/PUB11454

18% and 33% of all married men (6), 17% and 23% of all married women (7). 34% and 25% of all diseases are caused by smoking for men and women (8), respectively. This knowledge can be expressed as follows:

$$
\begin{aligned}
&\mathcal{T} = \{S \sqsubseteq A,\ CS \sqsubseteq S,\ FS \sqsubseteq S,\\
&FS \sqsubseteq \neg CS,\ S \sqsubseteq CS \sqcup FS,\\
&M \sqsubseteq A,\ W \sqsubseteq A,\ M \sqsubseteq \neg W,\ A \sqsubseteq M \sqcup W,\\
&\mathbb{P}(M \mid A) = 0.47, && (1)\\
&\mathbb{P}(CS \mid A) = 0.2,\ \mathbb{P}(FS \mid A) = 0.25, && (2)\\
&\mathbb{P}(CS \mid M) = 0.2,\ \mathbb{P}(FS \mid M) = 0.29, && (3)\\
&\mathbb{P}(CS \mid W) = 0.19,\ \mathbb{P}(FS \mid W) = 0.22, && (4)\\
&\mathbb{P}(CS \mid \exists m.A) = 0.17, \mathbb{P}(FS \mid \exists m.A) = 0.28, && (5)\\
&\mathbb{P}(CS \mid M \sqcap \exists m.W) = 0.18,\ \mathbb{P}(FS \mid M \sqcap \exists m.W) = 0.33, && (6)\\
&\mathbb{P}(CS \mid W \sqcap \exists m.M) = 0.17,\ \mathbb{P}(FS \mid W \sqcap \exists m.M) = 0.23, && (7)\\
&\mathbb{P}(\exists d.(D \sqcap \exists c.S) \mid M) = 0.34,\ \mathbb{P}(\exists d.(D \sqcap \exists c.S) \mid W) = 0.25\}, && (8)
\end{aligned}
$$

where A, M, W, S, CS, FS, D are concepts representing adults, men, women, smokers, current smokers, former smokers, diseases, respectively; m, d, c are roles representing marriage, having a disease, having a cause, respectively.

The semantics of a $\mathcal{SHIQ}_{\mathcal{P}}$ TBox is based on the statistical view of probability and inherited from Halpern's Type 1 probabilistic FOL [4].

Definition 2. *(TBox semantics) An* interpretation *of a $\mathcal{SHIQ}_{\mathcal{P}}$ TBox is a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mu)$ where $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a standard $\mathcal{SHIQ}$ interpretation and μ a discrete*[4] *probability distribution over $\Delta^{\mathcal{I}}$. The semantics of concept and role inclusions is defined as usual. For concepts C, D we set $P(C^{\mathcal{I}}) = \sum_{d \in C^{\mathcal{I}}} \mu(d)$ and*

$$
P(C^{\mathcal{I}} | D^{\mathcal{I}}) = \begin{cases} \frac{P(C^{\mathcal{I}} \cap D^{\mathcal{I}})}{P(D^{\mathcal{I}})} & \text{if } P(D^{\mathcal{I}}) > 0 \\ 0 & \text{otherwise} \end{cases}
$$

An interpretation $\mathcal{I}$ satisfies a probabilistic statement

(i) $\sum_{j=1}^{m'} a'_j \cdot \mathbb{P}(C_j) \bowtie r'$ *if* $\sum_{j=1}^{m'} a'_j \cdot P(C_j^{\mathcal{I}}) \bowtie r'$ *holds;*
(ii) $\sum_{j=1}^{m''} a''_j \cdot \mathbb{P}(C_j | D) \bowtie r''$ *if* $\sum_{j=1}^{m''} a''_j \cdot P(C_j^{\mathcal{I}} | D^{\mathcal{I}}) \bowtie r''$ *holds.*

An interpretation $\mathcal{I}$ is called a model *of a TBox $\mathcal{T} = \mathcal{T}_c \cup \mathcal{T}_p$, written $\mathcal{I} \models \mathcal{T}$, if it satisfies each concept inclusion and role inclusion in $\mathcal{T}_c$ and each probabilistic statement in $\mathcal{T}_p$. A TBox $\mathcal{T}$ is* equivalent *to a TBox $\mathcal{T}'$ if $\mathcal{T}$ and $\mathcal{T}'$ have the same models. A TBox $\mathcal{T}$* entails *an axiom α, $\mathcal{T} \models \alpha$, if all models of $\mathcal{T}$ satisfy α. Given probability distribution μ, a TBox $\mathcal{T}$ entails an axiom α under μ, written $\mathcal{T} \models_{\mu} \alpha$, if all models of $\mathcal{T}$ with probability distribution μ satisfy α.*

[4] A discrete function has a finite or countably infinite set of inputs.

Since μ is a probability distribution, $\sum_{d\in\Delta^{\mathcal{I}}}\mu(d)=1$. Hence, $\mathbb{P}(\top)=1$ because $\top$ has the standard DL definition. It also follows from Definition 2 that $\mathbb{P}(\bot)=0$. Halpern [4] presents an axiom system, which includes standard probabilistic laws, for the Type 1 probabilistic FOL and shows that it is sound. Since the semantics of $\mathcal{SHIQ}_{\mathcal{P}}$ is derived from the Type 1 probabilistic FOL as its fragment, it follows that all standard probabilistic laws hold.

The syntax of $\mathcal{SHIQ}_{\mathcal{P}}$ allows arbitrary linear combinations of probabilities with the same conditioning concept, e.g. considering (2) in Example 1 we could express $\mathbb{P}(FS\mid A)=1.25\cdot\mathbb{P}(CS\mid A)$, or "former smokers are 25% more likely than current smokers to be met among adults".

One may notice that $C\sqsubseteq D$ and $\mathbb{P}(D|C)=1$ are semantically different. For example, an interpretation $\mathcal{I}$ with $C^{\mathcal{I}}=\emptyset$ is a model of $C\sqsubseteq D$ and not a model of $\mathbb{P}(D|C)=1$. On the other hand, interpretation $\mathcal{I}=\{\Delta^{\mathcal{I}}=\{a,b\},C^{\mathcal{I}}=\{a,b\},D^{\mathcal{I}}=\{b\},\mu(a)=0,\mu(b)=1\}$ is a model of $\mathbb{P}(D|C)=1$ and not a model of $C\sqsubseteq D$.

An unconditional probability $\mathbb{P}(C)$ is a special case of the conditional probability $\mathbb{P}(C|D)$ with $D\equiv\top$, i.e. $\mathbb{P}(C)=\mathbb{P}(C|\top)$. We can also write all conditional statements in the unconditional form. The following lemma states this.

Lemma 1. *Each probabilistic statement in $\mathcal{SHIQ}_{\mathcal{P}}$ has an equivalent unconditional form $\sum_{j=1}^{m}a_j\mathbb{P}(D_j)\bowtie r$ where $\bowtie\in\{\geq,>\}$, $a_j,r\in\mathbf{R}$, D_j is a $\mathcal{SHIQ}$ concept.*

Proof. See Appendix.

Therefore, we further assume without loss of generality that each statement in $\mathcal{T}_p$ is in unconditional form $\sum_{j=1}^{m}a_j\mathbb{P}(D_j)\bowtie r$, i.e. a linear combination of unconditional probabilities. Thus, a probabilistic TBox $\mathcal{T}_p$ that consists of n probabilistic statements is written as follows:

$$\textstyle\sum_{j=1}^{m_i}a_{ij}\mathbb{P}(D_{ij})\bowtie r_i, i=1..n.$$

The *signature* of $\mathcal{T}_c,\mathcal{T}_p$ is the set $\widetilde{\mathcal{T}}_c,\widetilde{\mathcal{T}}_p$ of all concept names and role names occurring in $\mathcal{T}_c,\mathcal{T}_p$, respectively. Thus, $\widetilde{\mathcal{T}}=\widetilde{\mathcal{T}}_c\cup\widetilde{\mathcal{T}}_p$. We call $|\widetilde{\mathcal{T}}|$ the *size* of $\mathcal{T}$. This way of measuring TBox size underestimates the usual size.

3 Illustration and Comparison of the Semantics

Now let us discuss representation capabilities of the statistical and subjective semantics. As mentioned above, the statistical semantics specifies a probability distribution μ over a domain $\Delta^{\mathcal{I}}$, i.e. $P(C^{\mathcal{I}})=\sum_{d\in C^{\mathcal{I}}}\mu(d)$, whereas the subjective one specifies a probability distribution μ over a set W of possible worlds (which correspond to realizable types), i.e. $P(C)=\sum_{I\in W\mid C\in I}\mu(I)$ [9]. In other words, the statistical semantics represents proportions of domain elements while the subjective one represents degrees of belief. Importantly, in this section the probability distribution μ is fixed, i.e. uniform, and reasoning under restricted distribution may be harder than under unrestricted one due to the reasons discussed in [14]. Let us illustrate some differences between the statistical semantics and the subjective semantics of [9].

Example 2. The following TBox is given:

$$\mathcal{T} = \{H \equiv (= 1\ m.W),\ W \equiv (= 1\ m^{-}.H),\\ \mathbb{P}(H) = 0.3\},$$

where H, W are concepts representing husbands and wives, respectively, m is a role representing marriage.

In Example 2, the TBox $\mathcal{T}$ implies that there are *exactly* as many husbands in the domain as there are wives. Hence, given μ is uniform, i.e. all individuals in the domain have the same probability μ_0 to be randomly picked, one can expect $\mathcal{T} \models_\mu \mathbb{P}(W) = 0.3$. However, the subjective semantics is not able to handle this because relationships between individuals within a *single world* are ignored by the semantics since it operates on a set of possible worlds. As a result, the following is entailed: $\mathcal{T} \models \mathbb{P}(W) \geq 0$. On the other hand, the statistical semantics does capture this:

$$P(W^{\mathcal{I}}) = \textstyle\sum_{d \in W^{\mathcal{I}}} \mu(d) = \mu_0 \cdot |W^{\mathcal{I}}|\\ = \mu_0 \cdot |H^{\mathcal{I}}| = \textstyle\sum_{d \in H^{\mathcal{I}}} \mu(d) = P(H^{\mathcal{I}}).$$

Example 3 illustrates that there are *at least* as many pets in the domain as there are pet owners. Given μ is uniform, under the statistical semantics the following is entailed: $\mathcal{T} \models_\mu \mathbb{P}(Pe) \geq 0.2$. In contrast, the subjective one gives $\mathcal{T} \models \mathbb{P}(Pe) \geq 0$ due to similar reasons as in Example 2.

Example 3. The following TBox is given:

$$\mathcal{T} = \{PeO \equiv \exists o.Pe,\ Pe \equiv (= 1\ o^{-}.PeO),\\ \mathbb{P}(PeO) = 0.2\},$$

where Pe, PeO are concepts representing pets and pet owners, respectively, o is a role representing ownership.

Example 2 and 3 show the capabilities of the statistical semantics to handle statistics. The statistical semantics allows for incorporating prior information about the probability distribution over the domain in a natural way that leads to possibly interesting entailments. In contrast, the subjective semantics ignores relationships between individuals within a single world and is not able to handle proportions.

4 TBox Reasoning in $\mathcal{SHIQ}_{\mathcal{P}}$

In the context of TBox reasoning in $\mathcal{SHIQ}_{\mathcal{P}}$ we investigate two reasoning problems: deciding consistency and computing tight probability bounds. We state the problems, develop decision procedures, investigate computational complexity and its sources.

4.1 Deciding Consistency

As usual, a $\mathcal{SHIQ}_\mathcal{P}$ TBox $\mathcal{T} = \mathcal{T}_c \cup \mathcal{T}_p$ is called *consistent* if it admits a model and *inconsistent* otherwise. We call PCon the problem of deciding consistency of a $\mathcal{SHIQ}_\mathcal{P}$ TBox. Further description requires the definition of *types* similar to (complete) types in classic DLs (see e.g. [1]). This should not be confused with the Type 1 and Type 2 logic given by Halpern.

