Answer Set Programs Challenged by Ontologies*

Magdalena Ortiz, Sanja Pavlović, and Mantas Šimkus

Institute of Logic and Computation, TU Wien, Austria

Abstract. We introduce *resilient logic programs* that couple a non-monotonic logic program and a first-order theory or description logic (DL) ontology. Unlike previous hybrid languages, where the interaction between the program and the theory is limited to consistency or query entailment tests, in RLPs answers must be 'resilient' to the models of the ontology, allowing non-output predicates to respond differently to different models. RLPs can elegantly express ∃∀∃-QBFs, disjunctive ASP, and configuration problems under incompleteness of information. To guarantee decidability, we use a novel relaxation of DL-safeness that safeguards rules via predicates whose extensions can be inferred to have a finite bound. We present algorithms and tight complexity results for the case where ontologies are written in some prototypical DLs.

1 Introduction

Description Logics (DLs) [2,3] and *rule-based languages*—especially the ones supporting (non-monotonic) default negation—offer complementary modeling and reasoning capabilities. Most DLs are syntactic variants of first-order logic that are suitable for *open-world* reasoning, especially for reasoning about *anonymous objects*, i.e. objects whose identity is unknown but whose existence is implied by domain knowledge. In contrast, rule-based languages like *Answer Set Programming (ASP)*[9] are tailored to provide powerful *closed-world* reasoning about *known* objects, and features like the default negation are important when modeling dynamic domains, e.g., in reasoning about actions and change.

Motivated by this complementarity, combining DLs and rule-based languages into the so-called *Hybrid Knowledge Bases* (HKBs) is a well-established research topic in KR&R [21,22,10,17,14]. Despite the existence of several different languages for expressing HKBs, they can be divided into two classes: the *model-centric* and the *entailment-centric* approaches. The languages in [21,22,4] have model-centric semantics because an intended structure (i.e., an answer set or stable model) is a *single* first-order structure that is "acceptable" both to the rule and to the DL component of the input HKB. That is, the rules base their inferences on a given model of the DL component, and consistency with that model is the only very coarse way in which they access the *knowledge* captured in the DL part. The entailment-centric approaches like [10,17] are the other extreme: the rules do ontological reasoning by accessing the logical consequences of the DL component, but have basically no access to its different models.

* Funded by the Austrian Science Fund (FWF) projects P30360, P30873 and W1255.

For many KR problems both extremes are inadequate. We thus study HKBs that blur the lines between the model-centric and the entailment-centric approaches: different models of the input ontology may be processed in different ways, in the model-centric spirit, but the intended answer sets are defined via universal quantification over the models of the ontology, as in entailment-centric approaches. For a simple (synthetic) example, assume we want to generate a directed graph G from a given set of nodes $n_1, \ldots, n_k$, so that removing one node from the graph will always lead to graph that is strongly connected. Intuitively, here selecting which edges to include in G depends on the universal quantification over all induced subgraphs G' obtained by removing a single node from G, while the reachability relation in G' is different for different choices of G'.

In our formalism, *resilient logic programs (RLPs)*, a standard ASP program $\mathcal{P}$ is paired with a first-order theory (or a DL ontology) $\mathcal{T}$ that challenges it. Predicates are divided into *output*, *response*, and *open-world* predicates. The semantics is defined via a "negotiation" between $\mathcal{P}$ and $\mathcal{T}$: the two components need to agree on *answer sets* I over the output signature, so that no matter how I is extended into a model of $\mathcal{T}$ (by interpreting the open-world predicates), the program $\mathcal{P}$ can give a matching and justified interpretation to the response predicates. Both $\exists\forall\exists$-QBFs and *disjunctive* ASP are naturally captured by RLPs, and in fact, the QBF reduction shows that reasoning is Σ_3^P-hard in data complexity, setting RLPs apart form previous hybrid languages. We illustrate that RLPs are powerful for solving incompletely specified configuration problems.

Inevitably, reasoning in RLPs is undecidable unless restrictions are imposed on how rules are allowed to manipulate anonymous objects. While the usual notion of DL-safeness can be used [18,21], we instead propose a significant relaxation based on the following intuition. Assume that each employee of a company can take part in at most 5 projects, and that all projects have at least one employee; this is expressed in DLs as $\mathsf{Empl} \sqsubseteq \leq 5\ \mathsf{assgdTo.Proj}$ and $\mathsf{Proj} \sqsubseteq \exists \mathsf{assgdTo}^-.\mathsf{Empl}$. If we happen to know that the company has n employees, we can infer that there are at most $5n$ projects. Our relaxed safeness uses the ontology to identify concept names whose extension is forced to be relatively small, and lets the rules access their extensions as ordinary individuals, even if they may contain anonymous objects. E.g., if Empl is assumed to be complete, the rules can treat projects as safe. This idea of safeness is novel to the best of our knowledge, and seems useful in many settings. For the case where ontologies are in the DLs $\mathcal{ALCHOIQ}$, $\mathcal{ALCHI}$ and $\mathsf{DL\text{-}Lite}_{\mathcal{F}}$, we provide algorithms and complexity results.

Due to space limitations some constructions and proofs are provided the extended version of this paper [1].

2 Logic Programming Preliminaries

We assume countably infinite and mutually disjoint sets $\mathbf{S}_{\text{const}}$, $\mathbf{S}_{\text{var}}$, and $\mathbf{S}_{\text{pred}}$ of *constants*, *variables*, and *predicate symbols*, respectively. Each $p \in \mathbf{S}_{\text{pred}}$ is associated with a non-negative arity, denoted by $art(p)$. A *term* is a variable or

