Querying OWL 2 QL and Non-monotonic Rules

Matthias Knorr and José Júlio Alferes

CENTRIA, FCT, Universidade Nova de Lisboa

Abstract. Answering (conjunctive) queries is an important reasoning task in Description Logics (DL), hence also in highly expressive ontology languages, such as OWL. Extending such ontology languages with rules, such as those expressible in RIF-Core, and further with non-monotonic rules, integrating default negation as described in the RIF-FLD, yields an even more expressive language that allows for modeling defaults, exceptions, and integrity constraints.

Here, we present a top-down procedure for querying knowledge bases (KB) that combine non-monotonic rules with an ontology in $DL\text{-}Lite_{\mathcal{R}}$ – the DL underlying the OWL 2 profile OWL 2 QL. This profile aims particularly at answering queries in an efficient way for KB with large ABoxes. Our procedure extends the query-answering facility to KB that also include non-monotonic rules, while maintaining tractability of reasoning (w.r.t. data complexity). We show that the answers are sound and complete w.r.t. the well-founded MKNF model for hybrid MKNF KB $\mathcal{K}$.

1 Introduction

Combining highly expressive ontology languages, such as OWL [9], (or their underlying DL) and rules, such as in RIF-Core [3], is an important task in the on-going standardization driven by the World Wide Web Consortium[1] (W3C). Both languages are quite different in terms of expressiveness and how decidability is achieved, and providing a joint formalism is non-trivial, all the more if the rules include mechanisms for non-monotonic reasoning, such as the `NAF` operator described in the RIF-FLD [4].

Non-monotonic rules provide expressive features, such as the ability to model defaults, exceptions, and integrity constraints, and its usefulness is frequently being voiced. E.g., in [14], an ontology language is used for matching clinical trials criteria with patient records, but one open problem is that medication of patients, which is fully known, should be modeled by defaults.

Several approaches that combine rules and DL have been defined (see, e.g., [11,13] for an overview) but, among them, the approach of hybrid MKNF knowledge bases [13], which is based on the logics of minimal knowledge and negation as failure (MKNF) [12], is one of the most advanced. The integration of monotonic and non-monotonic reasoning is seamless yet allows for a modular re-use of reasoning algorithms of each of its components. Thus, hybrid MKNF is more expressive than comparable approaches but at least as competitive in terms of

[1] `http://www.w3.org/`

L. Aroyo et al. (Eds.): ISWC 2011, Part I, LNCS 7031, pp. 338–353, 2011.

computation. At the cost of having a weaker form of reasoning, the well-founded MKNF semantics for hybrid MKNF knowledge bases [11] achieves an even lower computational (data) complexity in general. For example, if reasoning in the DL is in PTime, then the computation of the well-founded MKNF model remains polynomial, while [13] is in NP. This is clearly preferable for large applications, such as the one described in [14], which uses data of over 240,000 patients.

A further improvement in efficiency can be achieved if we query for information in a top-down manner: instead of computing the entire model of a knowledge base we could, e.g., just query for the medication of one patient ignoring all the others. Queries are considered in the W3C with SPARQL [15] and answering (conjunctive) queries is an important reasoning task in DL [7].

In [1], **SLG($\mathcal{O}$)** is introduced, which allows us to pose queries to a hybrid MKNF KB and whose semantics is shown to correspond to that of [11]. The work is based on an extension of SLG – a procedure for query-answering in normal logic programs under the well-founded semantics resorting to tabling techniques – that besides the operations for resolution in the rules, also incorporates calls to a parametric oracle that deals with the reasoning task in the DL part of the KB. It is shown that, if the number of answers of the oracle is appropriately restricted, the favorable computational complexity of [11] is maintained. However, it is not spelled out how these conditions are achieved for a concrete DL.