Let $\mathcal{T}$ be a $\mathcal{SHIQ}_\mathcal{P}$ TBox. Let $\mathsf{sub}(\mathcal{T})$ be the set of all subconcepts of concepts occurring in $\mathcal{T}$, $\mathsf{nsub}(\mathcal{T}) = \{\dot{\neg}C \mid C \in \mathsf{sub}(\mathcal{T})\}$ the set of their negations in negation normal form, i.e. $\dot{\neg}C = NNF(\neg C)$, and $\mathsf{clos}(\mathcal{T}) = \mathsf{sub}(\mathcal{T}) \cup \mathsf{nsub}(\mathcal{T})$.

Definition 3. *A* type *of $\mathcal{T}$ is a set $t \subseteq \mathsf{clos}(\mathcal{T})$ such that the following conditions are satisfied:*

(i) $C \in t$ *iff* $\dot{\neg}C \notin t$, *for all* $C \in \mathsf{clos}(\mathcal{T})$;
(ii) $C \sqcap D \in t$ *iff* $C \in t$ *and* $D \in t$, *for all* $C \sqcap D \in \mathsf{clos}(\mathcal{T})$;
(iii) $C \sqcup D \in t$ *iff* $C \in t$ *or* $D \in t$, *for all* $C \sqcup D \in \mathsf{clos}(\mathcal{T})$;
(iv) $C \sqsubseteq D \in \mathcal{T}$ *and* $C \in t$ *implies* $D \in t$.

Given $\mathcal{I} \models \mathcal{T}$ and $e \in \Delta^\mathcal{I}$, we set $\mathsf{type}(e) = \{D \in \mathsf{clos}(\mathcal{T}) \mid e \in D^\mathcal{I}\}$. We say that a type t of $\mathcal{T}$ is *realized* in a model $\mathcal{I} \models \mathcal{T}$ if there is $e \in \Delta^\mathcal{I}$ such that $\mathsf{type}(e) = t$.

Definition 4. *A type t of $\mathcal{T}$ is* realizable *if $(\sqcap_{C \in t} C)$ is satisfiable w.r.t. $\mathcal{T}$.*

From now on, we consider only realizable types and omit "realizable". It is well-known that satisfiability w.r.t. a $\mathcal{SHIQ}$ TBox is decidable in ExpTime [19]. Lutz et al. (see Appendix in [15]) state the theorem for complexity of deciding consistency of a Prob1-$\mathcal{ALC}$ TBox and sketch the proof. It can be extended to a $\mathcal{SHIQ}_\mathcal{P}$ TBox.

Theorem 1. *Deciding consistency of a $\mathcal{SHIQ}_\mathcal{P}$ TBox $\mathcal{T}$ is* ExpTime*-complete.*

Proof. Given a $\mathcal{SHIQ}_\mathcal{P}$ TBox $\mathcal{T} = \mathcal{T}_c \cup \mathcal{T}_p$, we can compute the set T_c of all types of $\mathcal{T}_c$ in ExpTime. It is well-known for DLs with expressivity up to $\mathcal{SHIQ}$ that models are preserved under disjoint union (see e.g. [13]). This implies that there is always a $\mathcal{SHIQ}$ model $(\Delta^\mathcal{I}, \cdot^\mathcal{I}) \models \mathcal{T}_c$ where all (realizable) types are realized. If $\widetilde{\mathcal{T}}_p \backslash \widetilde{\mathcal{T}}_c \neq \emptyset$, T_c is trivially extended to match $\widetilde{\mathcal{T}}$ and denoted T.

Let a variable x_t be associated with each type $t \in T$. Then, by Definition 2 and Lemma 1, the TBox $\mathcal{T}$ induces the system of linear inequalities:

$$E(\mathcal{T}) := \begin{cases} \sum_{t \in T} x_t = 1;\ x_t \geq 0 \text{ for each } t \in T; \\ \sum_{j=1}^{m_i} a_{ij} \sum_{D_{ij} \in t} x_t \bowtie r_i,\ i = 1..n \end{cases}$$

System $E(\mathcal{T})$ can be solved using linear programming with the constant objective. Since linear programming is in P [18] and $E(\mathcal{T})$ is of exponential size in $\mathcal{T}$ we can decide in ExpTime whether there is a solution. It is sufficient to show that $E(\mathcal{T})$ has a solution iff $\mathcal{T}$ is consistent.

"If". Assume that $E(\mathcal{T})$ has a solution $\dot{X} = \{\dot{x}_t \mid t \in T\}$. There is a classic model $(\Delta^\mathcal{I}, \cdot^\mathcal{I}) \models \mathcal{T}_c$ where all types are realized. Choose any μ satisfying $\sum_{\mathsf{type}(d)=t} \mu(d) =$

$\dot{x}_t, d \in \Delta^{\mathcal{I}}$, e.g. if $\mathsf{type}(d) = t$ then $\mu(d) = \dot{x}_t/(\#\{e \in \Delta^{\mathcal{I}} \mid \mathsf{type}(e) = t\})$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mu)$. Then, by Definition 2 of the semantics and Lemma 1

$$\begin{aligned}\textstyle\sum_{D_{ij} \in t} \dot{x}_t &= \textstyle\sum_{D_{ij} \in t} \sum_{\mathsf{type}(d)=t} \mu(d) \\ &= \textstyle\sum_{d \in D_{ij}^{\mathcal{I}}} \mu(d) = P(D_{ij}^{\mathcal{I}}).\end{aligned}$$

This implies that all probabilistic statements in $\mathcal{T}_p$ are satisfied. Hence, $\mathcal{I} \models \mathcal{T}$.

"Only If". Assume $\mathcal{T}$ is consistent, i.e. there is a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mu) \models \mathcal{T}$. Let $\dot{x}_t := \sum_{\mathsf{type}(d)=t} \mu(d)$ for each $t \in T$. By Definition 2 of the semantics and Lemma 1, each probabilistic statement $\sum_{j=1}^{m_i} a_{ij} P(D_{ij}^{\mathcal{I}}) \bowtie r_i,\ i = 1..n$ holds. We observe that

$$\begin{aligned}P(D_{ij}^{\mathcal{I}}) &= \textstyle\sum_{d \in D_{ij}^{\mathcal{I}}} \mu(d) \\ &= \textstyle\sum_{D_{ij} \in t} \sum_{\mathsf{type}(d)=t} \mu(d) = \sum_{D_{ij} \in t} \dot{x}_t.\end{aligned}$$

Therefore, $\dot{X} = \{\dot{x}_t\}$ is a solution of $E(\mathcal{T})$. □

Reduction of a TBox *entailment* $\mathcal{T} \models C \sqsubseteq D$ to consistency is analogous to $\mathcal{SHIQ}$. By Lemma 1, a probabilistic TBox entailment is reduced to consistency as follows:

$$\begin{aligned}&\mathcal{T} \models \textstyle\sum_{j=1}^{m} a_j \mathbb{P}(D_j) \bowtie r \text{ iff} \\ &\mathcal{T} \cup \{(-1) \textstyle\sum_{j=1}^{m} a_j \mathbb{P}(D_j) \dot{\bowtie} -r\} \text{ is inconsistent,} \\ &\text{where } \dot{\bowtie} = \begin{cases} > \text{ if } \bowtie \text{ is } \geq \\ \geq \text{ if } \bowtie \text{ is } > \end{cases}.\end{aligned}$$

Thus, deciding TBox consistency, and consequently TBox entailment, in $\mathcal{SHIQ}_{\mathcal{P}}$ is not harder than in $\mathcal{SHIQ}$. The procedure is based on solving a system of linear inequalities over variables representing types.

4.2 Computing Tight Probability Bounds

In addition to consistency checking, one may be interested in probabilistic entailments that come in form of *tight bounds* $\mathcal{T} \models \mathbb{P}(C|D) \geq p_\ell$ and $\mathcal{T} \models \mathbb{P}(C|D) \leq p_u$. For example, tight bounds can identify possible data flaws, see Appendix. We define the reasoning task of computing tight probability bounds in $\mathcal{SHIQ}_{\mathcal{P}}$.

Definition 5. *(Tight bounds) Let $\mathcal{T}$ be a consistent $\mathcal{SHIQ}_{\mathcal{P}}$ TBox.*

- *A real value $p_\ell \in [0,\ 1]$ is a* lower bound *for $\mathbb{P}(C|D)$ w.r.t. $\mathcal{T}$ if $\mathcal{T} \models \mathbb{P}(C|D) \geq p_\ell$. p_ℓ is a* tight lower bound *for $\mathbb{P}(C|D)$ w.r.t. $\mathcal{T}$ if p_ℓ is maximal.*
- *A real value $p_u \in [0,\ 1]$ is an* upper bound *for $\mathbb{P}(C|D)$ w.r.t. $\mathcal{T}$ if $\mathcal{T} \models \mathbb{P}(C|D) \leq p_u$. p_u is a* tight upper bound *for $\mathbb{P}(C|D)$ w.r.t. $\mathcal{T}$ if p_u is minimal.*
- *A real value $p_o \in [0,\ 1]$ is called a* tight bound *for $\mathbb{P}(C|D)$ w.r.t. $\mathcal{T}$ if p_o is a tight lower or upper bound.*

– TEnt *is the problem of computing a pair* p_ℓ, p_u *for* $\mathbb{P}(C|D)$*, written* $\mathcal{T} \models_{tight} \{\mathbb{P}(C|D) \geq p_\ell,\ \mathbb{P}(C|D) \leq p_u\}$*. We set* $\ell[\mathbb{P}(C|D)] = p_\ell$ *and* $u[\mathbb{P}(C|D)] = p_u$ *for a tight lower and upper bound of* $\mathbb{P}(C|D)$*, respectively.*

We now state TEnt as an optimization problem.

Lemma 2. *Given a consistent* $\mathcal{SHIQ}_\mathcal{P}$ *TBox* $\mathcal{T}$*, tight bounds* p_ℓ, p_u *for* $\mathbb{P}(C|D)$ *are computed by solving the optimization problem, called* $OP(\mathcal{T})$*:*

$$\begin{aligned} &\textit{maximise} \quad s \cdot \frac{\sum_{C \sqcap D \in t} x_t}{\sum_{D \in t} x_t} \\ &\textit{subject to} \quad E(\mathcal{T}), \end{aligned}$$

such that $p_\ell = -p',\ p_u = p''$*, where* p', p'' *are optimal values of* $OP(\mathcal{T})$ *when* $s = -1$ *and* $s = 1$*, respectively.*

Proof. See Appendix.

As one can see, the objective function in $OP(\mathcal{T})$ is not linear. Fortunately, $OP(\mathcal{T})$ can be translated into an equivalent linear program using a substitution $x_t = y_t/z$ [2]:

$$\begin{aligned} \text{maximise} \quad & s \cdot \textstyle\sum_{C \sqcap D \in t} y_t \\ \text{subject to} \quad & \textstyle\sum_{t \in T} y_t - z = 0;\ y_t \geq 0 \text{ for each } t \in T; \\ & \textstyle\sum_{j=1}^{m_i} a_{ij} \sum_{D_{ij} \in t} y_t - r_i z \bowtie 0, i = 1..n; \\ & \textstyle\sum_{D \in t} y_t = 1;\ z > 0 \end{aligned}$$

In case of unconditional probability, a substitution is not required (i.e. it is trivial), since the objective function is already linear:

$$\begin{aligned} \text{maximise} \quad & s \cdot \textstyle\sum_{C \in t} x_t \\ \text{subject to} \quad & E(\mathcal{T}) \end{aligned}$$

Thus, the optimization problem $OP(\mathcal{T})$ is reducible to a linear program. Since linear programming is in P [18], finding a solution of $OP(\mathcal{T})$ requires a polynomial number of iterations in the size of $OP(\mathcal{T})$. Nonetheless, the optimization problem $OP(\mathcal{T})$ is of exponential size w.r.t. the TBox $\mathcal{T}$ due to exponentially many types $t \in T$. Therefore, computing TEnt is in ExpTime in the size of $\mathcal{T}$ which is stated by the following theorem.

