Data Complexity of Answering Unions of Conjunctive Queries in $\mathcal{SHIQ}^*$

Magdalena Ortiz[1,2], Diego Calvanese[1], Thomas Eiter[2]

[1] Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
calvanese@inf.unibz.it,
magdalena.ortiz@stud-inf.unibz.it

[2] Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, Vienna, Austria
eiter@kr.tuwien.ac.at

Abstract

The novel context of accessing and querying large data repositories through ontologies that are formalized in terms of expressive DLs requires on the one hand to consider query answering as the primary inference technique, and on the other hand to optimize it with respect to the size of the data, which dominates the size of ontologies. While the complexity of DLs has been studied extensively, data complexity in expressive DLs has been characterized only for answering atomic queries, and was still open for more expressive query languages, such as unions of conjunctive queries (UCQs). In this paper we advocate the need for studying this problem, and provide a significant technical contribution in this direction. Specifically, we prove a tight coNP upper bound for answering UCQs over $\mathcal{SHIQ}$ knowledge bases, for the case where the queries do not contain transitive roles. We thus establish that for a whole range of DLs from AL to $\mathcal{SHIQ}$, answering such UCQs has coNP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, inspired by the one in [20], but which manages the technical challenges of simultaneous inverse roles and number restrictions (which leads to a DL lacking the finite model property).

1 Introduction

Description Logics (DLs) [2] provide the formal foundations for the standard Web ontology languages [11]. In fact, OWL-Lite and OWL-DL are syntactic variants of the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$, respectively [12, 22]. In the Semantic Web and domains such as Enterprise Application Integration and Data Integration [19], ontologies provide a high-level, conceptual view of the relevant information. However, they are increasingly seen also as a mechanism to access and query data repositories.

*This work was partially supported by the Austrian Science Funds (FWF) project P17212, by the European Commission project REWERSE (IST-2003-506779), and by the European Commission FET project TONES (FP6-7603).

This novel context poses an original combination of challenges both in DLs/ontologies and in related areas such as data modeling and querying in databases:

(*i*) On the one hand, data repositories can be very large and are usually much larger than the intensional level expressing constraints on the data. Therefore, the contribution of the extensional data to inference complexity must be singled out, and one must pay attention to optimizing inference techniques with respect to data size, as opposed to the overall size of the knowledge base. In databases, this is accounted for by *data complexity* of query answering [26], where the relevant parameter is the size of the data, as opposed to *combined complexity*, which additionally considers the size of the query and of the schema.

(*ii*) On the other hand, the data underlying an ontology should be accessed using well established and flexible mechanisms such as those provided by database query languages. This goes well beyond the traditional inference tasks involving objects in DL-based systems, like *instance checking* [10, 23]. Indeed, since explicit variables are missing, DL concepts have limited possibility for relating specific data items to each other. *Conjunctive queries (CQs)* and unions of CQs (UCQs) provide a good tradeoff between expressive power and nice computational properties, and thus are adopted as core query language in several contexts, such as data integration [19].

(*iii*) Finally, the considered DLs should have sufficient expressive power to capture common constructs used in data modeling [4]. This calls for *expressive DLs* [5], such as $\mathcal{SHIQ}$ [13, 15], featuring besides full Booleans also number restrictions, inverse and transitive roles, and role hierarchies.

As for data complexity of DLs, [10, 23] showed that instance checking is coNP-hard already in the rather weak DL $\mathcal{ALE}$, and [7] that CQ answering is coNP-hard in the yet weaker DL $\mathcal{AL}$. For suitably tailored DLs, answering UCQs is polynomial (actually LogSpace) in data complexity [6, 7]; see [7] for an investigation of the NLogSpace, PTime, and coNP boundaries.