[1] `https://dbai.tuwien.ac.at/staff/simkus/papers/dl2019-long.pdf`

a constant, and an *atom* is an expression $p(t_1,\ldots,t_n)$, where $p \in \mathbf{S}_{\text{pred}}$ has arity n, and $t_1,\ldots,t_n$ are terms. A *program* is a set of *rules* of the form

$$r: \quad h \leftarrow b_1,\ldots,b_n, \mathit{not}\ b_{n+1},\ldots,\mathit{not}\ b_m$$

where $n,m \geq 0$, $h,b_1,\ldots,b_m$ are atoms, and each variable that occurs in $h,b_{n+1},\ldots b_m$ must occur in some $b_1,\ldots,b_n$. We let $\mathit{head}(r)=\{h\}$, $\mathit{body}^+(r)=\{b_1,\ldots,b_n\}$, and $\mathit{body}^-(r)=\{b_{n+1},\ldots,b_m\}$. If p is an atom, we call an expression of the form $\mathit{not}\ p$ a *negated atom*. A *literal* is an atom or a negated atom. A rule r is *positive* if $|\mathit{body}^-(r)|=0$. A program is positive if all its rules are positive. We call an atom, a rule, or a program *ground* if it contains no variables. *Facts* are ground rules of the form $h \leftarrow$, where " $\leftarrow$ " is often omitted. We let $\mathsf{adom}(\mathcal{P})$ be the set of constants in $\mathcal{P}$. Given a set of constants $C \subseteq \mathbf{S}_{\text{const}}$, the *grounding of a rule w.r.t.* C, in symbols $\mathit{ground}(r,C)$, is a set of rules obtained from r by uniformly replacing variables in r by all possible elements from C. The *grounding of a program w.r.t.* C is then given as $\mathit{ground}(\mathcal{P},C)=\bigcup_{r\in\mathcal{P}}\mathit{ground}(r,C)$.

An *(Herbrand) interpretation* over $\Sigma \subseteq \mathbf{S}_{\text{pred}}$ is a set of ground atoms using only predicates from Σ (if Σ is not specified we assume $\Sigma = \mathbf{S}_{\text{pred}}$). Given an interpretation I and a set $\Sigma \subseteq \mathbf{S}_{\text{pred}}$, we let $I_{|\Sigma}=\{p(\boldsymbol{u}) \mid p(\boldsymbol{u}) \in I \text{ and } p \in \Sigma\}$.

An interpretation I is a *model* of a ground positive program $\mathcal{P}$ if $\mathit{body}^+(r) \subseteq I$ implies $\mathit{head}(r) \cap I \neq \emptyset$, for each rule $r \in \mathcal{P}$. Further, I is a *minimal model* of $\mathcal{P}$ if there exists no $J \subset I$ that is a model of $\mathcal{P}$. The semantics to programs with negation is given using a program transformation due to Gelfond and Lifschitz [12]. Given an interpretation I and a program $\mathcal{P}$, we define a *reduct* of $\mathcal{P}$ w.r.t. I as $\mathcal{P}^I=\{\mathit{head}(r) \leftarrow \mathit{body}^+(r) \mid \mathit{body}^-(r) \cap I = \emptyset, r \in \mathit{ground}(\mathcal{P},\mathbf{S}_{\text{const}})\}$. We say that I is a *stable model* (or an *answer set*) of $\mathcal{P}$ if I is a minimal model of $\mathcal{P}^I$.

3 Resilient Logic Programs

We introduce our formalism. Syntactically, it pairs a program with a function-free first order logic (FO) theory—as usual in hybrid languages—and also defines a partition of the signature, which is relevant for our *resilient* semantics.

Given a program $\mathcal{P}$ and a theory $\mathcal{T}$, we use $\mathsf{sig}(\mathcal{P})$ and $\mathsf{sig}(\mathcal{T})$ to denote sets of predicate symbols that occur in $\mathcal{P}$ and $\mathcal{T}$, respectively.

Definition 1 (Syntax). A *resilient logic program* (RLP) is a tuple $\Pi = (\mathcal{P},\mathcal{T},\Sigma_{\mathsf{out}},\Sigma_{\mathsf{owa}},\Sigma_{\mathsf{re}})$ where $\mathcal{P}$ is a program, $\mathcal{T}$ is an FO theory, and the sets Σ_{out}, Σ_{owa}, Σ_{re} are a partition of $\mathsf{sig}(\mathcal{P}) \cup \mathsf{sig}(\mathcal{T})$ with $\Sigma_{\mathsf{re}} \cap \mathsf{sig}(\mathcal{T}) = \emptyset$. We call Σ_{out} the set of *output predicates*, Σ_{owa} the *open predicates*, and Σ_{re} the *response predicates* of Π. The predicates in $\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{re}}$ are called *closed predicates* of Π.

Our semantics uses the following generalized definition of a reduct. It is inherited from *Clopen KBs* [4], which in turn borrow from *r-hybrid KBs* [21].

Given a program $\mathcal{P}$, an interpretation I, and $\Sigma \subseteq \mathbf{S}_{\text{pred}}$, the reduct $\mathcal{P}^{I,\Sigma}$ of $\mathcal{P}$ w.r.t. I and Σ is the positive program obtained from $\mathit{ground}(\mathcal{P},\mathbf{S}_{\text{const}})$ by:

1. Deleting every rule r that contains a literal $p(\boldsymbol{u})$ such that

   (a) $p(\boldsymbol{u}) \in \mathit{body}^+(r)$, $p(\boldsymbol{u}) \notin I$, and $p \in \Sigma$,
   (b) $p(\boldsymbol{u}) \in \mathit{head}(r)$, $p(\boldsymbol{u}) \in I$, and $p \in \Sigma$, or
   (c) $p(\boldsymbol{u}) \in \mathit{body}^-(r)$ and $p(\boldsymbol{u}) \in I$.

2. In the remaining rules, deleting all negated atoms and atoms $p(\boldsymbol{u})$ with $p \in \Sigma$.

Intuitively, $\mathcal{P}^{I,\Sigma}$ is the result of partially evaluating $\mathcal{P}$ according to the facts in I, under the open-world assumption for the predicates in Σ.

Now we can define the semantics of RLPs. For convenience, we adopt the standard Herbrand semantics of FO theories.