Theorem 2. *Given a consistent* $\mathcal{SHIQ}_\mathcal{P}$ *TBox* $\mathcal{T}$*, computing tight probability bounds is in* ExpTime *in the size of* $\mathcal{T}$*.*

4.3 More Detailed Complexity Analysis

One can notice that signatures of classic $\widetilde{\mathcal{T}}_c$ and probabilistic part $\widetilde{\mathcal{T}}_p$ do not necessarily coincide. In particular, a probabilistic $\widetilde{\mathcal{T}}_p$ can be much smaller than classic $\widetilde{\mathcal{T}}_c$, $|\widetilde{\mathcal{T}}_c| \gg |\widetilde{\mathcal{T}}_p|$, e.g. for medical knowledge bases. Once realizable types are obtained from $\mathcal{T}_c$,

$\mathcal{T}_p$ produces an optimization problem $OP(\mathcal{T})$ which includes variables for each type from $\mathcal{T}_c$. If $|\widetilde{\mathcal{T}}_c| \gg |\widetilde{\mathcal{T}}_p|$, this makes $OP(\mathcal{T})$ unreasonably large and, consequently, reasoning over relatively simple probabilistic parts hard. Fortunately, as the following lemma shows, this can be avoided via a refinement of $OP(\mathcal{T})$.

Lemma 3. *Given a consistent TBox $\mathcal{T} = \mathcal{T}_c \cup \mathcal{T}_p$, there is a refinement of $OP(\mathcal{T})$, $OP_p(\mathcal{T})$, that gives the same tight bounds as $OP(\mathcal{T})$ and has exponentially many variables in the size of $\mathcal{T}_p$.*

Proof. Let T be the set of all types of $\mathcal{T}$ as above and T_p the set of all types of $\mathcal{T}_p$ alone. Let T' be the set of types of $\mathcal{T}_p$ "allowed" by $\mathcal{T}_c$, i.e. $T' := \{\tau \in T_p \mid \text{there is } t \in T \text{ s.t. } \tau \subseteq t\}$. Then, sums in the optimization problem $OP(\mathcal{T})$ can be rewritten as follows:

$$\textstyle\sum_{C \in t} x_t = \sum_{C \in \tau} \sum_{\tau \subseteq t} x_t = \sum_{C \in \tau} x_\tau,$$

where $t \in T, \tau \in T'$. Thus, every sum in $OP(\mathcal{T})$, except $\sum_{t \in T} x_t$, is potentially "squeezed" via substitutions, since $|T'| \leq |T|$. Let $T'' = \{t \in T \mid \text{there is no } \tau \in T' \text{ s.t. } \tau \subseteq t\}$. Then $\sum_{t \in T} x_t = \sum_{\tau \in T'} x_\tau + \sum_{t \in T''} x_t = \sum_{\tau \in T'} x_\tau + x$. Constraints $\{x_t \geq 0 \mid \text{there is } \tau \in T' \text{ s.t. } \tau \subseteq t\}$ are substituted by corresponding constraints $\{x_\tau \geq 0\}$. Constraints $\{x_t \geq 0 \mid t \in T''\}$ are substituted by single constraint $\{x \geq 0\}$.

Let $OP_p(\mathcal{T})$ be the optimization problem obtained from $OP(\mathcal{T})$ via the aforementioned substitutions. Since $|T'| \leq |T_p| = 2^{|\widetilde{\mathcal{T}}_p|}$, the number of variables in $OP_p(\mathcal{T})$ is at most $2^{|\widetilde{\mathcal{T}}_p|} + 1$. □

The result of Lemma 3 also holds for deciding TBox consistency in $\mathcal{SHIQ}_\mathcal{P}$.

Corollary 1. *Given a TBox $\mathcal{T} = \mathcal{T}_c \cup \mathcal{T}_p$, there is a refinement of $E(\mathcal{T})$, $E_p(\mathcal{T})$, that gives the same consistency result as $E(\mathcal{T})$ and has exponentially many variables in the size of $\mathcal{T}_p$.*

Lemma 3 gives a procedure to reduce, possibly massively, the number of variables in the linear program for both computing TEnt and deciding PCon. This is achieved via substituting suitable sums of type variables by fresh variables. As a result, the number of variables is reduced from exponentially many w.r.t. $\mathcal{T}$ to exponentially many w.r.t. $\mathcal{T}_p$ only.

It should be noted that Lemma 3 and Corollary 1 do not provide new complexity results: one has to compute the set of realizable types which is still in ExpTime. Nevertheless, it gives insight into complexity sources and may lead to a significant optimization because otherwise the size of a probabilistic part has to be significantly limited, e.g. in [16].

5 Related Work

There are several criteria to distinguish the existing approaches. Firstly, we distinguish *loose* and *tight* proposals to handle probabilities. Loose ones are mainly based

on the combination of logic with probabilistic graphical models such as Bayesian networks [10, 20, 3]. They mainly admit a single model. The influence in one direction is typical for them: logical knowledge affects probabilistic knowledge but not the other way around. Among their drawbacks is limited expressivity: the syntax is restricted due to underlying graphical models. In addition, the graphical models often have large sizes that make them hardly manageable. They can also require non-domain assumptions such as probabilistic independences.

By tight proposals we mean those which attempt to deal with uncertainty in purely logical ways. In contrast to loose proposals, they admit multiple models. Influence works in both directions between logical and probabilistic knowledge. The probabilistic DL $\mathcal{SHIQ}_{\mathcal{P}}$ is a tight proposal by this definition.

Probabilistic extensions differ in their syntax and semantics. One of the major syntax differences is how and where probabilities can occur. In particular, probabilistic statements can be added to classical DL axioms [11, 9] or probabilities can be embedded into axioms via application to concepts and roles [15], e.g. $\mathbb{P}_{=0.2}(A) \sqsubseteq CS$ expresses "20 % of adults are current smokers". The syntax of $\mathcal{SHIQ}_{\mathcal{P}}$ is an example of the first approach.

As pointed out above, there are two views of probability that separate the existing probabilistic DLs by their semantics: statistical and subjective. The statistical view has been intensely used by non-logic extensions where a probability distribution is typically represented by graphical models [10, 20, 3]. The examples of logic extensions taking the statistical view are [7, 15]. The subjective view is associated with possible worlds which are, in fact, the core of the semantics for many existing extensions [11, 15, 9]. Lutz et al. [15] obtain the probabilistic DL Prob-$\mathcal{ALC}$ with the subjective semantics from Halpern's Type 2 probabilistic FOL and study its expressivity and computational properties. They also derive Prob1-$\mathcal{ALC}$ from Halpern's Type 1 probabilistic FOL. The statistical semantics of $\mathcal{SHIQ}_{\mathcal{P}}$ is similarly inherited from Halpern's Type 1 probabilistic FOL. A TBox in $\mathcal{SHIQ}_{\mathcal{P}}$ admits linear combinations of conditional probabilities which are not explicitly permitted in Prob1-$\mathcal{ALC}$.

TBox reasoning in $\mathcal{SHIQ}_{\mathcal{P}}$ is the combination of classic DL reasoning and linear programming. The extensions with the subjective semantics [11, 15, 9] also perform reasoning via solving systems of linear inequalities, commonly over possible worlds. This has been implemented and used for medical applications [8].

6 Summary and Future Work

In this work, we study the probabilistic extension of the DL $\mathcal{SHIQ}$ that we call $\mathcal{SHIQ}_{\mathcal{P}}$. It has the statistical semantics inherited from the Type 1 probabilistic FOL [4]. We investigate two reasoning problems for TBoxes in $\mathcal{SHIQ}_{\mathcal{P}}$: deciding consistency PCon and computing tight probability bounds TEnt. We obtain the ExpTime complexity bounds for deciding consistency of a $\mathcal{SHIQ}_{\mathcal{P}}$ TBox. While deciding consistency of TBoxes is already explored for Prob1-$\mathcal{ALC}$, to the best of our knowledge, no studies are carried out for the problem of computing tight probability bounds for any extension with the same semantics. We state this problem for $\mathcal{SHIQ}_{\mathcal{P}}$ as an optimization problem. We also show that solving this optimization problem is, in fact, reducible to

linear programming and, hence, is in P w.r.t. its size. Therefore, computing TEnt is in ExpTime in the size of the TBox $\mathcal{T}$. Thus, both PCon and TEnt are not harder than reasoning in the classic DL $\mathcal{SHIQ}$. Moreover, we show that the size of a linear program for PCon and TEnt can be (significantly) reduced since it depends on the probabilistic part only which is an important insight into the sources of computational complexity.

In Section 3 we discuss abilities of the statistical semantics to handle the statistical knowledge and observe that it is naturally suited for this purpose. As noted by several authors [4, 11, 15], the main shortcoming of the statistical semantics, however, is its inability to represent probabilistic assertional knowledge, i.e. degrees of belief. For example, there is no way to encode "Martin is a smoker with probability 0.7" since he is either a smoker or he is not, according to the statistical view. Lutz et al. [15] state that "only TBox reasoning is relevant" for the statistical semantics and exclude ABoxes.

Nevertheless, we argue that ABox reasoning *may be relevant* for the statistical semantics if it is extended to capture population wide, incomplete data, i.e. it is of sufficient size and spread w.r.t. the whole population. Reasoning over such data might help to answer the question how well the data fits the background knowledge including probabilistic statements, i.e. whether they are compatible with the proportions of individuals in the data. In case of incomplete data, the incompatibility might show whether and which information is missing. Thus, we plan to further study ABox reasoning in the statistical semantics and its possible extensions.