For expressive DLs (with the features above, notably inverse roles), TBox+ABox reasoning has been studied extensively using techniques ranging from reductions to Propositional Dynamic Logic (PDL) (see, e.g., [8, 5]) over tableaux [3, 15] to automata on infinite trees [5, 25]. For many such DLs, the combined complexity of TBox+ABox reasoning is ExpTime-complete, including $\mathcal{ALCQI}$ [5, 25], $\mathcal{DLR}$ [8], and $\mathcal{SHIQ}$ [25]. However, until recently, little attention has been explicitly devoted to data complexity in expressive DLs. An ExpTime upper bound for data complexity of UCQ answering in $\mathcal{DLR}$ follows from the results on UCQ containment and view-based query answering in [8, 9]. They are based on a reduction to reasoning in PDL, which however prevents to single out the contribution to the complexity coming from the ABox. In [20] a tight coNP upper bound for CQ answering in $\mathcal{ALCNR}$ is shown. However, this DL lacks inverse roles and is thus not suited to capture semantic data models or UML. In [17, 18] a technique based on a reduction to Disjunctive Datalog is used for $\mathcal{ALCHIQ}$. For instance checking, it provides a (tight) coNP upper bound for data complexity, since it allows to single out the ABox contribution. The result can immediately be extended to tree shaped conjunctively queries, since these admit a representation as a description logic concept (e.g., by making use of the notion of tuple-graph of [8], or via rolling up [16]). However, this is not the case for general CQs, resulting in a non-tight

2ExpTime upper bound (matching also combined complexity).

Summing up, a precise characterization of data complexity for UCQ answering in expressive DLs was still open, with a gap between a coNP lower-bound and an ExpTime upper bound. We close this gap, thus simultaneously addressing the three challenges identified above. Specifically, we make the following contributions:

- Building on techniques of [20, 15], we devise a novel tableaux-based algorithm for answering UCQs where all roles are simple over $\mathcal{SHIQ}$ knowledge bases. Technically, to show its soundness and completeness, we have to deal both with a novel blocking condition (inspired by the one in [20], but taking into account inverse and transitive roles), and with the lack of the finite model property.

- This novel algorithm provides us with a characterization of data complexity for UCQ answering in expressive DLs. Specifically, we show that data complexity of answering such an UCQ over $\mathcal{SHIQ}$ knowledge bases is in coNP, and thus coNP-complete for all DLs ranging from $\mathcal{AL}$ to $\mathcal{SHIQ}$.

For space reasons, proofs are omitted here; they can be found in [21].

2 Preliminaries

We only briefly recall $\mathcal{SHIQ}$ and refer to the literature (e.g., [13, 15]) for details and background. We denote by $\mathbf{C}$, $\mathbf{R}$, $\mathbf{R}_+$ (where $\mathbf{R}_+ \subseteq \mathbf{R}$), and $\mathbf{I}$ the sets of *concept names*, *role names*, *transitive role names*, and *individuals* respectively. A *role* R is either an atomic role name P or its inverse P^-. A *concept* C is either an atomic concept name A or one of $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\geq n\, S.C$, or $\leq n\, S.C$, where C and D denote concepts, R a role, S a *simple* role (see later), and $n \geq 0$ an integer. A *knowledge base* is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where the *TBox* $\mathcal{T}$ is a set of *concept inclusion axioms* $C_1 \sqsubseteq C_2$; the *role hierarchy* $\mathcal{R}$ is a set of *role inclusion axioms* $R_1 \sqsubseteq R_2$; and the *ABox* $\mathcal{A}$ is a set of *assertions* $A(a)$, $P(a, b)$, and $a \neq b$, where A (resp., P) is an atomic concept (resp., role) and a and b are individuals.

A role S is *simple*, if for no role $R \in \mathbf{R}_+$ we have that $R \sqsubseteq^* S$, where $\sqsubseteq^*$ denotes the reflexive and transitive closure of the subrole relation $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R_1) \sqsubseteq \mathsf{Inv}(R_2) \mid R_1 \sqsubseteq R_2 \in \mathcal{R}\}$. Without loss of expressivity, we assume that all concepts in K are in *negation normal form* (NNF). For a concept C, $\mathsf{clos}(C)$ is the smallest set of concepts containing C that is closed under subconcepts and their negation (in NNF); and $\mathsf{clos}(K)$ denotes the union of all $\mathsf{clos}(C)$ for each C occurring in K. We will use K to denote a knowledge base $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, $\mathbf{R}_K$ the roles occurring in K and their inverses, and $\mathbf{I}_K$ the individuals occurring in $\mathcal{A}$.

The semantics of K is defined in terms of first-order *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ the valuation function, as usual (without unique names assumption;[1] see [2]). $\mathcal{I}$ is a *model* of K, denoted $\mathcal{I} \models K$, if it satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

[1] The unique names assumption can be easily emulated using $\not\approx$.

Example 1 *As a running example, we use the knowledge base*

$$K = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\}\rangle.$$

We assume that K has an associated set of *distinguished concept names*, denoted $\mathcal{C}_q$, which are the concepts that can occur in queries.