Definition 2 (Semantics). Let $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ be an RLP, and let I be an interpretation over Σ_{out}. Then I is an *answer set* of Π if

(i) there exists some model J of $\mathcal{T}$ such that $I_{|\Sigma_{\mathsf{out}}} = J_{|\Sigma_{\mathsf{out}}}$, and

(ii) for each model J of $\mathcal{T}$ with $I_{|\Sigma_{\mathsf{out}}} = J_{|\Sigma_{\mathsf{out}}}$, there is an interpretation H such that $J_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{owa}}} = H_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{owa}}}$ and $H_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{re}}}$ is a minimal model of $\mathcal{P}^{H,\Sigma_{\mathsf{owa}}}$.

We call H a *response to J w.r.t. I* and Π.

The main reasoning task for RLPs is *answer set existence*, i.e., given an RLP Π, does there exist an answer set I of Π. Other common reasoning problems like *skeptical* (resp., *brave*) *entailment* can be easily reduced to checking non-existence (resp., existence) of an answer set. In particular, a ground atom $p(\boldsymbol{c})$ is true in all answer sets of Π (i.e. $p(\boldsymbol{c})$ is a skeptical consequence) iff the result of adding $\leftarrow p(\boldsymbol{c})$ to (the program component of) Π does not have an answer set.[2] Similarly, a ground atom $p(\boldsymbol{c})$ is true in some answer set of Π (i.e. $p(\boldsymbol{c})$ is a brave consequence) iff Π augmented with $\leftarrow \mathit{not}\ p(\boldsymbol{c})$ does have an answer set.

We illustrate RLPs. For brevity, we use $h_1|\cdots|h_k \leftarrow \beta$ as a shorthand for the set of rules $\{h_i \leftarrow \beta, \mathit{not}\ h_1, \dots \mathit{not}\ h_{i-1}, \mathit{not}\ h_{i+1}, \dots, \mathit{not}\ h_k \mid 1 \leq i \leq k\}$. This *choice rule* tells us that at least one of $h_1, \cdots, h_k$ must be true in case β is true.

Example 1. We show how to express the graph problem from the introduction as an RLP. Given nodes $n_1, \dots, n_k$, let $\Pi = (\mathcal{P}, \mathcal{T}, \{V, E\}, \{\overline{E}, R\}, \{\mathit{in}, \mathit{out}\})$, where $\mathcal{T} = \{\forall x(V(x) \rightarrow \mathit{in}(x) \vee \mathit{out}(x)), \forall x(V(x) \rightarrow \neg \mathit{in}(x) \vee \neg \mathit{out}(x)), \exists x\, \mathit{out}(x), \forall x \forall y\ \mathit{out}(x) \wedge \mathit{out}(y) \rightarrow x = y\}$ and the program $\mathcal{P}$ consists of the following rules:

$$V(n_1), \cdots V(n_k), \qquad E(x,y)|\overline{E}(x,y) \leftarrow, \qquad R(x,z) \leftarrow R(x,y), R(y,z),$$
$$R(x,y) \leftarrow E(x,y), \mathit{not}\ \mathit{out}(x), \mathit{not}\ \mathit{out}(y)$$
$$\leftarrow V(x), V(y), x \neq y, \mathit{not}\ \mathit{out}(x), \mathit{not}\ \mathit{out}(y), \mathit{not}\ R(x,y)$$

Then each answer set of Π defines a directed graph G such that the following is true for all nodes n_i, $1 \leq i \leq k$: if n_i is removed from G, the remaining graph is still strongly connected.

[2] The *constraint* '$\leftarrow \beta$' is a shorthand for '$p \leftarrow \beta, \mathit{not}\ p$', where p is a fresh atom.

Example 2. Consider the evaluation problem for quantified Boolean formulas (QBFs) of the form $\Phi = \exists X_1, \ldots, X_n \forall Y_1, \ldots, Y_m, \exists Z_1, \ldots, Z_k \varphi.$, where φ is in 3-CNF. The quantifier alternation resembles the semantics of our RLPs, as reflected in the following RLP Π, which has an answer set iff Φ evaluates to true.

We let $\Pi = (\mathcal{P}, \mathcal{T}, \{V_X, V_Y, V_Z, T_X, F_X\}, \{T_Y, F_Y\}, \{T_Z, F_Z\})$ with

$$\mathcal{T} = \{\ \forall x(V_Y(x) \to T_Y(x) \lor F_Y(x)), \quad \forall x(V_Y(x) \land T_Y(x) \land F_Y(x) \to \bot)\}$$

$$\begin{aligned}\mathcal{P} = \{\ & V_X(c_1^X), \ldots, V_X(c_n^X), \quad V_Y(c_1^Y), \ldots, V_Y(c_m^Y), \quad V_Z(c_1^Z), \ldots, V_Z(c_k^Z) \\ & T_X(x)|F_X(x) \leftarrow V_X(x), \qquad T_Z(x)|F_Z(x) \leftarrow V_Z(x), \\ & \leftarrow \sigma(L_{i,1}), \sigma(L_{i,2}), \sigma(L_{i,3}), \text{ for each clause } C_i \text{ in } \varphi\}\end{aligned}$$

where, given a literal l, $\sigma(l)$ is defined as:

$$\sigma(l) = \begin{cases} T_\alpha(c_i^\alpha) & \text{if } l = \neg\alpha_i, i = 1, \ldots, n_\alpha \\ F_\alpha(c_i^\alpha) & \text{if } l = \alpha_i, i = 1, \ldots, n_\alpha \end{cases}$$

where $\alpha \in \{X, Y, Z\}$, $n_X = n, n_Y = m$, and $n_Z = k$.

In the extended version, we use a slightly modified reduction to show that answer set existence is Σ_p^3-hard in data-complexity for many classes of RLPs.

We note that there is a strong connection between RLPs and *disjunctive* logic programs with negation under the answer set semantics [8]. In particular, there is a polynomial time translation from the latter programs into RLPs, which not only preserves the answer sets, but is also *data-independent*, i.e. a disjunctive program $\mathcal{P}$ is converted into an RLP Π such that $\mathcal{P}$ and Π have the same answer sets under any addition of facts. This translation, which further demonstrates the power of RLPs, is presented in the extended version.