In this paper, we present a top-down procedure based on **SLG($\mathcal{O}$)** for querying KB that combine non-monotonic rules with an ontology in *DL-Lite*$_\mathcal{R}$ – the DL underlying the OWL 2 profile OWL 2 QL. This profile aims particularly at answering queries in an efficient way for KB with large ABoxes. It is thus a natural choice as DL language for a procedure for query answering in KB that, besides the ontology with large ABoxes, also includes the features of expressing defaults, constraints and exceptions, provided by non-monotonic rules. Our procedure achieves that, while maintaining tractability of reasoning (w.r.t. data complexity) on hybrid KB and reasoning in LogSpace on *DL-Lite*$_\mathcal{R}$ alone. In particular, query-answering is obtained by a combination of techniques from top-down procedures in logic programs, and reasoning in relational databases as done in *DL-Lite*$_\mathcal{R}$. We show that the answers are sound and complete w.r.t. the well-founded MKNF model for hybrid MKNF knowledge bases $\mathcal{K}$ if $\mathcal{K}$ is MKNF-consistent, and a paraconsistent approximation of that model otherwise. As such, our work provides a way for querying KB consisting of an ontology in the profile OWL 2 QL and a RIF dialect that allows for non-monotonic rules, and, together with [10], such a query procedure for each tractable OWL profile.

2 Preliminaries

2.1 *DL-Lite*$_\mathcal{R}$

The description logic underlying OWL 2 QL is *DL-Lite*$_\mathcal{R}$, one language of the *DL-Lite* family [5], which we recall in the following.

The syntax of *DL-Lite*$_\mathcal{R}$ is based on three disjoint sets of individual names $\mathsf{N_I}$, concept names $\mathsf{N_C}$, and role names $\mathsf{N_R}$. Complex concepts and roles can be

formed according to the following syntax, where $A \in \mathsf{N_C}$ is a concept name, $P \in \mathsf{N_R}$ a role name, and P^- its inverse.

$$C \longrightarrow A \mid \exists R \qquad R \longrightarrow P \mid P^- \qquad D \longrightarrow C \mid \neg C \qquad E \longrightarrow R \mid \neg R$$

A *DL-Lite*$_\mathcal{R}$ knowledge base $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The TBox contains general inclusion axioms (GCI) of the form $C \sqsubseteq D$, where C and D are defined as above. Thus, the left and right hand sides of GCI are of different expressiveness in *DL-Lite*$_\mathcal{R}$. Additionally, *DL-Lite*$_\mathcal{R}$ TBoxes contain role inclusion axioms (RI) of the form $R \sqsubseteq E$ where R and E are formed as introduced above. Such axioms permit to express properties, such as symmetry.

The ABox contains assertions of the form $A(a)$ and $P(a, b)$ where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. Assertions $C(a)$ for general concepts C are included by adding $A \sqsubseteq C$ to the TBox and $A(a)$ to the ABox for a new concept name A.

The semantics of *DL-Lite*$_\mathcal{R}$ is based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a nonempty interpretation domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns to each individual a a distinct[2] element $a^\mathcal{I}$ of $\Delta^\mathcal{I}$, to each concept C a subset $C^\mathcal{I}$, and to each role name R a binary relation $R^\mathcal{I}$ over $\mathcal{I}$. This can be generalized to complex expressions as usual:

$$(P^-)^\mathcal{I} = \{(i_2, i_1) \mid (i_1, i_2) \in P^\mathcal{I}\} \qquad (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$$
$$(\exists R)^\mathcal{I} = \{i \mid (i, i') \in R^\mathcal{I}\} \qquad (\neg R)^\mathcal{I} = \Delta \times \Delta \setminus R^\mathcal{I}$$

An interpretation $\mathcal{I}$ is a model of the GCI $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. It is a model of an RI $R \sqsubseteq E$ if $R^\mathcal{I} \subseteq E^\mathcal{I}$. $\mathcal{I}$ is also a model of an assertion $A(a)$ ($P(a, b)$) if $a^\mathcal{I} \in A^\mathcal{I}$ ($(a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}$). Given an axiom/assertion α we denote by $\mathcal{I} \models \alpha$ that $\mathcal{I}$ is a model of α. A model of a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \alpha$ holds for all $\alpha \in \mathcal{T}$ and all $\alpha \in \mathcal{A}$. A KB $\mathcal{O}$ is satisfiable if it has at least one model.

Standard reasoning tasks in *DL-Lite*$_\mathcal{R}$ are polynomial in the size of the TBox, and in LogSpace in the size of the ABox, i.e., in data complexity. The same holds for answering conjunctive queries, but if we consider the combined complexity (including the query), then answering conjunctive queries is NP-complete [5].