Definition 1 (Union of Conjunctive Queries) *A* conjunctive query *(CQ) Q over a knowledge base K is a set of atoms of the form $\{p_1(\overline{Y_1}), \ldots, p_n(\overline{Y_n})\}$ where each p_i is either a role name in $\mathbf{R}_K$ or a concept in $\mathcal{C}_q$, and $\overline{Y_i}$ is a tuple of variables or individuals in $\mathbf{I}_K$ matching its arity. A* union of CQs (UCQ) *U is defined as an expression of the form $Q_1 \vee \cdots \vee Q_m$ where Q_i is a CQ for each $i \in \{1, \ldots, m\}$.*

For short, a *query* is either a CQ or a UCQ. We will restrict our attention to queries where each role occurring in some $p_i(\overline{Y_i})$ is a *simple* role, i.e., transitive roles and super-roles of transitive roles are disallowed in queries. We denote by $\mathsf{varind}(Q)$ the set of variables and individuals in query Q. An interpretation $\mathcal{I}$ is a model of a CQ Q, denoted $\mathcal{I} \models Q$, if there is a substitution $\sigma : \mathsf{varind}(Q) \to \Delta^{\mathcal{I}}$ such that $\sigma(a) = a^{\mathcal{I}}$ for each individual $a \in \mathsf{varind}(Q)$ and $\mathcal{I} \models p(\sigma(\overline{Y}))$, for each $p(\overline{Y})$ in Q. For a UCQ $U = Q_1 \vee \cdots \vee Q_m$, $\mathcal{I} \models U$ is defined as $\mathcal{I} \models Q_i$ for some $i \in \{1, \ldots, m\}$. We say that K *entails* query Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K.

Example 2 *Let $\mathcal{C}_q = \{A\}$. We consider the CQs $Q_1 = \{P_1(x,y), P_2(x,z), A(y)\}$ and $Q_2 = \{P_2(x,y), P_2(y,z)\}$ and the UCQ $U = Q_1 \vee Q_2$. Note that $K \models Q_1$. Indeed, for an arbitrary model $\mathcal{I}$ of K, we can map x to $a^{\mathcal{I}}$, y to an object connected to $a^{\mathcal{I}}$ via role P_1 (which by the inclusion axiom $A \sqsubseteq \exists P_1.A$ exists and is an instance of A), and z to an object connected to $a^{\mathcal{I}}$ via role P_2 (which exists by the inclusion $A \sqsubseteq \exists P_2.\neg A$). Also, $K \not\models Q_2$. A model $\mathcal{I}$ of K that is not a model of Q_2 is the one with $\Delta^{\mathcal{I}} = \{o_1, o_2\}$, $a^{\mathcal{I}} = o_1$, $A^{\mathcal{I}} = \{o_1\}$, $P_1^{\mathcal{I}} = \{\langle o_1, o_1\rangle\}$, and $P_2^{\mathcal{I}} = \{\langle o_1, o_2\rangle\}$. Finally, since $K \models Q_1$, then also $K \models U$.*

Query answering for a certain DL $\mathcal{L}$ is in a complexity class $\mathcal{C}$, if given any knowledge base K in $\mathcal{L}$ and query Q, deciding $K \models Q$ is in $\mathcal{C}$; this is also called *combined complexity*. The *data complexity* of query answering is the complexity of deciding $K \models Q$ where Q and all of K except $\mathcal{A}$ is fixed.

We only consider Boolean queries (i.e., yes/no queries, with no free variables), to which queries with distinguished variables are reducible as usual. Note that query answering is not reducible to knowledge base satisfiability, since the negated query is not expressible within the knowledge base.

3 The Query Answering Algorithm

In this section, Q denotes a CQ and U a UCQ. As mentioned, we assume that all roles occurring in queries are simple. We first describe our method for deciding $K \models Q$, and then how it is extended to $K \models U$. Our technique builds on the $\mathcal{SHIQ}$ satisfiability algorithm in [15]. The adaption to query answering is inspired by [20], yet we deal with DLs that have no finite model property.