4 Resilient Logic Programs with DL Theories

This section focuses on RLPs whose theory is a DL TBox. We consider $\mathcal{ALCHI}$, $\mathcal{ALCHOIQ}$, and DL-Lite$_\mathcal{F}$, defined as usual. We assume $\mathbf{S}_{\text{cn}} \cup \mathbf{S}_{\text{rn}} \subseteq \mathbf{S}_{\text{pred}}$ and $\mathbf{S}_{\text{in}} \subseteq \mathbf{S}_{\text{const}}$ where $\mathbf{S}_{\text{cn}}$, $\mathbf{S}_{\text{rn}}$, and $\mathbf{S}_{\text{in}}$ denote respectively the set of concept names, the set of role names, and the set of individual names. As for FO theories, also for DLs we use Herbrand interpretations. Recall that an Herbrand interpretation I can be viewed as an ordinary DL interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I} = \mathbf{S}_{\text{const}}$ and $p^\mathcal{I} = \{\boldsymbol{c} \mid p(\boldsymbol{c}) \in I\}$ for each $p \in \mathbf{S}_{\text{cn}} \cup \mathbf{S}_{\text{rn}}$.

RLPs in their general form are undecidable. This is not hard to show adapting analogous results for some well-known hybrid languages [15,23]. In order to regain decidability, we introduce a *safeness* condition that ensures that variables in the program range only over a finite number of constants, which is one of the key pieces for devising a terminating algorithm for answer set existence for RLPs (see Section 5). To this end, we use the notion of safeness presented in [4] which was inspired by the well-known *DL-safeness* [18,21]:

Definition 3. A resilient logic program $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ fulfills the *safeness condition* if for each rule $r \in \mathcal{P}$: each variable x that occurs in r must occur in some atom $p(\boldsymbol{u}) \in \mathit{body}^+(r)$ where p is a closed predicate.

This restriction ensures that variables only take on those constants that are explicitly mentioned in $\mathcal{P}$. However, this is often too strong. We relax it by safeguarding variables with positive literals whose predicate is a *bounded concept name*.

Definition 4. Let $\mathcal{T}$ be an $\mathcal{ALCHOIQ}$ TBox, $\Sigma \subseteq \mathbf{S}_{\text{pred}}$ be a finite set of predicate symbols, and $\mathcal{N} = \{\{c\} \mid c \text{ occurs in } \mathcal{T}\}$. The set $\mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$ of *bounded concept names* in $\mathcal{T}$ w.r.t. Σ is the smallest set such that:

1. if A is a concept name in $\Sigma \cap \mathsf{sig}(\mathcal{T})$ then $A \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$
2. if A is a concept name in $\mathsf{sig}(\mathcal{T})$ and there is a role name P in $\mathsf{sig}(\mathcal{T}) \cap \Sigma$ s.t. $\mathcal{T} \models A \sqsubseteq \exists P.\top$ or $\mathcal{T} \models A \sqsubseteq \exists P^-.\top$, then $A \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$
3. if A is a concept name in $\mathsf{sig}(\mathcal{T})$ and $\mathcal{T} \models A \sqsubseteq \bigsqcup_{B \in \mathcal{B} \cup \mathcal{N}} B$, then $A \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$
4. if A is a concept name in $\mathsf{sig}(\mathcal{T})$ and there exists $B \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma) \cup \mathcal{N}$, a role R occurring in $\mathcal{T}$, and integers $n, m \geq 0$ s.t. $\mathcal{T} \models A \sqsubseteq \geq mR.B$ and $\mathcal{T} \models B \sqsubseteq \leq nR^-.A$, then $A \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$

If $\mathcal{T}$ is in $\mathcal{ALCHI}$, item 4. in the definition above is omitted and $\mathcal{N} = \emptyset$. Similarly, if $\mathcal{T}$ is in $\mathsf{DL\text{-}Lite}_{\mathcal{F}}$, item 3. is omitted and item 4. is replaced by the following:

4a. if A is a concept name in $\mathsf{sig}(\mathcal{T})$ and there exists $B \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$ and a role R occurring in $\mathcal{T}$ s.t. $\mathcal{T} \models A \sqsubseteq \exists R^-$, $\mathcal{T} \models \exists R \sqsubseteq B$, and $(\text{funct } R) \in \mathcal{T}$, then $A \in \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$.

Note that, given $\mathcal{T}$ and Σ, computing the set $\mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$ requires a polynomial number of steps, some of which use an entailment test as an oracle. A similar computation for bounded concepts is done in [11] for $\mathsf{DL\text{-}Lite}_{\mathcal{F}}$. The idea behind bounded concept names is as follows. Assume we know which elements are in the extensions of predicates in Σ, these predicates are interpreted under the closed-world view and thus uniquely fixed once an interpretation I is given. For any concept name in $\mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$ we can compute a relatively small upper bound on the number of possible elements in the extension of this concept name in any model of $\mathcal{T}$ that agrees with I on Σ. Hence, although we might not know the exact constants that occur in the extensions of bounded concept names in such models, we know that there is finitely many of them. Thus, safeguarding rule variables with positive literals using bounded concept names still results in the variables ranging only over a finite amount of constants.

Proposition 1. *Let $\mathcal{T}$ be a TBox in one of the considered DLs, $\Sigma \subseteq \boldsymbol{S}_{pred}$, and $b \geq 0$. Then, we can compute $b' \geq 0$ s.t. $|\{c \mid A(c) \in J, A \in \boldsymbol{B}_{cn}(\mathcal{T}, \Sigma)\}| \leq b'$ holds in every model J of $\mathcal{T}$ in which $|\{c \mid p(\mathbf{c}) \in J, p \in \Sigma, c \text{ occurs in } \mathbf{c}\}| \leq b$. If $\mathcal{T}$ is in $\mathcal{ALCHOIQ}$, b' is at most exponential in the size of $\mathcal{T}$. For $\mathcal{ALCHI}$ and $\mathsf{DL\text{-}Lite}_{\mathcal{F}}$, b' is polynomial in the size of $\mathcal{T}$.*

We remark that the definition above is incomplete, in the sense that $\mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma)$ need not contain all concept names for which a bound exists. Nonetheless, it is sufficient to relax our previous notion of safeness. To this end, given a program $\mathcal{P}$, for each $p \in \mathsf{sig}(\mathcal{P})$ and $1 \leq i \leq art(p)$, we define a *position* $p[i]$. Given a set of predicates Σ, the set $\mathsf{ap}(\mathcal{P}, \Sigma)$ of *affected positions* (see [7]) in $\mathcal{P}$ w.r.t. Σ is inductively defined as follows: (i) $p[i] \in \mathsf{ap}(\mathcal{P}, \Sigma)$, for $p \in \Sigma$ and $1 \leq i \leq art(p)$, and (ii) if there exists a rule $r \in \mathcal{P}$ s.t. a variable x appears in $body^+(r)$ only in affected positions and x appears in $head(r)$ in position π, then $\pi \in \mathsf{ap}(\mathcal{P}, \Sigma)$.