2.2 Well-Founded Semantics for Hybrid MKNF

Hybrid MKNF knowledge bases are introduced in [13] as a combination of nonmonotonic rules and a DL that is translatable into first-order logic, and in which standard reasoning tasks, namely satisfiability and instance checking, are decidable. Here, we recall only the version with rules without disjunction in heads and the computation of the complete well-founded MKNF model for such knowledge bases [11], for which we define a top-down procedure for *DL-Lite*$_\mathcal{R}$. Recalling the computation of the complete well-founded MKNF model is not strictly needed for the definition of the top-down procedure itself, but we present it here since

[2] Hence, the unique name assumption is applied and, as shown in [2], dropping it would increase significantly the computational complexity of *DL-Lite*$_\mathcal{R}$.

it provides valuable insights into the combined model for which we query. Moreover, the operator $D_{\mathcal{K}_G}$, defined in this section, provides exactly the counterpart of the oracle to the DL in the top-down procedure we present in this paper.

Definition 1. *Let $\mathcal{O}$ be a DL knowledge base. A function-free first-order atom $P(t_1, \ldots, t_n)$ such that P is $\approx$ or occurs in $\mathcal{O}$ is called a DL-atom; otherwise it is called non-DL-atom. An* MKNF rule *r has the following form where H, A_i, and B_i are function-free first-order atoms:*

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m \tag{1}$$

K-*atoms (resp.* **not**-*atoms) are atoms with a leading* **K** *(resp.* **not***). A program is a finite set of MKNF rules, and a* hybrid MKNF knowledge base *$\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$ and* positive *if $m = 0$ holds for all MKNF rules in $\mathcal{K}$. The* ground instantiation *of $\mathcal{K}$ is the KB $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ where $\mathcal{P}_G$ is obtained from $\mathcal{P}$ by replacing each rule r of $\mathcal{P}$ with a set of rules substituting each variable in r with constants from $\mathcal{K}$ in all possible ways.*

There is no restriction on the interaction between $\mathcal{O}$ and $\mathcal{P}$, i.e., DL-atoms may appear anywhere in the rules.

The semantics of $\mathcal{K}$ is based on a transformation of $\mathcal{K}$ into an MKNF formula to which the MKNF semantics can be applied (see [11,12,13] for details). Decidability is achieved by applying the well-known notion of DL-safety, in which each variable in a rule appears in at least one non-DL **K**-atom [13]. This essentially restricts the application of rules to individuals explicitly appearing in the knowledge base in consideration [13]. Instead of spelling out the technical details of the original MKNF semantics [13] or its three-valued counterpart [11], we focus on a compact representation of models for which the computation of the well-founded MKNF model is defined[3]. This representation is based on a set of **K**-atoms and $\pi(\mathcal{O})$, the translation of $\mathcal{O}$ into first-order logic.

Definition 2. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base. The* set of **K**-atoms of *$\mathcal{K}_G$, written $\mathsf{KA}(\mathcal{K}_G)$, is the smallest set that contains (i) all ground* **K**-*atoms occurring in $\mathcal{P}_G$, and (ii) a* **K**-*atom $\mathbf{K}\xi$ for each ground* **not**-*atom $\mathbf{not}\,\xi$ occurring in $\mathcal{P}_G$. For a subset S of $\mathsf{KA}(\mathcal{K}_G)$, the* objective knowledge *of S w.r.t. $\mathcal{K}_G$ is the set of first-order formulas $\mathsf{OB}_{\mathcal{O},S} = \{\pi(\mathcal{O})\} \cup \{\xi \mid \mathbf{K}\,\xi \in S\}$.*

The set $\mathsf{KA}(\mathcal{K}_G)$ contains all atoms occurring in $\mathcal{K}_G$, only with **not**-atoms substituted by corresponding modal **K**-atoms, while $\mathsf{OB}_{\mathcal{O},S}$ provides a first-order representation of $\mathcal{O}$ together with a set of known/derived facts.

In the three-valued MKNF semantics, this set of **K**-atoms can be divided into true, undefined and false modal atoms. Next, we recall operators from [11] that derive consequences based on $\mathcal{K}_G$ and a set of **K**-atoms that is considered to hold. To further simplify notation, in the remainder of the paper we abuse notation and consider all operators **K** implicit.

[3] Strictly speaking, this computation yields the so-called well-founded partition from which the well-founded MKNF model is defined (see [11] for details).