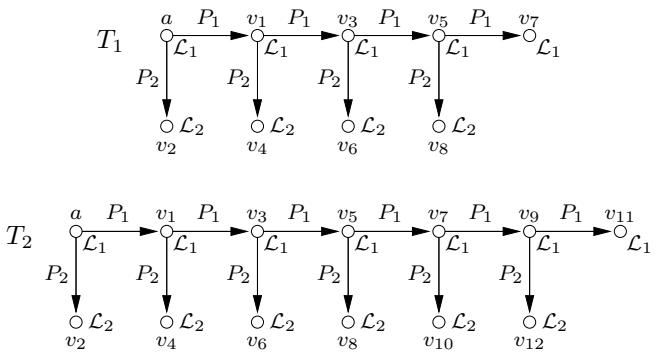

Figure 1: Trees and completion forests for the example knowledge base

We use *completion forests*, which are finite relational structures capturing sets of models of K. Roughly speaking, the models of K are represented by an initial completion forest $\mathcal{F}_K$. By applying tableaux-style *expansion rules* repeatedly, new completion forests are generated non-deterministically where also new individuals might be introduced. Each model of K is preserved in some of the resulting forests. Therefore, checking $K \models Q$ equals checking $\mathcal{F} \models Q$ for each completion forest $\mathcal{F}$. The blocking conditions on the rules, which ensure that expansion terminates, are more involved than those in [15]. They require a depth parameter n (depending on Q) that ensures a sufficient expansion of each forest $\mathcal{F}$, in such a way that semantical entailment of Q in $\mathcal{F}$ can be checked effectively via a syntactic mapping of the variables in Q to the nodes in $\mathcal{F}$. Thus, to witness that $K \not\models Q$, it is sufficient to (nondeterministically) construct a large enough forest $\mathcal{F}$ to which Q cannot be mapped.

A *variable tree* T is a tree whose nodes are variables except the root, which might be also an individual, and where each node v is labeled with a set of concepts $\mathcal{L}(v)$ and each arc $v \to w$ is labeled with a set of roles $\mathcal{L}(v \to w)$. For any integer $n \geq 0$, the *n-tree* of a node v in T, denoted T_v^n, is the subtree of T rooted at v that contains all descendants of v within distance n. Variables v and v' in T are *n-tree equivalent in T*, if T_v^n and $T_{v'}^n$ are isomorphic (i.e., there is a bijection from the nodes of T_v^n to those of $T_{v'}^n$ which preserves all labels). If, for such v and v', v' is an ancestor of v in T and v is not in $T_{v'}^n$, then we say that $T_{v'}^n$ *tree-blocks* T_v^n. A *completion forest* (cf. [15]) for K is constituted by (i) a set of variable trees whose roots are the individuals in $\mathbf{I}_K$ and can be arbitrarily connected by arcs; and (ii) a binary relation $\not\approx$ on the individuals in $\mathbf{I}_K$, implicitly assumed to be symmetric.

Example 3 *Consider the variable tree T_1 in Figure 1, with a as root, $\mathcal{L}_1 = \{A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\neg A, A \sqcup \neg A, \exists P_1.A, \exists P_2.\neg A\}$, and $\mathcal{L}_2 = \{\neg A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\neg A, A \sqcup \neg A\}$. Then, v_1 and v_5 are 1-tree equivalent in T_1 and ${T_1}_{v_1}^1$ tree-blocks ${T_1}_{v_5}^1$.*

Now we introduce the initial completion forest for K. We use a set of *global concepts* $\mathsf{gcon}(K, \mathcal{C}_q) = \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \{C \sqcup \neg C \mid C \in \mathcal{C}_q\}$. Informally, by requiring that each individual belongs to all global concepts, satisfaction of the TBox

is enforced and, by case splitting, each individual can be classified with respect to the distinguished concepts (i.e., those appearing in queries).

The *initial completion forest* for K, denoted $\mathcal{F}_K$, is defined as follows:[2]

- The nodes are the individuals $a \in \mathbf{I}_K$, and $\mathcal{L}(a) = \{A \mid A(a) \in \mathcal{A}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$.
- The arc $a \to a'$ is present iff $\mathcal{A}$ contains some assertion $P(a, a')$, and $\mathcal{L}(a \to a') = \{P \mid P(a, a') \in \mathcal{A}\}$.
- $a \not\approx a'$ iff $a \neq a' \in \mathcal{A}$.