A predicate symbol occurring in an RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ is called a *bounded predicate* if it is (i) a closed predicate or (ii) in $\mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}})$. We now relax the safeness condition from Definition 3 as follows:

Definition 5. An RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ fulfills the *relaxed safeness condition* if the following holds for each rule $r \in \mathcal{P}$: each variable in r must also occur in some atom $p(\boldsymbol{u}) \in body^+(r)$ where p is a bounded predicate in Π and, additionally, $q[i] \notin \mathsf{ap}(\mathcal{P}, \mathbf{B}_{\text{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}}) \setminus \Sigma_{\mathsf{out}})$, for all $q \in \Sigma_{\mathsf{out}}$ and $1 \leq i \leq art(q)$.

Example 3. Assume a company wants to process a certain amount of customer orders per day. Incoming orders will consist of tasks, and the tasks will be associated with services that are offered by the company (active services). The company needs to decide which services it should offer, so that *for all the possible configurations* of received orders (including the tasks they comprise and the services they require, which are not yet known), they can schedule tasks to employees in such a way that all tasks are finished in the working day.

This problem is solved by an RLP where the models of the theory correspond to possible configurations of received orders, and the models of the program define viable schedules of tasks. Its answer sets correspond to sets of services that, if activated, guarantee that for any configuration of received orders there is a schedule in which each task will be completed before the end of the working day. We first model the time line of a work day as follows:

$$\begin{aligned}
\mathcal{P}_1 = \{&\mathit{Next}(\mathit{0}, \mathit{1}), \mathit{Next}(\mathit{1}, \mathit{2}), \ldots, \mathit{Next}(t_{max} - \mathit{1}, t_{max})\\
&\mathit{Time}(x) \leftarrow \mathit{Next}(x, y), \quad \mathit{Time}(y) \leftarrow \mathit{Next}(x, y),\\
&\mathit{Earlier}(x, y) \leftarrow \mathit{Next}(x, y), \quad \mathit{Earlier}(x, z) \leftarrow \mathit{Earlier}(x, y), \mathit{Earlier}(y, z),\\
&\mathit{less10}(x_0, x_n) \leftarrow \mathit{Next}(x_0, x_1), \ldots, \mathit{Next}(x_{n-1}, x_n), \text{for } 0 \leq n < 10\}
\end{aligned}$$

$\mathcal{P}_1$ also contains rules for *less30* and *less60*, defined similarly as for *less10*. Assuming that employees work eight hours per day and that the granularity of *Next* is one minute, we set $t_{max} = 480$.

As facts, we store our employees (*Employee*), the services that could be offered (*Service*), as well as which employee can provide which service (*Provides*). We also store the length of each service. For simplicity reasons we consider three types of services: short, medium, and long, captured by unary relations *Short*, *Medium*, and *Long*, respectively. Short services take ten minutes, medium-length services 30 minutes, and long services takes 60 minutes to be completed. Assume our company receives two orders per day. We also encode this information using facts. Finally, it is reasonable to assume that for a given order there might be services that need to be completed before other services can be performed. We store this information using a binary relation *ServiceBefore* and once again, facts. Consider, for demonstration purposes, the following set of facts:

$$\begin{aligned}\mathcal{P}_2 = \{&\mathit{Service}(s_1), \mathit{Service}(s_2), \mathit{Service}(s_3), \mathit{Employee}(e_1), \mathit{Employee}(e_2),\\ &\mathit{Provides}(e_1, s1), \mathit{Provides}(e_1, s_2), \mathit{Provides}(e_2, s_1), \mathit{Provides}(e_2, s_3),\\ &\mathit{Short}(s_1), \mathit{Medium}(s_2), \mathit{Long}(s_3), \mathit{Order}(o_1), \mathit{Order}(o_2), \mathit{Before}(s_1, s_3)\}\end{aligned}$$

We do not know how orders will comprise tasks, and which services these tasks will require. We only know that orders consist of at least one and at most five tasks (*Task*), each belonging to exactly one order and requiring (*Req*) one service. The possible configurations of orders are given by the models of this theory:

$$\begin{aligned}\mathcal{T} = \{&\mathit{Order} \sqsubseteq\ \geq 1 \mathit{hasTask}.\top \sqcap \leq 5 \mathit{hasTask}.\top, \quad \geq 1 \mathit{Req}^-.\top \sqsubseteq \mathit{ActiveService}\\ &\mathit{Task} \sqsubseteq\ = 1 \mathit{hasTask}^-.\mathit{Order} \quad \mathit{Task} \sqsubseteq\ = 1 \mathit{Req}.\top, \quad \geq 1 \mathit{hasTask}^-.\top \sqsubseteq \mathit{Task}\},\end{aligned}$$

The rules that select services to be provided are defined as follows:

$$\mathcal{P}_3 = \{\mathit{ActiveService}(x) | \mathit{InactiveService}(x) \leftarrow \mathit{Service}(x)\}$$

We look for schedules (*Sched*) consisting of tuples (x, y, z), assigning task y to employee x to be performed starting at time point z. These rules populate the response predicates, trying to find viable schedules for the different models of the theory. Due to space restrictions, we show only the rules for short services.