As Example 2 and 3 show, prior knowledge about probability distribution acts as an additional parameter to the reasoning procedure. We plan to further investigate how restrictions of a probability distribution affect entailments and their complexity. We also consider extending expressivity of probabilistic statements in TBoxes, e.g. with probabilistic independence constraints, and investigate related complexity issues. We plan implementations of developed procedures and their optimizations.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. CUP (2003)
2. Charnes, A., Cooper, W.W.: Programming with linear fractional functionals. Naval Research Logistics Quarterly 9(3-4), 181–186 (1962)
3. Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006). pp. 237–249. IOS Press, Amsterdam, The Netherlands, The Netherlands (2006)
4. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence 46, 311–350 (1990)
5. Horrocks, I., Patel-Schneider, P., Van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Semantics 1(1), 7–26 (2003)
6. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8, 2000 (2000)
7. Jaeger, M.: Probabilistic role models and the guarded fragment. In: In Proceedings IPMU-2004. pp. 235–242 (2004)
8. Klinov, P., Parsia, B.: Pronto: A practical probabilistic description logic reasoner. In: Bobillo, F., da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW (LNCS Vol.). Lecture Notes in Computer Science, vol. 7123, pp. 59–79. Springer (2013)
9. Klinov, P., Parsia, B., Sattler, U.: On correspondences between probabilistic first-order and description logics. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
10. Koller, D., Levy, A., Pfeffer, A.: P-CLASSIC: A tractable probabilistic description logic. In: In Proceedings of AAAI-97. pp. 390–397 (1997)
11. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172(6-7), 852–883 (2008)
12. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
13. Lutz, C., Piro, R., Wolter, F.: Description logic tboxes: Model-theoretic characterizations and rewritability. In: Walsh, T. (ed.) IJCAI. pp. 983–988. IJCAI/AAAI (2011)
14. Lutz, C., Sattler, U., Tendera, L.: The complexity of finite model reasoning in description logics. Inf. Comput. 199(1-2), 132–171 (May 2005)
15. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR. AAAI Press (2010)
16. Näth, T.H., Möller, R.: ContraBovemRufum: A system for probabilistic lexicographic entailment. In: Description Logics (2008)
17. Price, C., Spackman, K.: SNOMED clinical terms. British Journal of Healthcare Computing and Information Management 17(3), 27–31 (2000)
18. Schrijver, A.: Theory of Linear and Integer Programming. No. 0471982326, 9780471982326, John Wiley & Sons (1998)
19. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. Artif. Intell. Res. (JAIR) 12, 199–217 (2000)
20. Yelland, P.M.: An alternative combination of bayesian networks and description logics. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) KR. pp. 225–234. Morgan Kaufmann (2000)

Optimizations for Decision Making and Planning in Description Logic Dynamic Knowledge Bases

Michele Stawowy

IMT Institute for Advanced Studies, Lucca, Italy
michele.stawowy@imtlucca.it

Abstract. Artifact-centric models for business processes recently raised a lot of attention, as they manage to combine structural (i.e. data related) with dynamical (i.e. process related) aspects in a seamless way. Many frameworks developed under this approach, although, are not built explicitly for planning, one of the most prominent operations related to business processes. In this paper, we try to overcome this by proposing a framework named Dynamic Knowledge Bases, aimed at describing rich business domains through Description Logic-based ontologies, and where a set of actions allows the system to evolve by modifying such ontologies. This framework, by offering action rewriting and knowledge partialization, represents a viable and formal environment to develop decision making and planning techniques for DL-based artifact-centric business domains.

1 Introduction

Classically, management of business processes always focused on workflows and the actions/interactions that take part in them, an approach called *process-centric*. One of the most prominent operations related to business processes is *planning* [7], namely finding a sequence of operations/actions that allows to reach a desired goal. Lately, such approach has been call into question, as the sole focus on the workflow leaves out the *informational context* in which the workflow is executed.

Artifact-centric models for business processes recently raised a lot of attention [2,6], as they manage to combine structural (i.e. data related) with dynamical (i.e. process related) aspects in a seamless way, thus overcoming the limits of process-centric approach. In this context, we can see the development of the framework called *Knowledge and Actions Bases* [9], the later higher formalization of it named *Description Logic Based Dynamic Systems* [5], and the Golog-based work of [1]. These works all share the same concept: handle the data-layer through a *Description Logic ontology*, while the process-layer, since DLs are only able to give a static representation of the domain of interest, is defined as *actions* that update the ontology (the so-called "functional view of knowledge bases" [10]). The combination of these two elements generates a transition system in which states are represented by DL knowledge bases. They do

also share a similar objective: verification of temporal formulas over the afore-mentioned transition system. Since finding a path that lead to a goal state can be expressed as a reachability temporal formula, these environments can be used for planning purposes, but they are not explicitly meant for this task. From their definition, we are limited to explore the state-space in a forward manner (we could end up having to explore the full state-space) and only by using the full body of the available knowledge, which is not ideal for developing different ways to search the state-space, as well as under a performance point of view.

In this paper we propose an artifact-centric framework, called *Dynamic Knowledge Bases*, aimed at describing data-rich business domains and be a more versatile environment for planning and decision-making: the data-layer is taken care of by a DL knowledge base, while a set of actions allows the system to evolve by adding/removing assertions, as well as introducing new instances to the system. To reach our goals, and overcome the afore-mentioned limitations, our framework relies on few optimizations. First of all, although our framework is based on Description Logic, it is desirable to skip completely the use of the TBox: this would allow us to avoid executing reasoning tasks and only work with facts from the ABox, simplifying especially the transition-building process. We fulfil this aspect with *action rewriting*, which rewrites actions and introduces a *blocking query*: such query (which is fixed for each action) tells if, given a state, we can perform the given action and built the ending state of the transition, or if the action will lead us to an inconsistent state w.r.t. the TBox. These operations are done without calculating the ending state, and without the need of the TBox (while keeping the consistency w.r.t. it).

Secondly, while the totality of the available knowledge is necessary to asses the consistency of the overall system, it bounds us to work with details that might not be of interests immediately. In decision making [8], "*an heuristic is a strategy that ignores part of the information, with the goal of making decisions more quickly, frugally, and/or accurately than more complex methods*". Being able to work with partial information is vital when we deal with systems described by complex ontologies and are composed of millions (if not more) instances. To allow our framework to be used for such strategies we introduce *partialization*, so that users can focus on a chosen subset of knowledge (partial knowledge); it allows to build a transition system which starts from a subset of the original ABox (the facts that describe the complete system), and, for each transition, choose which knowledge to transfer to the next state. Lastly, we demonstrate how, given a path found over the partial knowledge transition system, we can calculate a *global blocking query*, which tells if such path can be performed in the original transition system with no modifications.

The resulting framework constitutes a sound base on top of which researchers can develop new planning techniques useful for all those situations in which is necessary to manipulate both actions and data together (e.g. the decision making process in agents, composition of web services, etc.).

2 Dynamic Knowledge Bases

Dynamic Knowledge Bases (DKBs) are, briefly, a variation of *Knowledge and Action Bases* (KABs) [9], namely *dynamic systems* (more precisely labelled transition systems) in which states are constituted by DL *knowledge bases* (KBs), and a set of *actions* that makes the system evolve by modifying those KBs.

Definition 1. *A* DKB *is a tuple* $\mathcal{D} = (T, A_0, \Gamma)$*, where* (T, A_0) *is a DL-Lite*$_\mathcal{A}$ *KB, while* Γ *is a finite set of actions.*

We adopt a restricted version of DL-Lite$_\mathcal{A}$ knowledge bases [4], which does not use attributes (available in full DL-Lite$_\mathcal{A}$ KBs). DL-Lite$_\mathcal{A}$ employs the *Unique Name Assumption*, thus equality assertions are not allowed. We adopt DL-Lite$_\mathcal{A}$ as it is, like other DL-Lite dialects, quite expressive while maintaining decidability, good complexity results, and enjoys the FOL-rewritability property. In the followings, the set $\textsc{adom}(A)$ identifies the individual constants in the ABox A, which are defined over a countably infinite (object) universe Δ of individuals (it follows that $\textsc{adom}(A) \subseteq \Delta$). $\mathcal{A}_T$ denotes the set of all possible consistent ABoxes w.r.t. T that can be constructed using atomic concept and atomic role names in T, and individuals in Δ. The adopted semantic is the standard one based on first-order interpretations and on the notion of model: a TBox is satisfiable if admits at least one model, an ABox A is consistent w.r.t. a TBox T if (T, A) is satisfiable, and (T, A) logically implies an ABox assertion α (denoted $(T, A) \models \alpha$) if every model of (T, A) is also a model of α.

We define an *action* as:

$$\mathsf{a}\colon q, N \rightsquigarrow E$$

where a is the *action name*, q is a query called *action guard*, N is a set of variables which are used in an *instance creation function*, and E are the *action effects*.
The guard q is a standard *conjunctive query* (CQ) of the type $q = \exists \overrightarrow{y}.conj(\overrightarrow{x}, \overrightarrow{y})$, where $conj(\overrightarrow{x}, \overrightarrow{y})$ is a conjunction of atoms using free variables $\overrightarrow{x}$ and existentially quantified variables $\overrightarrow{y}$, no individuals. Atoms of q uses concepts and roles found in T. $Vars(q)$ represents the variables in q (i.e., $\overrightarrow{x} \cup \overrightarrow{y}$), while $Vars(q)_{\not\exists}$ (resp., $Vars(q)_\exists$) only the set $\overrightarrow{x}$ (resp., $\overrightarrow{y}$).
The set N contains variables which do not appear in q (i.e., $Vars(q) \cap N = \emptyset$), and which are fed to an assignment function m when the action is executed. The set E is a set of atomic effects (i.e., atomic non-grounded ABox assertions) which is divided in two subsets: the set E^- of *negative effects*, and the set E^+ of *positive effects*. All atoms of E^- must use variables that are in $Vars(q)_{\not\exists}$, while the atoms of E^+ uses variables from the set $Vars(q)_{\not\exists} \cup N$. All variables are defined over a countably infinite (object) universe V of variables.

Definition 2. *The transition system* $\Upsilon_\mathcal{D}$ *is defined as a tuple* $(\Delta, T, \Sigma, A_0, \Rightarrow)$*, where: (i)* Δ *is the universe of individual constants; (ii)* T *is a TBox; (iii)* Σ *is a set of states, namely ABoxes from the set* $\mathcal{A}_T$ *(*$\Sigma \subseteq \mathcal{A}_T$*); (iv)* A_0 *is the initial state; (v)* $\Rightarrow \ \subseteq \Sigma \times \mathcal{L} \times \Sigma$ *is a labelled transition relation between states, where* $\mathcal{L} = \Gamma \times \Theta$ *is the set of labels containing an action instantiation* $\mathsf{a}\vartheta$*, where* a *is an action from* Γ *and* ϑ *a variable assignment in* Θ *from* V *to* Δ*.*

The *transition system* $\Upsilon_{\mathcal{D}}$ represent the dynamics of a DKB $\mathcal{D}$. Given a state A and selected an action a, the informal semantic of a transition is:

1. extract the certain answers $\text{ANS}(q, T, A)$ of the guard q from the state A;
2. pick *randomly* one tuple from $\text{ANS}(q, T, A)$ and use it to initiate the variable assignment ϑ_{a} for the variables $Vars(\mathsf{a})$ (at this point we covered only the free variables in $Vars(q)_{\not\exists}$);
3. choose an assignment for the variables in N and use it to extend ϑ_{a}. We define an assignment function $m(N, A) : N \to (\Delta \setminus \text{ADOM}(A))$, which assigns to each variable of N an individual from Δ which does not appear in A;
4. use ϑ_{a} to instantiate the effects E and calculate A_{next} by applying the instantiated effects to A.

The sets Σ and $\Rightarrow$ are thus mutually defined using induction (starting from A_0) as the smallest sets satisfying the following property: for every $A \in \Sigma$ and action $\mathsf{a} \in \Gamma$, if exists an action instantiation $\mathsf{a}\vartheta_{\mathsf{a}}$ s.t.

$$A_{next} = A \setminus sub(ent(E^-, T)\vartheta_{\mathsf{a}}, A) \cup E^+\vartheta_{\mathsf{a}}$$

and $A_{next} \in \mathcal{A}_T$, then $A_{next} \in \Sigma$ and $A \overset{l}{\Rightarrow} A_{next}$, with $l = \mathsf{a}\vartheta_{\mathsf{a}}$. $\mathsf{a}\vartheta_{\mathsf{a}}$ is called an *instantiation* of a.

$ent(E^-, T)$ represents a set of atoms derived from E^-, which represents all the atoms which entail one or more single negative effects e^- in E^- w.r.t. to the TBox T. We take each single negative effect e^- and, by considering e^- as a CQ composed only by one atom, obtain an UCQ $rew_T(e^-)$ by using the *query reformulation algorithm* [3, Chapter 5.2]. Since we consider a single atom at a time, the algorithm produces an UCQ composed only by CQs with a single atom e^-_{rew} in them. Each atom e^-_{rew} either contains variables found in e^- or, in case of a role term, one of the two variables can be a non-distinguished non-shared variable represented by the symbol '_' (never both variables). We add each atom e^-_{rew} to the set $ent(E^-, T)$. Given $ent(E^-, T)$, we calculate the set $sub(ent(E^-, T)\vartheta_{\mathsf{a}}, A)$ in the following way. For each atom e^-_{rew} in $ent(E^-, T)$, we apply the variable transformation ϑ_{a} to it (the symbol '_' remains untouched, as it is not linked to any variable that appears in ϑ_{a}); we then check if it exists in the ABox A an assertion α such that $e^-_{rew}\vartheta_{\mathsf{a}} = \alpha$, assuming that the symbol '_' can be evaluated equal to any individual ($\_ = ind\ , \forall ind \in \text{ADOM}(A)$).