Example 4 *In our running example, $\mathcal{F}_K$ contains only the node a which has the label $\mathcal{L}(a) := \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A\}$.*

In the expansion rules, we use a notion of blocking that depends on a depth parameter n: a node is *n-blocked* if it's a leaf of a tree-blocked n-tree in $\mathcal{F}$. Note that if v is n-blocked, then it is m-blocked for each $m \leq n$. For $n \geq 1$, n-blocking implies blocking as in [15], and for $n = 0$ amounts to blocking by equal node labels. Starting from $\mathcal{F}_K$, we can generate a set $\mathbb{F}_K$ of new completion forests by applying expansion rules. The rules are analogous to those in [15], but they use "n-blocking" and initialize newly introduced nodes with a label containing $\mathsf{gcon}(K, \mathcal{C}_q)$. When applying the expansion rules, a *clash* may arise in some $\mathcal{F}$, i.e., two complementary concepts C and $\neg C$ occur in a node label, or a node that should satisfy a number restriction $\leq n\, S.C$ has more than n distinct S successors marked C. If this is not the case, then $\mathcal{F}$ is *clash free*. We call a completion forest *n-complete*, if (under n-blocking) no rule can be applied to it. We denote by $\mathsf{ccf}_n(\mathbb{F}_K)$ the set of *n-complete* and clash free completion forests in $\mathbb{F}_K$. For the set of expansion rules and more details, see [21].

Example 5 *Consider the completion forest $\mathcal{F}_1$ with the variable tree T_1 from Example 3 and with empty $\not\approx$. $\mathcal{F}_1$ is 1-blocked. Analogously, consider $\mathcal{F}_2$ that has T_2 in Figure 1 and where $\not\approx$ is also empty. $\mathcal{F}_2$ is 2-blocked. Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be obtained from $\mathcal{F}_K$ by applying the expansion rules. They are both complete and clash-free, so $\mathcal{F}_1 \in \mathsf{ccf}_1(\mathbb{F}_K)$ and $\mathcal{F}_2 \in \mathsf{ccf}_2(\mathbb{F}_K)$.*

Models of a completion forest

Viewing variables in a completion forest $\mathcal{F}$ as individuals, we can define models of $\mathcal{F}$ as models of K (over an extended vocabulary). An interpretation $\mathcal{I}$ is a *model* of a completion forest $\mathcal{F}$ for K, denoted $\mathcal{I} \models \mathcal{F}$, if $\mathcal{I} \models K$ and for all nodes v, w in $\mathcal{F}$ it holds that (i) $v^{\mathcal{I}} \in C^{\mathcal{I}}$ if $C \in \mathcal{L}(v)$, (ii) $\langle v^{\mathcal{I}}, w^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ if $\mathcal{F}$ has an arc $v \to w$ and $R \in \mathcal{L}(\langle v, w \rangle)$, and (iii) $v^{\mathcal{I}} \neq w^{\mathcal{I}}$ if $v \not\approx w \in \mathcal{F}$.

Clearly, the models of the initial completion forest $\mathcal{F}_K$ and of K coincide, and thus $\mathcal{F}_K$ semantically represents K. Then, each time an expansion rule is applied, all models are preserved in some resulting forest. As a consequence, it holds that for each model $\mathcal{I}$ of K, there exists some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ and a model of $\mathcal{F}$ which extends $\mathcal{I}$ (for any $n \geq 0$). Since the set of n-complete and clash-free forests for K semantically

[2] If $\mathcal{A} = \emptyset$, then $\mathcal{F}_K$ contains a single node a with $\mathcal{L}(a) = \mathsf{gcon}(K, \mathcal{C}_q)$.

captures K (modulo new individuals), we can transfer query entailment $K \models Q$ to logical consequence of Q from completion forests as follows. For any completion forest $\mathcal{F}$ and CQ Q, let $\mathcal{F} \models Q$ denote that $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of $\mathcal{F}$.

Proposition 1 *Let Q be a CQ in which all roles are simple and let $n \geq 0$ be arbitrary. $K \models Q$ iff $\mathcal{F} \models Q$ for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$.*

Now we will show that, if n is sufficiently large, we can decide $\mathcal{F} \models Q$ for an $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ by syntactically mapping the query Q into $\mathcal{F}$. We say that Q can be *mapped into* $\mathcal{F}$, denoted $Q \hookrightarrow \mathcal{F}$, if there is a mapping μ from the variables and individuals in $\mathsf{varind}(Q)$ into the nodes of $\mathcal{F}$, such that (i) for each individual a, $\mu(a) = a$; (ii) for each atom $C(x)$ in Q, $C \in \mathcal{L}(\mu(x))$; and (iii) for each atom $R(x, y)$ in Q, there is an arc $\mu(x) \to \mu(y)$ whose label contains R' or an arc $\mu(y) \to \mu(x)$ whose label contains the inverse of R' for some $R' \sqsubseteq^* R$.