$$\begin{aligned}\mathcal{P}_4 = \{\mathit{Sched}(x, y, z) &\leftarrow \mathit{Task}(y), \mathit{Req}(y, u), \mathit{Provides}(x, u), \mathit{Time}(z), \mathit{not}\ \mathit{Illegal}(x, y, z)\\ \mathit{Illegal}(x, y, z) &\leftarrow \mathit{Task}(y), \mathit{Req}(y, u), \mathit{Short}(u), \mathit{Time}(z), \mathit{less10}(z, t_{max}), \mathit{Employee}(x)\\ \mathit{Illegal}(x, y, z) &\leftarrow \mathit{Sched}(x', y, z'), \mathit{less10}(z', z), \mathit{Req}(y, u), \mathit{Short}(u), \mathit{Employee}(x), x \neq x'\\ \mathit{Illegal}(x, y, z) &\leftarrow \mathit{Sched}(x, y, z'), \mathit{less10}(z', z), \mathit{Req}(y, u), \mathit{Short}(u), z \neq z'\\ \mathit{Illegal}(x, y, z) &\leftarrow \mathit{Task}(y), \mathit{Sched}(x, y', z'), \mathit{less10}(z', z), \mathit{Req}(y', u), \mathit{Short}(u), y \neq y'\\ \mathit{FaultyServ}(x') &\leftarrow \mathit{Order}(z), \mathit{Task}(y), \mathit{Task}(y'), \mathit{hasTask}(z, y), \mathit{hasTask}(z, y'), \mathit{Req}(y, x),\\ & \mathit{Req}(y', x'), \mathit{Before}(x, x')\mathit{Sched}(w, y, t), \mathit{Sched}(w', y', t'), \mathit{Earlier}(t', t)\\ \mathit{FaultyServ}(x') &\leftarrow \mathit{Order}(z), \mathit{Task}(y), \mathit{Task}(y'), \mathit{hasTask}(z, y), \mathit{hasTask}(z, y'),\\ & \mathit{Sched}(w, y, t), \mathit{Sched}(w', y', t'), \mathit{Req}(y, x),\\ & \mathit{Short}(x), \mathit{Req}(y', x'), \mathit{Before}(x, x'), \mathit{less10}(t, t')\\ \mathit{OKService}(x) &\leftarrow \mathit{not}\ \mathit{FaultyServ}(x), \mathit{Service}(x) \qquad\qquad \mathit{OKTask}(y) \leftarrow \mathit{Sched}(x, y, z)\\ &\leftarrow \mathit{not}\ \mathit{OKService}(x), \mathit{ActiveService}(x) \quad \leftarrow \mathit{not}\ \mathit{OKTask}(y), \mathit{Task}(y)\}\end{aligned}$$

Let $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4$ and $\Sigma_{\mathsf{out}} = \{\mathit{Order}, \mathit{ActiveService}\}$. The RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \mathsf{sig}(\mathcal{T}) \setminus \Sigma_{\mathsf{out}}, \mathsf{sig}(\mathcal{P}) \setminus (\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{owa}}))$ fulfills the relaxed safeness condition and its answer sets represent sets of services that can be offered and completed within the given time constraint regardless of the type of received orders. E.g., $I = \{\mathit{ActiveService}(s_1), \mathit{ActiveService}(s_2),\ \mathit{Order}(o_1), \mathit{Order}(o_2)\}$ is an answer set of Π. In the worst case, both orders consist of five tasks each that in turn require a medium-length service s_2. A valid schedule in this case is: $\mathit{Sched}(e_1, y_1, 0)$, $\mathit{Sched}(e_1, y_2, 20) \ldots, \mathit{Sched}(e_1, y_{10}, 180)$. Note that, unlike traditional ASP, we can have comparable answer sets, e.g., $\{\mathit{ActiveService}(s_1), \mathit{Order}(o_1), \mathit{Order}(o_2)\}$ is also an answer set of Π.

Consider $J = \{\mathit{ActiveService}(s_3), \mathit{Order}(o_1), \mathit{Order}(o_2)\}$ and a model of $\mathcal{T}$ in which each order consist of five tasks associated with the long service s_3. Since only employee e_2 can perform s_3, she would have to perform it ten times, thus taking more than 480 minutes. Hence, there is no valid schedule and J is not an

5 Decidability and Complexity Results

Algorithm 1 decides answer set existence for a given $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$. In a nutshell, it guesses a candidate interpretation I and subsequently uses two oracle calls to check whether I is an answer set of Π. The first call checks whether there are any models of $\mathcal{T}$ that agree with I on Σ_{out}. The second call tries to verify whether each such model of $\mathcal{T}$ has a response w.r.t. I and Π by attempting to guess a model J of $\mathcal{T}$ for which this does not hold. In order to check that J has no response, another oracle call is made which attempts to find a response by guessing it. If this fails, J is a counter example to I being an answer set of Π.

Assuming RLPs fulfill (relaxed) safeness condition, we have the following:

Proposition 2. *Let $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ be an RLP and let I be an answer set of Π. For every $c \in \boldsymbol{S}_{const}$ occurring in I, $c \in \mathsf{adom}(\mathcal{P})$ holds.*

Proposition 3. *Let $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$ be an RLP , I and J be interpretations over Σ and $\mathsf{sig}(\mathcal{T})$, resp. Let $J' = \{p(\boldsymbol{c}) \in J \mid p \in \Sigma_{\mathsf{owa}} \cap \mathsf{sig}(\mathcal{P}), \boldsymbol{c} \in \Delta^{art(p)}\}$, where Δ is the set of constants that occur in $\mathcal{P}$ or in some $q(\boldsymbol{a}) \in J$ with $q \in \boldsymbol{B}_{cn}(\mathcal{T}, \Sigma_{\mathsf{out}})$. Then J and J' have the same responses w.r.t. I and Π.*