For clarity, from now on we will denote the set $sub(ent(E^-, T)\vartheta_{\mathsf{a}}, A)$ with $E^-_{sub(\vartheta_{\mathsf{a}})}$. Notice that the set $E^-_{sub(\vartheta_{\mathsf{a}})}$ is not uniquely determined, as it depends on the ABox on which it is applied. This behaviour is intentional, as our aim is to have the certainty that an assertion e^- marked for removal will not appear in the next state nor in the ABox A_{next}, nor as an inferable assertion ($\langle T, A_{next}\rangle \not\models e^-$); to reach such goal, we have to remove all possible assertions that entail e^-. The set $ent(E^-, T)$, instead, depends only on E^- and T, thus it's constant and can be calculated only one time at the beginning.

As we see from the definition of A_{next}, actions modify only ABox assertions: it follows that the TBox is fixed, while the ABox changes as the system evolves (thus an ABox A_i is sufficient to identify the state i of the system). The transition system $\Upsilon_{\mathcal{D}}$ clearly can be infinite, as we have the possibility to introduce new

constants. We call a *path* π a (possibly infinite) sequence of transitions over $\Upsilon_{\mathcal{D}}$ that start from A_0 ($\pi = A_0 \overset{\mathsf{a}_1\vartheta_1}{\Rightarrow} \ldots \overset{\mathsf{a}_n\vartheta_n}{\Rightarrow} A_n$).

Example 1. Consider the DKB $\mathcal{D}$ described by the following elements and which models a simple business scenario:

- the TBox $T = \{\mathsf{Employee} \sqsubseteq \neg\mathsf{Product}, \mathsf{Technician} \sqsubseteq \mathsf{Employee}\}$;
- the ABox $A_0 = \{\mathsf{Technician(t1)}, \mathsf{Product(p1)}\}$;
- the action set Γ composed of the following actions:
  create: $\{\mathsf{Employee}(x)\}, \{y\} \rightsquigarrow \{\mathsf{Product}(y)\}^+$
  fire: $\{\mathsf{Employee}(x)\} \rightsquigarrow \{\mathsf{Employee}(x)\}^-$

If we consider A_0 as the initial state in $\Upsilon_{\mathcal{D}}$, then a possible transition is $A_0 \overset{\mathsf{create}\vartheta}{\Rightarrow} A_1$ where: $\vartheta = \{x \mapsto \mathsf{t1}, y \mapsto \mathsf{p2}\}$ (notice that we introduce a new individual p2), and $A_1 = \{\mathsf{Technician(t1)}, \mathsf{Product(p1)}, \mathsf{Product(p2)}\}$.

We could also perform the action fire, as it exists a proper instantiation of it by using the variable assignment $\vartheta_{\mathsf{fire}} = \{x \mapsto \mathsf{t1}\}$. The set $ent(E^-, T)$ for the action fire corresponds to the set $\{\mathsf{Employee}(x), \mathsf{Technician}(x)\}$, thus $E^-_{sub(\vartheta_{\mathsf{fire}})}$ would be equal to $\{\mathsf{Technician(t1)}\}$. Performing the action instantiation would get us to the state $A_2\{\mathsf{Product(p1)}\}$, and it's clear that $\langle T, A_2\rangle \not\models \mathsf{Employee(t1)}$. If we would simply remove the instantiated negative effects in $E^-\vartheta_{\mathsf{fire}}$), we wouldn't achieve the same result (as the assertion Technician(t1) would still appear in the final state), as if the action didn't have any effect at all.

3 Optimizations

3.1 Action Rewriting

The first optimization we bring to the framework regards actions, and, more specifically, the guard q. Using the *query reformulation algorithm* [3, Chapter 5.2], we can transform a query q into an UCQ $rew_T(q)$ such that $\textsc{ans}(q, T, a) = \textsc{ans}(rew_T(q), \emptyset, A)$. We then take every action a, calculate $rew_T(q)$, and, for every CQ $q^{rew} \in rew_T(q)$, create an action $\mathsf{a}^{\mathsf{rew}}$: $q^{rew}, N \rightsquigarrow E$ (with N and E taken from a without modifications). These new actions slightly modify the transition function $\Rightarrow$: the guard is now evaluated without using the TBox, and the variable assignment $\vartheta_{\mathsf{a}^{\mathsf{rew}}}$ must be taken from the certain answers $\textsc{ans}(q^{rew}, \emptyset, A)$, while the rest of the transition function remains the same.

The second optimization regards the ending state of the transition: in the specification of a DKB, actions could lead to inconsistent states. We introduce an additional element called *blocking query* B, a boolean UCQ used as a block test in the state A before performing the action: if B returns *false*, then we can perform the action and have the guarantee that the ending state A_{next} is consistent w.r.t. T. The building of B is based on the *NI-closure of* T (denoted $cln(T)$) defined in [3]. Each positive effect $e^+ \in E^+$ (column 1 in Table 1, we need to change the variables accordingly to the ones in e^+) could take part in a negative inclusion assertion $\alpha \in cln(T)$ (column 2 in Table 1); this mean that we have to look for a possible assertion β (column 3 in Table 1) which

could break α when e^+ is added ($\mathbf{z}$ represents a newly introduced variable, thus $\mathbf{z} \notin \mathit{Vars}(q) \cup N \cup \mathit{Vars}(B)$). To do so, for each possible β we get from e^+ and α, we perform the following steps (we start from $B = \bot$, where $\bot$ indicates a predicate whose evaluation is *false* in every interpretation):

1. we check if β is present in the positive effects E^+ by executing $\text{ANS}(\beta, \emptyset, E^+)$ and retrieve all the certain answers ϕ_{E^+}. For each ϕ_{E^+}, it means it exist an assertion $\beta\phi_{E^+}$ which poses a problem. Since we are dealing with variables (the effects are not instantiated yet), we have to express in B under which conditions $\beta\phi_{E^+}$ would make A_{next} inconsistent; we do this by adding the corresponding CQ β_{E^+} (column 4 in Table 1) to B by *or*-connecting it to the rest of the CQs.
   Notice that we treat $\mathbf{z}$ as an existential variable, as it does not appear in e^+ and thus we have no constrains about it.
2. we check if in E^- there are negative effects that could block β by removing it (thus eliminating the threat of an inconsistency). by executing $\text{ANS}(\beta, \emptyset, \mathit{ent}(E^-, T))$ and retrieve all the certain answers ϑ_{E^-}. For each ϑ_{E^-}, it means it exist an assertion $\beta\phi_{E^-}$ which is removed. Since we are dealing with variables (the effects are not instantiated yet), we have to express in B under which conditions $\beta\phi_{E^-}$ can't block an inconsistency in A_{next}; we do this by adding the corresponding UCQ β_{E^-} (column 5 in Table 1) to B by *or*-connecting it to the rest of the CQs.
3. if E^- can't block any inconsistency (thus $\text{ANS}(\beta, \emptyset, \mathit{ent}(E^-, T)) = \emptyset$), then we have to express in B under which conditions there will be a inconsistency in A_{next} due to an assertion β in A w.r.t e^+; we do so by adding β_A (column 6 in Table 1) to B by *or*-connecting it to the rest of the CQs.

Note that while building the blocking query B, we could have, for the UCQs β_{E^-}, inequalities of the type $x \neq \_$, with $\_$ the non-distinguished non-shared variable generated by $\mathit{ent}(E^-, T)$. Such inequalities always evaluate to *False*.

Definition 3. *Given an action* $\mathsf{a} \in \Gamma$*, its* rewritten action $\mathsf{a}^{\mathsf{rew}}$ *is defined as:*

$$\mathsf{a}^{\mathsf{rew}}\text{: } q^{rew}, N, B \rightsquigarrow E$$

where $q^{rew} \in \mathit{rew}_T(q)$*, and* B *is the* blocking query *of* $\mathsf{a}^{\mathsf{rew}}$*.*

The union of all possible rewritten actions defines the set of actions Γ^{rew}.

Example 2. Let's consider the action create: $\{\mathsf{Employee}(x)\}, \{y\} \rightsquigarrow \{\mathsf{Product}(y)\}^+$. First we calculate $\mathit{rew}_T(q)$, which is the UCQ $\mathsf{Employee}(x) \vee \mathsf{Technician}(x)$.
We can now calculate the blocking query B. We see that the concept term Product of the positive effect $e^+ = \mathsf{Product}(y)$ takes part in the negative-inclusion assertion $\mathsf{Employee} \sqsubseteq \neg\mathsf{Product}$, and, by the definition of $\mathit{cln}(T)$, also in the assertion $\mathsf{Technician} \sqsubseteq \neg\mathsf{Product}$: we thus have two β assertions, $\mathsf{Employee}(y)$, and $\mathsf{Technician}(y)$. By following the procedure for building B, we have no β_{E^+} elements (as $\text{ANS}(\beta, \emptyset, E^+) = \emptyset$, and no β_{E^-} elements (as $\text{ANS}(\beta, \emptyset, \mathit{ent}(E^-, T)) = \emptyset$). The final query is thus composed only of β_A elements, and is