Example 6 *$Q_1 \hookrightarrow \mathcal{F}_2$ holds, as witnessed by the mapping $\mu(x) = a$, $\mu(y) = v_2$, and $\mu(z) = v_1$. Note that there is no mapping of Q_2 into $\mathcal{F}_2$ satisfying the above conditions.*

It is clear that if Q can be mapped to $\mathcal{F}$ via μ, then Q is satisfied in each model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{F}$ by assigning to each variable x in Q the value of its image $\mu(x)^{\mathcal{I}}$. Thus $Q \hookrightarrow \mathcal{F}$ implies $\mathcal{F} \models Q$. To prove that the converse also holds, we have to show that if n is large enough, a mapping of Q into $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ can be constructed from a distinguished canonical model of $\mathcal{F}$.

The *canonical model* $\mathcal{I}_{\mathcal{F}}$ of $\mathcal{F}$ is constructed by unraveling the forest $\mathcal{F}$ in the standard way, where the blocked nodes act like 'loops'. Its domain comprises the set of all paths from some root in $\mathcal{F}$ to some node of $\mathcal{F}$ (thus, it can be infinite). Note that in order for $\mathcal{I}_{\mathcal{F}}$ to be a model, $\mathcal{F}$ must be in $\mathsf{ccf}_n(\mathbb{F}_K)$ for some $n \geq 1$. The formal definition of $\mathcal{I}_{\mathcal{F}}$ is then straightforward yet complex, and we must refer to the extended report [21] for the details. Instead, we provide an example.

Example 7 *By unraveling $\mathcal{F}_2$, we obtain a model $\mathcal{I}_{\mathcal{F}_2}$ that has as domain the infinite set of paths from a to each v_i. Note that a path actually comprises a sequence of pairs of nodes, in order to witness the loops introduced by blocked variables. When a node is not blocked, like v_1, the pair $\frac{v_1}{v_1}$ is added to the path. Since $T^2_{v_1}$ tree-blocks $T^2_{v_7}$, every time a path reaches v_{11}, which is a leaf of a blocked tree, we add $\frac{v_5}{v_{11}}$ to the path and 'loop' back to the successors of v_5. In this way, we obtain the following infinite set of paths:*

$$
\begin{array}{ll}
p_0 = [\frac{a}{a}], & p_7 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}], \\
p_1 = [\frac{a}{a}, \frac{v_1}{v_1}], & p_8 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_8}{v_8}], \\
p_2 = [\frac{a}{a}, \frac{v_2}{v_2}], & p_9 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}], \\
p_3 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}], & p_{10} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_{10}}{v_{10}}], \\
p_4 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_4}{v_4}], & p_{11} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}], \\
p_5 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}], & p_{12} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_{12}}{v_{12}}], \\
p_6 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_6}{v_6}], & p_{13} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}, \frac{v_7}{v_7}], \\
 & p_{14} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}, \frac{v_8}{v_8}], \\
 & \vdots
\end{array}
$$

This set of paths is the domain of $\mathcal{I}_{\mathcal{F}_2}$. The extension of each concept C is determined by the set all p_i such that C occurs in the label of the last node in p_i. For the extension of each role R, we consider the pairs $\langle p_i, p_j \rangle$ such that the last node in p_j is an R-successor of p_i (the extension of the roles in $\mathbf{R}_+$ are transitively expanded). Therefore p_0, p_1, p_3, ... are in $A^{\mathcal{I}_{\mathcal{F}_2}}$, and $\langle p_0, p_1 \rangle$, $\langle p_1, p_3 \rangle$, $\langle p_3, p_5 \rangle$, $\langle p_5, p_7 \rangle$, ... are all in $P_1^{\mathcal{I}_{\mathcal{F}_2}}$.

In the following, let n_Q denote the number of role atoms in Q, and let $n \geq n_Q$. For any forest $\mathcal{F}$ that is n-complete, a mapping μ of Q into $\mathcal{F}$ can be obtained from any mapping σ of $\mathsf{varind}(Q)$ into $\mathcal{I}_{\mathcal{F}}$ satisfying Q. If we see the image of Q under σ as a graph G (considering only the edges that correspond to simple roles), then the length of a path connecting the images $\sigma(x)$ and $\sigma(y)$ of any two variables x and y in Q will be at most n_Q. Since $\mathcal{F}$ contains at least two non-overlapping trees of size n, it is big enough to ensure that for each path in G there is an isomorphic one in $\mathcal{F}$.