Due to Proposition 2, the maximum number of different constants occurring in some answer set of Π is bounded by $|\mathsf{adom}(\mathcal{P})|$. In view of Proposition 1, we can thus compute an integer b that limits the amount of different constants that can be in the extension of any concept name in $\mathbf{B}_{\mathrm{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}})$ in any model of $\mathcal{T}$ that agrees with some answer set of Π on Σ_{out}, referred to as a *relevant model* of $\mathcal{T}$. We construct a set Δ of size b consisting of $\mathsf{adom}(\mathcal{P})$, constants occurring in $\mathcal{T}$ and additionally of fresh constants giving a name to each anonymous element that could hypothetically occur in the extension of some $A \in \mathbf{B}_{\mathrm{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}})$ in some relevant model of $\mathcal{T}$. Checking whether there are models of $\mathcal{T}$ that agree with I on Σ_{out} is a well-known problem in the literature and it boils down to checking the consistency of a description logic knowledge base $(\mathcal{T}, I)$ in the presence of closed predicates from Σ_{out}, where I is seen as an ABox. In view of Proposition 3, to find a model of $\mathcal{T}$ with no response we only guess the extensions of open predicates that occur in $\mathcal{P}$ using the constants from Δ. We next check whether our partial guess J actually corresponds to some model of $\mathcal{T}$ and if so, we check whether there is a response H to J. Due to our safeness condition, for this guess it suffices to consider only the constants from J and $\mathsf{adom}(\mathcal{P})$. Note that to verify whether H is a response to J, we need not fully compute the possibly infinite reduct $\mathcal{P}^{H,\Sigma_{\mathsf{owa}}}$. Instead, we use the reduct $ground(\mathcal{P}, \Delta)^{H,\Sigma_{\mathsf{owa}}}$ which is guaranteed to be finite and has the same minimal model as the original reduct.

As reasoning in the considered logics in the presence of closed predicate is decidable [24,19], it is easy to verify that Algorithm 1 terminates. Moreover, by analysis of the algorithm we can obtain the complexity bounds listed in Table 1.

```
Algorithm hasAnswerSet($\Pi$)
    Input   : An RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$
    Output : true iff $\Pi$ has an answer
    Guess an interpretation $I$ over $\Sigma_{\mathsf{out}}$ using constants from $\mathsf{adom}(\mathcal{P})$
    $\mathcal{B} \leftarrow \mathbf{B}_{\mathrm{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}})$
    $b \leftarrow$ max number of different const. in extensions of bounded concept names
    $\Delta \leftarrow \mathsf{adom}(\mathcal{P}) \cup \{c \mid c$ occurs in $\mathcal{T}\} \cup \{a_1, \dots, a_{b-b'}\}$, where
      $b' = |\mathsf{adom}(\mathcal{P})| + |\{c \mid c$ occurs in $\mathcal{T}\}|$
    return isConsistent($\mathcal{T}, I, \Sigma_{\mathsf{out}}$) and not hasCE($I, \Pi, \Delta, \mathcal{B}$)

Subroutine hasCE($I, \Pi, \Delta, \mathcal{B}$)
    Input   : An interpretation $I$, an RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$,
              $\Delta \subseteq \mathbf{S}_{\mathrm{const}}$, and $\mathcal{B} \subseteq \mathsf{sig}(\mathcal{T})$
    Output : true iff there is a counter example to $I$ being an answer set of $\Pi$
    Guess an interpretation $J$ over $\mathsf{sig}(\mathcal{P}) \cap \mathsf{sig}(\mathcal{T}) \cup \Sigma_{\mathsf{out}}$ and constants from $\Delta$
      s.t. $J_{|\Sigma_{\mathsf{out}}} = I_{|\Sigma_{\mathsf{out}}}$
    $J' \leftarrow J \cup \{\neg p(\boldsymbol{c}) \mid \boldsymbol{c} \in \Delta^{art(p)}, p \in \mathsf{sig}(\mathcal{P}) \cap (\mathsf{sig}(\mathcal{T}) \setminus \mathcal{B}),$ and $p(\boldsymbol{c}) \notin J\}$
    return isConsistent($\mathcal{T}, J', \mathcal{B}$) and not hasResp($J, \Pi$)

Subroutine hasResp($J, \Pi$)
    Input   : An interpretation $J$ and an RLP $\Pi = (\mathcal{P}, \mathcal{T}, \Sigma_{\mathsf{out}}, \Sigma_{\mathsf{owa}}, \Sigma_{\mathsf{re}})$
    Output : true iff there is a response for $J$ w.r.t. $J_{|\Sigma_{\mathsf{out}}}$ and $\Pi$
    Guess an interpretation $H$ over $\mathsf{sig}(\mathcal{P})$ and the constants from $J$ and
      $\mathsf{adom}(\mathcal{P})$ s.t. $H_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{owa}}} = J$
    if $H_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{re}}}$ is a min. model of $\mathcal{P}^{H, \Sigma_{\mathsf{owa}}}$ then return true
    else return false
```

Algorithm 1: Answer set existence of RLPs. Here isConsistent($\mathcal{T}, I, \Sigma$) checks consistency of DL KB ($\mathcal{T}, I$) with closed predicates Σ.

We briefly guide the reader through the entries in this table. Observe that for predicates of arbitrary arities, I is at most exponential in $|\mathcal{P}|$, and consider the logic $\mathcal{ALCHI}$. In view of Proposition 1, Δ is polynomial in $|\Pi|$ and thus there are only exponentially many different guesses for J. As $\mathcal{ALCHI}$ with closed predicates is ExpTime-complete [24,25], we modify the algorithm to do the following in exponential time: guess I, compute $\mathbf{B}_{\mathrm{cn}}(\mathcal{T}, \Sigma_{\mathsf{out}})$, construct Δ and check whether there is a model of $\mathcal{T}$ that agrees with I on Σ_{out}. If so, iterate through possible guesses for J and check for each whether it corresponds to some model. If so, call an oracle to determine whether there is a response H to J. Due to unbounded predicate arities, both H and the grounding *ground*($\mathcal{P}, \Delta$) may be at most of exponential size. Thus, we can guess H, compute the corresponding reduct and verify whether $H_{|\Sigma_{\mathsf{out}} \cup \Sigma_{\mathsf{re}}}$ is its minimal model in exponential time. Observe that, by the standard padding argument [20,1], this oracle can be implemented as an NP oracle, and so the overall complexity of the algorithm is NExpTime$^{\mathrm{NP}}$. If predicate arities are bounded, instead of guessing I and H we simply iterate through all the possibilities. The ExpTime upper bound follows. Data complexity follows from the fact that consistency checking in $\mathcal{ALCHI}$ with closed predicates is in NP in terms of data complexity [16]. Hence, each part of the original algorithm can be implemented as an NP oracle and the complexity result is