$$B = \mathsf{Employee}(y) \vee \mathsf{Technician}(y)$$

e^+	α	β	$\vartheta_{E^+} \to \beta_{E^+}$	$\vartheta_{E^-} \to \beta_{E^-}$	β_A
$\mathsf{A}(x)$	$\mathsf{A} \sqsubseteq \neg \mathsf{A}_1$ $\mathsf{A}_1 \sqsubseteq \neg \mathsf{A}$	$\mathsf{A}_1(x)$	$\{x \mapsto y\} \to x = y$	$\{x \mapsto y\} \to x \neq y \wedge \mathsf{A}_1(x)$	$\mathsf{A}_1(x)$
$\mathsf{A}(x)$	$\mathsf{A} \sqsubseteq \neg \exists \mathsf{P}$ $\exists \mathsf{P} \sqsubseteq \neg \mathsf{A}$	$\mathsf{P}(x, \mathbf{z})$	$\{x \mapsto y\} \to x = y$	$\{x \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(x, \mathbf{z}) \wedge x \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}(x, \mathbf{z})$
$\mathsf{A}(x)$	$\mathsf{A} \sqsubseteq \neg \exists \mathsf{P}^-$ $\exists \mathsf{P}^- \sqsubseteq \neg \mathsf{A}$	$\mathsf{P}(\mathbf{z}, x)$	$\{x \mapsto y\} \to x = y$	$\{x \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x) \wedge x \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x)$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P} \sqsubseteq \neg \mathsf{A}$ $\mathsf{A} \sqsubseteq \neg \exists \mathsf{P}$	$\mathsf{A}(x_1)$	$\{x_1 \mapsto y\} \to x_1 = y$	$\{x_1 \mapsto y\} \to x_1 \neq y \wedge \mathsf{A}(x_1)$	$\mathsf{A}(x_1)$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P}^- \sqsubseteq \neg \mathsf{A}$ $\mathsf{A} \sqsubseteq \neg \exists \mathsf{P}^-$	$\mathsf{A}(x_2)$	$\{x_2 \mapsto y\} \to x_2 = y$	$\{x_2 \mapsto y\} \to x_2 \neq y \wedge \mathsf{A}(x_2)$	$\mathsf{A}(x_2)$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P} \sqsubseteq \neg \exists \mathsf{P}_1$ $\exists \mathsf{P}_1 \sqsubseteq \neg \exists \mathsf{P}$	$\mathsf{P}_1(x_1, \mathbf{z})$	$\{x_1 \mapsto y\} \to x_1 = y$	$\{x_1 \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(x_1, \mathbf{z}) \wedge x_1 \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}_1(x_1, \mathbf{z})$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P}^- \sqsubseteq \neg \exists {\mathsf{P}_1}^-$ $\exists {\mathsf{P}_1}^- \sqsubseteq \neg \exists \mathsf{P}^-$	$\mathsf{P}_1(\mathbf{z}, x_2)$	$\{x_2 \mapsto y\} \to x_2 = y$	$\{x_2 \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x_2) \wedge x_2 \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}_1(\mathbf{z}, x_2)$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P} \sqsubseteq \neg \exists {\mathsf{P}_1}^-$ $\exists {\mathsf{P}_1}^- \sqsubseteq \neg \exists \mathsf{P}$	$\mathsf{P}_1(\mathbf{z}, x_1)$	$\{x_1 \mapsto y\} \to x_1 = y$	$\{x_1 \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x_1) \wedge x_1 \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}_1(\mathbf{z}, x_1)$
$\mathsf{P}(x_1, x_2)$	$\exists \mathsf{P}^- \sqsubseteq \neg \exists \mathsf{P}_1$ $\exists \mathsf{P}_1 \sqsubseteq \neg \exists \mathsf{P}^-$	$\mathsf{P}_1(x_2, \mathbf{z})$	$\{x_2 \mapsto y\} \to x_2 = y$	$\{x_2 \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $\exists \mathbf{z}.\mathsf{P}(x_2, \mathbf{z}) \wedge x_2 \neq y_1 \wedge \mathbf{z} \neq y_2$	$\exists \mathbf{z}.\mathsf{P}_1(x_2, \mathbf{z})$
$\mathsf{P}(x_1, x_2)$	$\mathsf{P} \sqsubseteq \neg \mathsf{P}_1$ $\mathsf{P}_1 \sqsubseteq \neg \mathsf{P}$ $\mathsf{P}^- \sqsubseteq \neg {\mathsf{P}_1}^-$ ${\mathsf{P}_1}^- \sqsubseteq \neg \mathsf{P}^-$	$\mathsf{P}_1(x_1, x_2)$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\}$ $\to x_1 = y_1 \wedge x_2 = y_2$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\} \to$ $(x_1 \neq y_1 \wedge \mathsf{P}(x_1, x_2)) \vee$ $(x_2 \neq y_2 \wedge \mathsf{P}(x_1, x_2))$	$\mathsf{P}_1(x_1, x_2)$
$\mathsf{P}(x_1, x_2)$	$\mathsf{P} \sqsubseteq \neg {\mathsf{P}_1}^-$ ${\mathsf{P}_1}^- \sqsubseteq \neg \mathsf{P}$ $\mathsf{P}^- \sqsubseteq \neg \mathsf{P}_1$ $\mathsf{P}_1 \sqsubseteq \neg \mathsf{P}^-$	$\mathsf{P}_1(x_2, x_1)$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\}$ $\to x_1 = y_1 \wedge x_2 = y_2$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\} \to$ $(x_1 \neq y_1 \wedge \mathsf{P}(x_2, x_1)) \vee$ $(x_2 \neq y_2 \wedge \mathsf{P}(x_2, x_1))$	$\mathsf{P}_1(x_2, x_1)$
$\mathsf{P}(x_1, x_2)$	funct P	$\mathsf{P}(x_1, \mathbf{z})$ $\wedge\ x_2 \neq \mathbf{z}$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\}$ $\to x_1 = y_1 \wedge x_2 \neq y_2$	$\{x_1 \mapsto y_1, \mathbf{z} \mapsto y_2\} \to$ $(\exists \mathbf{z}.\mathsf{P}(x_1, \mathbf{z}) \wedge x_1 \neq y_1 \wedge \mathbf{z} \neq x_2) \vee$ $(\exists \mathbf{z}.\mathsf{P}(x_1, \mathbf{z}) \wedge x_1 = y_1 \wedge \mathbf{z} \neq x_2 \wedge \mathbf{z} \neq y_2)$	$\exists \mathbf{z}.\mathsf{P}(x_1, \mathbf{z})$ $\wedge\ x_2 \neq \mathbf{z}$
$\mathsf{P}(x_1, x_2)$	funct P^-	$\mathsf{P}(\mathbf{z}, x_2)$ $\wedge\ x_1 \neq \mathbf{z}$	$\{x_1 \mapsto y_1, x_2 \mapsto y_2\}$ $\to x_2 = y_2 \wedge x_1 \neq y_1$	$\{\mathbf{z} \mapsto y_1, x_2 \mapsto y_2\} \to$ $(\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x_2) \wedge x_2 \neq y_2 \wedge \mathbf{z} \neq x_1) \vee$ $(\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x_2) \wedge x_2 = y_2 \wedge \mathbf{z} \neq x_1 \wedge \mathbf{z} \neq y_1)$	$\exists \mathbf{z}.\mathsf{P}(\mathbf{z}, x_2)$ $\wedge\ x_1 \neq \mathbf{z}$

Table 1: Assertions β_{E^+}, β_{E^-}, and β_A for a given positive effect e^+ and assertion α

We get the following two rewritten actions:
$\mathsf{create}_1^{\mathsf{rew}}$: $\{\mathsf{Employee}(x)\}, \{y\}, \{\mathsf{Employee}(y) \vee \mathsf{Technician}(y)\} \rightsquigarrow \{\mathsf{Product}(y)\}^+$
$\mathsf{create}_2^{\mathsf{rew}}$: $\{\mathsf{Technician}(x)\}, \{y\}, \{\mathsf{Employee}(y) \vee \mathsf{Technician}(y)\} \rightsquigarrow \{\mathsf{Product}(y)\}^+$

Theorem 1. *Given a satisfiable KB (T, A), an action $\mathsf{a}^{\mathsf{rew}} \in \Gamma^{rew}$ such that $\vartheta_{\mathsf{a}^{\mathsf{rew}}} \in \text{ANS}(q^{rew}, \emptyset, A)$ and $\text{ANS}(B\vartheta_{\mathsf{a}^{\mathsf{rew}}}, \emptyset, A) = \emptyset$, then the ABox $A_{next} = A \setminus E^-_{sub(\vartheta_{\mathsf{a}^{\mathsf{rew}}})} \cup E^+\vartheta_{\mathsf{a}^{\mathsf{rew}}}$ is consistent w.r.t. T.*

Proof. For the proof of the theorem we remind the reader to the Appendix of the extended version of this paper [11].

Lemma 1. *Given an action $\mathsf{a}^{\mathsf{rew}} \in \Gamma^{rew}$, for every ABox A such that $\vartheta_{\mathsf{a}^{\mathsf{rew}}} \in \text{ANS}(q^{rew}, \emptyset, A)$ and $\text{ANS}(B\vartheta_{\mathsf{a}^{\mathsf{rew}}}, \emptyset, A) = \emptyset$, we can always perform the transition $A \overset{\mathsf{a}^{\mathsf{rew}}\vartheta_{\mathsf{a}^{\mathsf{rew}}}}{\Rightarrow} A_{next}$, with $A_{next} \in \mathcal{A}_T$.*

Thanks to the rewriting of actions, we can build the transition system $\Upsilon_{\mathcal{D}}$ without the need of the TBox T, while still having the guarantee that the system is consistent w.r.t. it.

3.2 Partial Transition System

We now build a *partialization* $\Upsilon^p_{\mathcal{D}}$ of the transition system $\Upsilon_{\mathcal{D}}$, which is built in the same way as $\Upsilon_{\mathcal{D}}$, apart from two points: i) the initial state is a subset of the ABox A_0 ii) it uses a looser transition function.

Definition 4. *A* partial transition system $\Upsilon^p_{\mathcal{D}}$ *is a tuple $(\Delta, T, \Sigma^p, A^p_0, \rightarrow)$, where: (i) Δ is the universe of individual constants; (ii) T is a TBox; (iii) Σ^p is a set of states, namely ABoxes from the set $\mathcal{A}_T$ ($\Sigma^p \subseteq \mathcal{A}_T$); (iv) A^p_0 is a subset of the initial ABox A_0 ($A^p_0 \subseteq A_0$); (v) $\rightarrow \ \subseteq \Sigma^p \times \mathcal{L} \times \Sigma^p$ is a labelled transition relation between states, where $\mathcal{L} = \Gamma^{rew} \times \Theta$ is the set of labels containing an action instantiation $\mathsf{a}^{\mathsf{rew}}\vartheta$, where $\mathsf{a}^{\mathsf{rew}}$ is an action from Γ^{rew} and ϑ a variable assignment in Θ from V to Δ.*

As $A^p_0 \subseteq A_0$, we have the guarantee that $A^p_0 \in \mathcal{A}_T$. The sets Σ^p and $\rightarrow$ are mutually defined using induction (starting from A^p_0) as the smallest sets satisfying the following property: for every $A^p \in \Sigma^p$ and action $\mathsf{a}^{\mathsf{rew}} \in \Gamma^{rew}$, if exists an action instantiation $\mathsf{a}^{\mathsf{rew}}\vartheta_{\mathsf{a}^{\mathsf{rew}}}$ s.t.

$$A^p_{next} \subseteq A^p \setminus E^-_{sub(\vartheta_{\mathsf{a}^{\mathsf{rew}}})} \cup E^+\vartheta_{\mathsf{a}^{\mathsf{rew}}}$$

and $A^p_{next} \in \mathcal{A}_T$, then $A^p_{next} \in \Sigma^p$ and $A^p \overset{l}{\Rightarrow} A^p_{next}$, with $l = \mathsf{a}^{\mathsf{rew}}\vartheta_{\mathsf{a}^{\mathsf{rew}}}$.

Notice that A^p_{next} can be any subset of $A^p \setminus E^-_{sub(\vartheta_{\mathsf{a}^{\mathsf{rew}}})} \cup E^+\vartheta_{\mathsf{a}^{\mathsf{rew}}}$, thus allowing to select which knowledge to focus on, unlike in $\Upsilon_{\mathcal{D}}$ where we transfer all the knowledge from one state to another. We now define the existing relation between the the partial transition system $\Upsilon^p_{\mathcal{D}}$ and the transition system $\Upsilon_{\mathcal{D}}$. Given a path π^p in $\Upsilon^p_{\mathcal{D}}$, we say that π^p is a *proper partialization* of a path π in $\Upsilon_{\mathcal{D}}$ (resp., π is a *proper completion* of π^p) if:
– each state A^p_i is a subset of the relative state A_i ($A^p_i \subseteq A_i$);

– each transition is caused by the same action $\mathsf{a}_\mathsf{i}^{\mathsf{rew}}$ and the related variable assignments are equal ($\vartheta_i^p = \vartheta_i$).

Between $\Upsilon_\mathcal{D}$ and $\Upsilon_\mathcal{D}^p$ there is no relation such as bisimulation or even simulation; this is a clear (and intended) consequence of working with partial knowledge. This also means that we have no immediate way to know if, given a partial path π^p in $\Upsilon_\mathcal{D}^p$, we can use the same actions instantiations in $\Upsilon_\mathcal{D}$, and thus if it exists a path π that is a proper completion π^p. To overcome this problem, we extend the definition of the blocking query B by creating a *global blocking query B_{π^p} w.r.t to a finite partial path π^p*. B_{π^p} is a boolean UCQ that can be evaluated in the complete initial state A_0, and, if it is evaluated *False*, gives us the certainty that we can use the same actions instantiations found in π^p starting from A_0 without generating any inconsistent state w.r.t. T.