Proposition 2 *Let Q be a CQ in which all roles are simple, let $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, with $n \geq n_Q$, and let $\mathcal{I}_{\mathcal{F}} \models Q$. Then $Q \hookrightarrow \mathcal{F}$.*

Example 8 *$\mathcal{I}_{\mathcal{F}_2}$ models Q_1, as witnessed by the substitution $\sigma(x) = p_7$, $\sigma(y) = p_9$ and $\sigma(z) = p_{10}$. From it we can obtain the mapping $\mu(x) = v_7$, $\mu(y) = v_9$ and $\mu(z) = v_{10}$, which shows that $Q_1 \hookrightarrow \mathcal{F}_2$. Note that $n_{Q_1} = 2$ and the image of Q_1 under σ has no paths of length > 2.*

The results given above can be extended straightforwardly to a UCQ U. As before, we will use $\mathcal{F} \models U$ to denote that $\mathcal{F}$ semantically entails U (i.e., every model of $\mathcal{F}$ is a model of U), and $U \hookrightarrow \mathcal{F}$ to denote syntactical mappability, which is defined as $Q_i \hookrightarrow \mathcal{F}$ for some Q_i in U. We already know that to decide $K \models U$ it is sufficient to verify whether $\mathcal{F} \models U$ for every $\mathcal{F}$ in $\mathsf{ccf}_n(\mathbb{F}_K)$ for an arbitrary $n \geq 0$ (in fact, Proposition 1 holds for any kind of query). It is only left to prove that for a suitable n, $\mathcal{F} \models U$ can be effectively reduced to $U \hookrightarrow \mathcal{F}$.

Again, one direction is trivial. If $U \hookrightarrow \mathcal{F}$, then by definition there is some Q_i in U such that $Q_i \hookrightarrow \mathcal{F}$, and as this implies $\mathcal{F} \models Q_i$, we also have that $\mathcal{F} \models U$.

Example 9 *Since $Q_1 \hookrightarrow \mathcal{F}_2$ (see Examples 6 and 8), we have $U \hookrightarrow \mathcal{F}_2$. $\mathcal{F}_2 \models Q_1$ implies $\mathcal{F}_2 \models U$.*

The other direction is also quite straightforward. Consider n_U as the maximal n_{Q_i} for all Q_i in U. For each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, with $n \geq n_U$, if $\mathcal{I}_{\mathcal{F}} \models U$, then $\mathcal{I}_{\mathcal{F}} \models Q_i$ for some Q_i in U. Since $n \geq n_U \geq n_{Q_i}$, by Proposition 2 we know that $Q_i \hookrightarrow \mathcal{F}$ and then $U \hookrightarrow \mathcal{F}$. Thus $\mathcal{I}_{\mathcal{F}} \models U$ implies $U \hookrightarrow \mathcal{F}$ as well.

Finally, we establish our key result: answering $K \models U$ for a UCQ U reduces to finding a mapping of U into every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ for any $n \geq n_U$.

Theorem 3 *Let U be a UCQ in which all roles are simple and let $n \geq n_U$. Then $K \models U$ iff $U \hookrightarrow \mathcal{F}$ holds for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$.*

4 Complexity of Query Answering

In the following, $\|K, Q\|$ denotes the total size of the string encoding a given knowledge base K and a query Q. As shown in [21], for a CQ Q, branching in each variable tree in a completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ is polynomially bounded in $\|K, Q\|$, and the maximal depth of a variable is double exponential in $\|K, Q\|$ if n is polynomial in $\|K, Q\|$. Therefore, $\mathcal{F}$ has at most triple exponentially many nodes. Since each rule can be applied only polynomially often to a node, the expansion of the initial completion forest $\mathcal{F}_K$ into some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ terminates in nondeterministic triple exponential time in $\|K, Q\|$ for $n = n_Q$.

Theorem 4 *Given a $\mathcal{SHIQ}$ knowledge base K and a union of conjunctive queries U in which all roles are simple, deciding whether $K \models U$ is in* co-3NExpTime.