	$\mathcal{ALCHI}$	DL-Lite$_{\mathcal{F}}$	$\mathcal{ALCHOIQ}$
combined complexity	NExpTime$^{\text{NP}}$- c	NExpTime$^{\text{NP}}$- c	NExpTime$^{\text{NExpTime}}$
combined complexity with bounded predicate arities	ExpTime - c	Σ_3^P- c	NP$^{\text{NExpTime}}$
data complexity	Σ_3^P- c	Σ_3^P- c	NP$^{\text{NExpTime}}$

Table 1. Complexity of answer set existence for RLPs (c denotes completeness results).

immediate. As DL-Lite$_{\mathcal{F}}$ with closed predicates is in NP both in data and in combined complexity [11,13], arguments for DL-Lite$_{\mathcal{F}}$ are virtually the same as for $\mathcal{ALCHI}$. The difference is that in the case of bounded predicate arities, we obtain the Σ_3^P upper bound due to lower combined complexity of DL-Lite$_{\mathcal{F}}$ with closed predicates. For $\mathcal{ALCHOIQ}$, the constructed set of constants Δ is at most exponential in $|\mathcal{T}|$, and consistency checking of $\mathcal{ALCHOIQ}$ knowledge bases (with closed predicates) is NExpTime-complete [25], it suffices to iterate through possible guesses for J (exponentially many in $|\Pi|$) and have a single NExpTime oracle to check for consistency and the existence of response to J. The upper bound follows. Similar reasoning is used for the other two $\mathcal{ALCHOIQ}$ bounds.

All bounds for DL-Lite$_{\mathcal{F}}$ and $\mathcal{ALCHI}$ are tight. ExpTime-hardness for $\mathcal{ALCHI}$ in the case of bounded predicate arities is due to the same hardness of $\mathcal{ALCHI}$ with closed predicates. NExpTime$^{\text{NP}}$-hardness in the case of unbounded predicate arities is obtained by a reduction from answer set existence problem for disjunctive datalog which is complete for this class. Finally, Σ_p^3-hardness is shown by a reduction from $\exists\forall\exists$-QBFs to RLPs with $\mathcal{ALCHI}$/DL-Lite$_{\mathcal{F}}$ theories, given in the extended version. The presented data complexity bound for $\mathcal{ALCHOIQ}$ is inherited from the combined complexity, as the data complexity of $\mathcal{ALCHOIQ}$ with closed predicates is unknown, and it is likely that it is not optimal.

6 Discussion

We note that the decidability of the answer set existence problem under plain safety (see Definition 3) easily generalizes from DLs to other fragments of first-order logic. The only requirement for such fragments is the decidability of theories in the presence of closed predicates, which is enjoyed, e.g., by the *guarded negation fragment (GNFO)* [5] (see [6,4] for the treatment of closed predicates in GNFO).

In this paper, DL ontologies are interpreted over Herbrand interpretations (whose domain is a fixed infinite set of constants). This was done for mathematical clarity of the semantics of RLPs, but it has some side-effects, e.g., by ruling out ontology models with a finite domain. This can be easily fixed using a more complicated definition that handles two kinds of interpretations: Herbrand interpretations for rules, and ordinary first-order structures for DL ontologies.

The are several questions for future research. We plan to investigate the complexity of RLPs under various restrictions on rules. In particular, we believe that disallowing default negation in front of response predicates will often lead to lower computational complexity. We also plan to study rewritability of RLPs into standard disjunctive ASP, for which efficient implementations exist.

References

1. Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York, NY, USA, 2003.
3. Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic.* Cambridge University Press, 2017.
4. Labinot Bajraktari, Magdalena Ortiz, and Mantas Šimkus. Combining rules and ontologies into Clopen knowledge bases. In *Proc. of AAAI 2018*, 2018.
5. Vince Bárány, Balder Ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, June 2015.
6. Michael Benedikt, Pierre Bourhis, Balder ten Cate, and Gabriele Puppis. Querying visible and invisible information. In *Proc. of LICS 2016*, pages 297–306. ACM, 2016.
7. Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res. (JAIR)*, 48:115–174, 2013.
8. Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
9. Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web*, volume 5689 of *LNCS*, pages 40–110. Springer, 2009.
10. Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artif. Intell.*, 172(12-13):p. 1495, 2008.
11. Enrico Franconi, Yazmin Angélica Ibáñez-García, and Inanç Seylan. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.*, 278:71–84, 2011.
12. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP/SLP 1988*, pages 1070–1080. MIT Press, 1988.
13. Tomasz Gogacz, Víctor Gutiérrez-Basulto, Yazmin Angélica Ibáñez-García, Filip Murlak, Magdalena Ortiz, and Mantas Šimkus. Ontology focusing: Knowledge-enriched databases on demand, 2019. Available at `http://arxiv.org/abs/1904.00195`.
14. Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.*, 175(9-10):1528–1554, 2011.
15. Alon Y. Levy and Marie-Christine Rousset. Combining horn rules and description logics in CARIN. *Artif. Intell.*, 104(1-2):165–209, 1998.
16. Carsten Lutz, Inanç Seylan, and Frank Wolter. Ontology-mediated queries with closed predicates. In *Proc. of IJCAI 2015*, pages 3120–3126. AAAI Press, 2015.
17. Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.
18. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.
19. Nhung Ngo, Magdalena Ortiz, and Mantas Šimkus. Closed predicates in description logics: Results on combined complexity. In *Proc. of KR 2016*, 2016.
20. Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

21. Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.*, 3(1):61–73, 2005.
22. Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. of KR 2006*. AAAI Press, 2006.
23. Riccardo Rosati. The limits of querying ontologies. In *Proc. of ICDT 2007*, volume 4353 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2007.
24. Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over dboxes. pages 923–925, 2009.
25. Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.