B_{π^p} is built by iteratively adding the single instantiated blocking queries $B_i\vartheta_i^p$ of the actions that compose π^p (Algorithm 1, the symbol $\top$ indicates a predicate whose evaluation is *true* in every interpretation). At each step, before adding the i-th instantiated blocking query $B_i\vartheta_i^p$ to B_{π^p}, we perform the following operations:

– check that $\text{ANS}(B_{\pi^p}, \emptyset, E_i^+\vartheta_i^p)$ is *False*;
– remove any CQ β in B_{π^p} that evaluates always *False* (i.e., contains (in)equalities that evaluates always to *False*, like $\mathsf{ind_i} = \mathsf{ind_l}$, or $\mathsf{ind_i} \neq \mathsf{ind_i}$);
– remove from each CQ the (in)equalities that evaluates always to *True*, as they do not influence the ending result. We are sure that no CQ will be left empty, because it would mean the whole CQ would always evaluate to *True*, and this would have blocked the first step;
– for each CQ β, generate a temporary CQ β_{temp} by removing all the (in)equalities and transform existential variables in free ones. Looking at how the blocking query is built, we have that β_{temp} is either empty (β is composed only of (in)equalities) or contains only one atomic assertion with at most one free variable. For example, if $\beta = \exists z.\mathsf{P}(\mathsf{i}_1, z) \wedge \mathsf{i}_2 \neq z$, then $\beta_{temp} = \mathsf{P}(\mathsf{i}_1, z)$;
– perform $\text{ANS}(\beta_{temp}, \emptyset, E^-_{sub(\vartheta_i^p)})$:
  - if it evaluates to *True*, then it means that the instantiated negative effects $E^-_{sub(\vartheta_i^p)}$ remove the atom β_{temp}, and in this case we can remove the CQ β from B_{π^p};
  - if it returns answers of the type $\vartheta_{\beta_{temp}} = \{\mathsf{z} \mapsto \mathsf{ind}\}$, then it means that the instantiated negative effects $E^-_{sub(\vartheta_i^p)}$ remove the atom β_{temp} only if z is mapped to the individual ind. We thus add to the CQ β the inequality $z \neq \mathsf{ind}$.

Theorem 2. *Given a DKB $\mathcal{D}$, a finite partial path π^p, and its global blocking query B_{π^p}, if $\text{ANS}(B_{\pi^p}, \emptyset, A_0) = \emptyset$, then it exists a concretion π of π^p such that $\pi \in \Upsilon_\mathcal{D}$.*

Proof. For the proof of the theorem we remind the reader to the Appendix of the extended version of this paper [11].

Example 3. Consider the DKB $\mathcal{D}$ described by the following elements and which models a simple business scenario:

Algorithm 1: The algorithm to build the global blocking query B_{π^p}

input : A partial path π^p
output: An UCQ B_{π^p}

$B_{\pi^p} := \{\bot\}$
$i :=$ n. of transitions in π^p `// counter variable`
while $i > 0$ **do** `// each cycle refers to transition` $A^p_{i-1} \xrightarrow{\mathsf{a}_i\vartheta^p_i} A^p_i$
  if $\text{ANS}(B_{\pi^p}, \emptyset, E^+_i\vartheta^p_i) \neq \emptyset$ **then**
    $B_{\pi^p} := \top$ `// inconsistency in the i-th transition`
    break
  end
  foreach $\beta \in B_{\pi^p}$ **do**
    if β *contains (in)equalities that are always False* **then**
      $B_{\pi^p} := B_{\pi^p} \setminus \beta$ `// remove CQs that are always` *`False`*
    end
    remove from β (in)equalities that are always *True*
    $\beta_{temp} := \beta$ without (in)equalities and existential operator
    if $\text{ANS}(\beta_{temp}, \emptyset, E^-_{sub(\vartheta^p_i)}) = True$ **then**
      $B_{\pi^p} := B_{\pi^p} \setminus \beta$ `//` $E^-_{sub(\vartheta^p_i)}$ `erases the CQ` β_{temp}
    else if $\text{ANS}(\beta_{temp}, \emptyset, E^-_{sub(\vartheta^p_i)}) \neq \emptyset$ **then**
      foreach $\vartheta_{\beta_{temp}} = \{\boldsymbol{z} \mapsto \mathsf{ind}\} \in \text{ANS}(\beta_{temp}, \emptyset, E^-_{sub(\vartheta^p_i)})$ **do**
        $\beta := \beta \wedge \mathbf{z} \neq \mathsf{ind}$ `// update the CQ` β
      end
    end
  end
  $B_{\pi^p} := B_{\pi^p} \cup B_i\vartheta^p_i$ `// add the blocking query of action` a_i
  $i := i - 1$
end

- the TBox $T = \{\mathsf{Stored} \sqsubseteq \neg\mathsf{Shipped}\}$;
- the ABox $A_0 = \{\mathsf{Product}(\mathsf{p1}), \mathsf{Stored}(\mathsf{p1}), \mathsf{Product}(\mathsf{p2})\}$;
- the action set Γ composed by the following actions:
  pack: $\{\mathsf{Product}(x)\} \rightsquigarrow \{\mathsf{Packed}(x)\}^+$,
  ship: $\{\mathsf{Packed}(x)\} \rightsquigarrow \{\mathsf{Shipped}(x)\}^+$
  which becomes the set Γ^{rew} composed of the actions:
  $\mathsf{pack}^{\mathsf{rew}}$: $\{\mathsf{Product}(x)\} \rightsquigarrow \{\mathsf{Packed}(x)\}^+$,
  $\mathsf{ship}^{\mathsf{rew}}$: $\{\mathsf{Packed}(x)\}, \{\mathsf{Stored}(x)\} \rightsquigarrow \{\mathsf{Shipped}(x)\}^+$

At this point, we develop a partial transition system $\widehat{\Upsilon}_{\mathcal{D}}$ by considering the partial initial state $A^p_0 = \{\mathsf{Product}(\mathsf{p1})\}$. We can perform the sequence of transitions $\pi^p = A^p_0 \xrightarrow{\mathsf{pack}\vartheta} A^p_1 \xrightarrow{\mathsf{ship}\vartheta} A^p_2$, where: $\vartheta = \{x \mapsto \mathsf{p1}\}$, $A^p_1 = \{\mathsf{Packed}(\mathsf{p1})\}$, and $A^p_2 = \{\mathsf{Shipped}(\mathsf{p1})\}$. The global blocking query B_{π^p} is $\mathsf{Stored}(\mathsf{p1})$, and we see that, if we try to transpose π^p in the original ABox A_0, we have $\text{ANS}(B_{\pi^p}, \emptyset, A_0) \neq \emptyset$, thus meaning that π^p doesn't have a proper concretion π (indeed if we perform the two actions, we would end up having an inconsistent state A_2).

If we would consider instead the partial initial state $A_0^p = \{\mathsf{Product(p2)}\}$, instead, we woould be able to find a proper completion of π^p, as B_{π^p} would be $\mathsf{Stored(p2)}$ and $\text{ANS}(B_{\pi^p}, \emptyset, A_0) = \emptyset$.

Given a finite partial path π^p and its global blocking query B_{π^p}, we have a way to know if we can transform π^p into a complete path π without actually calculating it, only by performing an UCQ over the initial state A_0. Notice also that this result can be applied to all possible ABoxes, not only A_0; as long as A_0^p is contained in an ABox A, and $\text{ANS}(B_{\pi^p}, \emptyset, A) = \emptyset$, then it exists a path π which starts from A and is a proper concretion of π^p.

4 Conclusions

In this paper we formalize a framework, called Dynamic Knowledge Bases, aimed at modelling the dynamics of artifact-centric business processes. Such framework is represented by a transition system where states are defined by DL-Lite$_\mathcal{A}$ knowledge bases, and where a set of actions allows the system to evolve by adding or removing assertions, along with the possibility to introduce new instances. The expressive power and reasoning services of Description Logics are very helpful to describe and manage the domain knowledge, but constitute a difficult environment to deal with when it comes to the dynamics of the processes. To tackle this problem, we introduce two optimizations, namely action rewriting and the partialization of the transition system related to a Dynamic Knowledge Base: these optimizations give us a framework where we can work with partial knowledge and where the TBox is not needed, still guaranteeing that the resulting system is consistent with it. Given a path valid for the partial transition system, we can calculate its global blocking query, and know if it can be transferred to the complete transition system without any change, and without the need to do any other calculation.

Our work does not aim to propose a planning technique, neither try to give a solution w.r.t. the decidability/undecidability problem of plan research in our environment (since it is possible to generate an infinite transition system), but to create a framework that can be used as a formal domain-independent base to develop planning and decision making techniques for data-rich business domains by taking full advantage of the DL-Lite reasoning power.

We are currently working to further expand this framework in various directions. Under the theoretical side, we are already developing an abstraction of the transition system, in particular by expressing the needed knowledge by using only queries, which can be then used over the complete transition system. Under the practical side, we intend to propose a backward planning algorithm, which takes advantage of the abstract transition system and the possibility to work with partial knowledge to return all plans of interest w.r.t. a goal.

Although further investigation is surely needed, Dynamic Knowledge Bases are a promising framework that can be usefully employed to tackle the problem of planning and decision making in artifact-centric business domains.

References

1. Baader, F., Zarrieß, B.: Verification of Golog programs over description logic actions. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8152 LNAI, 181–196 (2013)
2. Bhattacharya, K., Gerede, C., Hull, R.: Towards formal analysis of artifact-centric business process models. Business Process Management pp. 288–304 (2007)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and Databases: the DL-Lite Approach 5689, 255–356 (2009)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Linking data to ontologies: The description logic DL-Lite$_{\mathcal{A}}$. CEUR Workshop Proceedings 216 (2006)
5. Calvanese, D., De Giacomo, G., Montali, M., Patrizi, F.: Verification and Synthesis in Description Logic Based Dynamic Systems, Lecture Notes in Computer Science, vol. 7994. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
6. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull 32(3), 3–9 (2009)
7. Ghallab, M., Nau, D.S., Traverso, P.: Automated planning - theory and practice. Elsevier (2004)
8. Gigerenzer, G., Gaissmaier, W.: Heuristic decision making. Annual review of psychology 62, 451–482 (2011)
9. Hariri, B.B., Calvanese, D., Montali, M., De Giacomo, G., Masellis, R.D., Felli, P.: Description Logic Knowledge and Action Bases. J. Artif. Intell. Res. (JAIR) 46, 651–686 (2013)
10. Levesque, H.J.: Foundations of a Functional Approach to Knowledge Representation. Artif. Intell. 23(2), 155–212 (1984)
11. Stawowy, M.: Optimizations for decision making and planning in description logic based dynamic knowledge bases (2015), `http://arxiv.org/abs/1502.04665`

On Implementing Temporal Query Answering in *DL-Lite** (extended abstract)

Veronika Thost[1], Jan Holste[2], and Özgür Özçep[3]

[1] Technische Universität Dresden, Germany, thost@tcs.inf.tu-dresden.de
[2] Technische Universität Hamburg-Harburg, Germany, mail@janholste.com
[3] University of Lübeck, Germany, oezcep@ifis.uni-luebeck.de

Temporal information plays a central role in many applications of ontology-based data access (OBDA). For example, knowledge about the past is usually kept in patient records, and collected by companies or scientific projects as MesoWest[4], focusing on weather data. Such applications obviously benefit from using ontologies for data integration and access (e.g., the wind force 'Storm' on the well-known Beaufort Wind Force Scale is equally characterized by wind speed and wave height, which can be represented by a general concept inclusion as $\mathsf{HighWindSpeed} \sqcup \mathsf{HighWaves} \sqsubseteq \mathsf{Storm}$). Temporal knowledge is however not taken into account by systems implementing OBDA, in general. Though, assuming that we consider several weather stations' data of the past 24 hours, a query such as the following could be interesting: *"Get the heritage sites that are nearby a weather station, for which at some time in the past (24 hours) a danger of a hurricane was detected, since then, the wind force has been continuously very high, and it increased considerably during the two latest times of observation."*