Proof (Sketch). It is sufficient to check for every $\mathcal{F} \in \mathsf{ccf}_{n_U}(\mathbb{F}_K)$ whether $U \hookrightarrow \mathcal{F}$, i.e., $Q_i \hookrightarrow \mathcal{F}$ for some CQ Q_i in U. The size of $\mathcal{F}$ is at most triple exponential in $\|K, Q_{i^*}\|$, where Q_{i^*} is such that $n_{Q_{i^*}} = n_U$, thus also in $\|K, U\|$. Furthermore, $Q_i \hookrightarrow \mathcal{F}$ can be checked by naive methods in triple exponential time in $\|K, Q_{i^*}\|$ and thus in $\|K, U\|$ as well. (We stress that this test is NP-hard even for fixed $\mathcal{F}$.) □

Notice that the result holds for binary encoding of number restrictions in K. An exponential drop results for unary encoding if Q is fixed.

Under data complexity, U and all components of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ except for the ABox $\mathcal{A}$ are fixed. Therefore, n_U is constant. Thus every completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ has *linearly* many nodes in $|\mathcal{A}|$, and any expansion of $\mathcal{F}_K$ terminates in polynomial time. Furthermore, deciding whether $U \hookrightarrow \mathcal{F}$ is polynomial in the size of $\mathcal{F}$ by simple methods. As a consequence,

Theorem 5 *For a knowledge base K in $\mathcal{SHIQ}$ and a union of conjunctive queries U in which all roles are simple, deciding $K \models U$ is in* coNP *w.r.t. data complexity.*

Matching coNP-hardness follows from the respective result for $\mathcal{ALE}$ [24], which has been extended later to DLs even less expressive than $\mathcal{AL}$ [7]. Thus we obtain the following main result.

Theorem 6 *On knowledge bases in any DL from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering unions of conjunctive queries in which all roles are simple is* coNP*-complete w.r.t. data complexity.*

This result not only exactly characterizes the data complexity of UCQs for a range of DLs, but also extends two previous coNP-completeness results w.r.t. data complexity which are not obvious: the result on CQs over $\mathcal{ALCNR}$ given in [20] to $\mathcal{SHIQ}$, and the result in [18] for atomic queries in $\mathcal{SHIQ}$ to UCQs.

5 Discussion and Conclusion

By the correspondence between query containment and query answering [1], our algorithm can also be applied to decide containment of two UCQs over a $\mathcal{SHIQ}$ knowledge

base. Also, the technique is applicable to the DL $\mathcal{SHOIQ}$, which extends $\mathcal{SHIQ}$ with *nominals*, i.e., concepts denoting single individuals, by tuning of the $\mathcal{SHOIQ}$ tableaux rules [14]. It remains open whether the proposed technique can be applied to even more expressive logics, for example, containing reflexive-transitive closure in the TBox (in the style of PDL), or to more expressive query languages.

Several issues remain for further work. The restriction of the query languages to allow only simple roles is also adopted in [18]. To the best of our knowledge, it is yet unknown whether conjunctive query answering remains decidable in $\mathcal{SHIQ}$ when this constraint is relaxed. It remains unclear whether the algorithm which we have presented here can be exploited, since the presence of transitive roles imposes difficulties in establishing a bound on the depth of completion forests which need to be considered for answering a given query. Also the combined complexity of UCQs remains for further study. As follows from [17], it is in 2ExpTime, and thus Theorem 4 gives not a tight bound. However, we can use a more relaxed blocking condition in our algorithm, where the root of the blocked tree is not necessarily an ancestor of the blocked one. This lowers the worst-case size of a forest exponentially, leading to a co-2NExpTime upper bound, which is not optimal either. We want to point out that such blocking could be similarly used in the standard tableau algorithms for satisfiability checking, but might be inconvenient from an implementation perspective.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[2] 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, 2003.

[3] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[4] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10, pages 349–372.

[5] D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [2], chapter 5, pages 178–218.

[6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[9] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[11] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[12] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[13] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[14] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.

[15] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In D. McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[16] I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.

[17] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of the 11th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, pages 21–35, 2004.

[18] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.

[19] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[20] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[21] M. M. Ortiz de la Fuente, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, Mar. 2006. Available at `http://www.inf.unibz.it/~calvanese/papers/orti-calv-eite-TR-2006-03.pdf%`.

[22] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[23] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278, 1993.

[24] A. Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1994.

[25] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[26] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.