Definition 3. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a positive, ground hybrid MKNF knowledge base. The operators $R_{\mathcal{K}_G}$, $D_{\mathcal{K}_G}$, and $T_{\mathcal{K}_G}$ are defined on subsets of $\mathsf{KA}(\mathcal{K}_G)$:*

$$\begin{aligned} R_{\mathcal{K}_G}(S) =& \{H \mid \mathcal{P}_G \text{ contains a rule of the form } H \leftarrow A_1, \dots A_n \\ & \text{such that, for all } i,\ 1 \leq i \leq n, A_i \in S\} \\ D_{\mathcal{K}_G}(S) =& \{\xi \mid \xi \in \mathsf{KA}(\mathcal{K}_G) \text{ and } \mathsf{OB}_{\mathcal{O},S} \models \xi\} \\ T_{\mathcal{K}_G}(S) =& R_{\mathcal{K}_G}(S) \cup D_{\mathcal{K}_G}(S) \end{aligned}$$

The operator $T_{\mathcal{K}_G}$ is monotonic, and thus has a least fixpoint $T_{\mathcal{K}_G} \uparrow \omega$. Transformations can be defined that turn an arbitrary hybrid MKNF KB $\mathcal{K}_G$ into a positive one (respecting the given set S) to which $T_{\mathcal{K}_G}$ can be applied. To ensure coherence, i.e., that classical negation in the DL enforces default negation in the rules, two slightly different transformations are defined (see [11] for details).

Definition 4. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \mathsf{KA}(\mathcal{K}_G)$. The* MKNF transform *$\mathcal{K}_G/S$ is defined as $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$, where $\mathcal{P}_G/S$ contains all rules $H \leftarrow A_1, \dots, A_n$ for which there exists a rule $H \leftarrow A_1, \dots, A_n, \mathbf{not}\ B_1, \dots, \mathbf{not}\ B_m$ in $\mathcal{P}_G$ with $B_j \notin S$ for all $1 \leq j \leq m$. The* MKNF-coherent transform *$\mathcal{K}_G//S$ is defined as $\mathcal{K}_G//S = (\mathcal{O}, \mathcal{P}_G//S)$, where $\mathcal{P}_G//S$ contains all rules $H \leftarrow A_1, \dots, A_n$ for which there exists a rule $H \leftarrow A_1, \dots, A_n, \mathbf{not}\ B_1, \dots, \mathbf{not}\ B_m$ in $\mathcal{P}_G$ with $B_j \notin S$ for all $1 \leq j \leq m$ and $\mathsf{OB}_{\mathcal{O},S} \not\models \neg H$. We define $\Gamma_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G/S} \uparrow \omega$ and $\Gamma'_{\mathcal{K}_G}(S) = T_{\mathcal{K}_G//S} \uparrow \omega$.*

Based on these two antitonic operators [11], two sequences $\mathbf{P}_i$ and $\mathbf{N}_i$ are defined, which correspond to the true and non-false derivations.

$$\begin{aligned} \mathbf{P}_0 &= \emptyset & \mathbf{N}_0 &= \mathsf{KA}(\mathcal{K}_G) \\ \mathbf{P}_{n+1} &= \Gamma_{\mathcal{K}_G}(\mathbf{N}_n) & \mathbf{N}_{n+1} &= \Gamma'_{\mathcal{K}_G}(\mathbf{P}_n) \\ \mathbf{P}_\omega &= \bigcup \mathbf{P}_i & \mathbf{N}_\omega &= \bigcap \mathbf{N}_i \end{aligned}$$

The fixpoints, which are reached after finitely many iterations, yield the well-founded MKNF model [11].

Definition 5. *The* well-founded MKNF model *of an MKNF-consistent ground hybrid MKNF knowledge base $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ is defined as $(\mathbf{P}_\omega, \mathsf{KA}(\mathcal{K}_G) \setminus \mathbf{N}_\omega)$.*

If $\mathcal{K}_G$ is MKNF-inconsistent, then there is no MKNF model, hence no well-founded MKNF model.

We use a simple example adapted from [5] to illustrate this computation.