For that reason, we investigate different approaches for answering *temporal conjunctive queries* (TCQs) [4, 5] w.r.t. ontologies written in the description logic (DL) *DL-Lite*$_{core}$ (hereinafter called *DL-Lite*). TCQs combine conjunctive queries (CQs) via LTL operators[5] and have already been studied extensively in the context of *DL-Lite* [13, 8]. The above example query could be specified as the following TCQ:

$$\mathsf{HeritageSite}(x) \wedge \mathsf{WeatherStation}(y) \wedge \mathsf{nearby}(x,y) \wedge$$
$$\big(\mathsf{HighWind}(y)\ \mathsf{S}\ \mathsf{DangerOfHurricane}(y)\big) \wedge\ \bigcirc^{-}\ \mathsf{Storm}(y) \wedge \mathsf{ViolentStorm}(y),$$

asking for all pairs (x, y) of heritage sites and nearby weather stations, whose sensor values at some point in time implied a danger of a hurricane, *since* (S) then, the measurements have implied Beaufort category 'high wind', in the *previous* ($\bigcirc^{-}$) moment of observation they implied category 'storm', and the latest values imply 'violent storm'. The semantics of TCQs is based on temporal knowledge bases, which, in addition to the ontology (assumed to hold at every point in time), contain a sequence of fact bases $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$, representing the data collected at specific points in time. Especially note that the ontology and the fact bases itself are formulated in a classical DL.

Related Work

On the DL side, there are various optimized systems realizing OBDA [10]. In particular, the so-called *rewriting approach* realized by Ontop [15] allows for efficient query

*Partially supported by the DFG in CRC 912 (HAEC) and by the EU Commission as part of the FP7 project Optique.

[4] http://mesowest.utah.edu/

[5] Please note that we do not consider negation.

answering w.r.t. an ontology written in *DL-Lite*. Specifically, Ontop internally rewrites a given conjunctive query, which is written in the abstract vocabulary of the ontology, into an SQL query that encodes the relevant ontological knowledge but addresses a standard database system; the latter can then be used to store the data and efficiently answer the queries. But whereas a lot of DL research is studying temporal extensions of ontology and query languages [2, 6, 3, 11, 13, 5, 8, 14], none of the freely available systems takes the temporal nature of the data into account, yet (i.e., the query languages supported do not provide operators for explicitly referencing different points in time). Nevertheless, recently, several practical approaches for answering temporal queries have been developed in the fields of (RDF) stream reasoning [7] and complex event processing [9, 1]. These systems are tailored to continuously answering given queries over an infinite stream of data—which is usually realized by restricting the focus to a *window* of the data (i.e., instead of considering all the past data available, the number of considered time points or data instances is fix). But only few of these systems support ontologies, yet, and standards for a common stream representation and processing, for query languages, and for operation semantics are still to be developed.

Our work complements these applied approaches by starting from a DL perspective, where many use cases consider static data, ontologies are important, and there are several well-investigated query languages. Specifically, we study three pragmatic approaches for answering TCQs based on the work of [8]. We prototypically implemented and evaluated them, and in this paper report on our experiences.

Algorithms for Temporal Query Answering

In particular, [8] propose algorithms for answering temporal queries (w.r.t. TKBs) that generalize TCQs in that they combine queries of a generic atemporal query language $\mathcal{Q}$ via LTL-operators (i.e., we restrict $\mathcal{Q}$ to CQs).[6] Similar to the streaming scenario, [8] assume a fix set of TCQs to be answered continuously at time point n.

We first implemented the Iterative Algorithm (IA) (cf. Section 6 in [8]), which iteratively computes sets of answers to several subqueries of the TCQ to be answered, for each time point i, $0 \leq i \leq n$. For example, the answers to $\bigcirc^{-}\mathsf{Storm}(x)$ at i are obtained by evaluating $\mathsf{Storm}(x)$ at $i-1$. Since the processing at i only uses $\mathcal{A}_i$ and the sets computed for the previous moment, whose sizes are bounded, the IA achieves a so-called *bounded history encoding* (i.e., it's runtime does not depend on the number of considered fact bases). A growing number of data from the past however usually leads to an increase in processing time, in practice. We therefore describe a window-based variant of the IA, the Vector Algorithm (VA) (cf. Section 4 in [12]). With this approach, the entire history can still be regarded (i.e., by considering it as one window) but, for example, streaming scenarios can be managed, too. The VA specifically supports *sliding* windows, where the TCQs are not evaluated at every time point, but in fix intervals, by only regarding the then necessary of the above mentioned sets of answers. To answer the query $\bigcirc^{-}\mathsf{Storm}(x)$ at every second point in time, for example, $\mathsf{Storm}(x)$ does not always have to be evaluated. Our implementations of both algorithms are based on materializing the fact bases and thus may be extended in the future w.r.t. other rewritable

[6] [8] assume $\mathcal{Q}$ to be a *rewritable* query language, which basically means that the certain answers to every query in $\mathcal{Q}$ w.r.t. a knowledge base $\mathcal{K}$ can be obtained by answering an appropriate adaptation of the query in a canonical model of $\mathcal{K}$.

query languages (e.g., by employing a corresponding reasoner for the atemporal query answering, temporal queries over $\mathcal{EL}$-TKBs could be answered).

To get an impression of the performance of a temporal rewriting approach, we finally consider (a rather simple) Rewriting Algorithm (RA) translating TCQs into standard database queries. Note that the idea is similarly described by [8]. However, in [8], the TCQs are assumed to be rewritten into a *temporal* standard query language. Since it turned out that such query languages are only partially supported by existing databases, we describe a rewriting into basic SQL. In particular, we developed the QuAnTOn library, which translates TCQs w.r.t. a *DL-Lite* ontology into SQL queries encoding the relevant ontological knowledge and addressing a standard database. QuAnTOn applies Ontop for rewriting the CQs into SQL, appropriately combines these SQL queries, and adds clauses specifying the temporal conditions. For example, the TCQ $\bigcirc^{-}\mathsf{Storm}(x)$ could be translated into the below SQL query with the inner query coming from Ontop.

```
SELECT loc FROM
(SELECT loc, sensor, timestamp FROM sensorvalues
        WHERE (sensor = 'wind-speed' AND value > 24.5) OR
                (sensor = 'wave-height' AND value > 9))
WHERE timestamp = current_time-1;
```

Evaluation Results

We focus on the querying of weather data as outlined above and regard sequences of fact bases each containing 1000 assertions representing sensor measurements. We chose this (streaming) scenario, sketched in [17], to facilitate a comparison with stream reasoning systems, which is planned for future work. We also use an extension of the corresponding sensor-observation ontology[7]. To especially learn about the performance of different kinds of TCQs, we only use very simple CQs within the latter. We investigate the time needed for rewriting and answering TCQs in dependence of the number $n + 1$ of fact bases considered and show the below;[8] the full details can be found in [16, 12].

- The time required for the initial preprocessing (per TCQ) is negligible: less than 1 ms for the IA/VA and about 200 ms for the application of the RA.
- The scenario considering a fix set of sensors suits our constant domain assumption, and the IA shows a rather constant performance (Figure 1).
- The VA is only applicable for windows of moderate size, due to memory issues.
- The time taken for answering the queries of the RA strongly depends on the kind of the queries and the application (e.g., the impact of fetching the results is large).

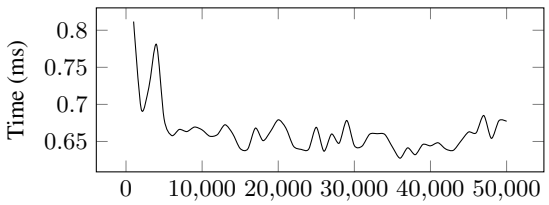


Fig. 1. The time the IA takes for answering a TCQ in dependence of the number of fact bases, $n+1$.

[7] http://wiki.knoesis.org/index.php/LinkedSensorData

[8] The tests were run on an 2,2 GHz Intel Core i7 machine with 4 GB RAM. For answering the queries returned by QuAnTOn, we applied a MySQL (v5.2.38) database.

References

1. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semant. web 3(4), 397–407 (Oct 2012)
2. Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: Goranko, V., Wang, X.S. (eds.) Proceedings of the 14th International Symposium on Temporal Representation and Reasoning (TIME 2007), pp. 11–22. IEEE Press (2007)
3. Artale, A., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: A cookbook for temporal conceptual data modelling with description logics. ACM Transactions on Computational Logic 15(3), 25 (2014)
4. Baader, F., Borgwardt, S., Lippmann, M.: Temporalizing ontology-based data access. In: Bonacina, M.P. (ed.) Proc. of the 24th Int. Conf. on Automated Deduction (CADE'13). Lecture Notes in Computer Science, vol. 7898, pp. 330–344. Springer-Verlag (2013)
5. Baader, F., Borgwardt, S., Lippmann, M.: Temporal query entailment in the description logic $\mathcal{SHQ}$. Journal of Web Semantics (2015)
6. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. ACM Transactions on Computational Logic 13(3), 21:1–21:32 (2012)
7. Balduini, M., Calbimonte, J.P., Corcho, O., Dell'Aglio, D., Della Valle, E.: Stream reasoning for linked data, `http://streamreasoning.org/events/sr4ld2014`
8. Borgwardt, S., Lippmann, M., Thost, V.: Temporalizing rewritable query languages over knowledge bases. Journal of Web Semantics (2015), in press.
9. Cugola, G., Margara, A.: Tesla: A formally defined event specification language. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS '10). pp. 50–61. ACM, New York, NY, USA (2010)
10. Gonçalves, R.S., Bail, S., Jimenez-ruiz, E., Parsia, B., Glimm, B., Kazakov, Y.: OWL Reasoner Evaluation (ORE) workshop 2013 results: Short report
11. Gutiérrez-Basulto, V., Jung, J.C., Schneider, T.: Lightweight description logics and branching time: A troublesome marriage. In: Proc. of the 14th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'14). AAAI Press (2014)
12. Holste, J.: Ontology-Based Temporal Reasoning on Streams. Master's thesis, Hamburg University of Technology (October 2014), `http://www.sts.tuhh.de/pw-and-m-theses/2014/holste14a.pdf`
13. Klarman, S., Meyer, T.: Querying temporal databases via OWL 2 QL. In: Kontchakov, R., Mugnier, M.L. (eds.) Web Reasoning and Rule Systems, Lecture Notes in Computer Science, vol. 8741, pp. 92–107. Springer International Publishing (2014)
14. Özçep, O., Möller, R., Neuenstadt, C.: A stream-temporal query language for ontology based data access. In: Lutz, C., Thielscher, M. (eds.) KI 2014: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 8736, pp. 183–194. Springer International Publishing (2014)
15. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over *DL-Lite* ontologies. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (2012)
16. Thost, V., Holste, J., Özgür Özçep: On implementing temporal query answering in *DL-Lite*. LTCS-Report 15-12, Chair for Automata Theory, TU Dresden, Germany (2015), `http://lat.inf.tu-dresden.de/research/reports/2015/THO-LTCS-15-12.pdf`, see http://lat.inf.tu-dresden.de/research/reports.html.
17. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.P.: SRBench: a streaming RDF/SPARQL benchmark. In: Proceedings of the 11th International Conference on The Semantic Web - Volume Part I. pp. 641–657. ISWC'12, Springer-Verlag, Berlin, Heidelberg (2012)