Example 1. Consider the hybrid MKNF KB $\mathcal{K}$[4] consisting of $\mathcal{O}$:

$$\begin{aligned} \texttt{Professor} &\sqsubseteq \exists \texttt{TeachesTo} & \texttt{Student} &\sqsubseteq \exists \texttt{HasTutor} \\ \exists \texttt{TeachesTo}^- &\sqsubseteq \texttt{Student} & \exists \texttt{HasTutor}^- &\sqsubseteq \texttt{Professor} \\ \texttt{Professor} &\sqsubseteq \neg \texttt{Student} & \texttt{HasTutor}^- &\sqsubseteq \texttt{TeachesTo} \end{aligned}$$

and the rules (including the facts):

[4] We use capital letters for DL-atoms and individuals. Note that $\mathcal{K}$ is not DL-safe, but we assume that each rule contains implicitly, for all variables $\mathtt{x}$ appearing in the rule, an additional atom $\mathtt{o}(\mathtt{x})$, and that $\mathcal{K}$ contains facts $\mathtt{o}(\mathtt{x})$, for every object x.

$$\texttt{hasKnownTutor(x)} \leftarrow \texttt{Student(x)}, \texttt{HasTutor(x,y)} \quad (2)$$

$$\texttt{hasUnknownTutor(x)} \leftarrow \texttt{Student(x)}, \mathbf{not}\ \texttt{HasTutor(x,y)} \quad (3)$$

$$\texttt{Student(Paul)} \leftarrow \quad \texttt{HasTutor(Jane,Mary)} \leftarrow \quad \texttt{TeachesTo(Mary,Bill)} \leftarrow$$

We only consider the set $\mathsf{KA}(\mathcal{K}_G)$ for the computation, i.e., only the atoms that actually appear in the ground rules and we abbreviate the names in this example appropriately. Starting with $\mathbf{P}_0 = \emptyset$ and $\mathbf{N}_0 = \mathsf{KA}(\mathcal{K}_G)$, we compute $\mathbf{P}_1 = \{\texttt{S(P)}, \texttt{HT(J,M)}, \texttt{TT(M,B)}, \texttt{S(J)}, \texttt{S(B)}, \texttt{HT(B,M)}, \texttt{hKT(J)}, \texttt{hKT(B)}\}$ and $\mathbf{N}_1 = \mathbf{P}_1 \cup \{\texttt{hUT(P)}, \texttt{hUT(J)}, \texttt{hUT(B)}\}$. We continue with $\mathbf{P}_2 = \mathbf{P}_1 \cup \{\texttt{hUT(P)}\}$ and $\mathbf{N}_2 = \mathbf{P}_2$, and these are already the fixpoints.We obtain that `Jane` and `Bill` have a known tutor, while `Paul` has not. Other derivations, such as `Professor(Mary)` can be obtained from the fixpoints and $\mathcal{O}$. Note that rule (3) can be understood as a default: it states that by default students have unknown tutors.

3 Queries in Hybrid MKNF

In [1], a procedure, called **SLG($\mathcal{O}$)**, is defined for querying arbitrary hybrid MKNF knowledge bases. This procedure extends SLG resolution with tabling [6] with an oracle to $\mathcal{O}$ that handles ground queries to the DL-part of $\mathcal{K}_G$: given the already derived information and $\mathcal{O}$, the oracle returns a (possibly empty) set of atoms that allows us, together with $\mathcal{O}$ and the already derived information, to derive the queried atom. Recalling the full procedure would be beyond the scope of this paper, so we just give an intuitive overview of the general procedure and only point out the specific technical details that are required for the concrete oracle to *DL-Lite*$_\mathcal{R}$ in terms of an interface. All the details can be found at [1].

The general idea is that **SLG($\mathcal{O}$)** creates a forest of derivation trees starting with the initial query. If the initial query is unifiable with a rule head, then the rule body is added as a child node to the respective tree. In each such new node, we consider each atom according to a given selection order, create new trees, and try to resolve these so-called goals. Due to potential circularities in the derivations, some goals may be delayed, possibly giving rise to conditional answers. If we achieve an unconditional answer for such a goal, i.e., a leaf node without any remaining goals, then the (instantiated) atom in the root is true, and the goal is resolved. If no further operation is applicable, but no leaf node is empty, then the root(s) corresponding to such a (set of) tree(s) is (are) considered false. If a (default) negated atom **not** A is selected, then a tree for A is created: if A succeeds then **not** A fails; otherwise **not** A is resolved.

If the queried/selected atom is a DL-atom, then a query is posed to an oracle, which encapsulates reasoning in the DL part of the KB, to check whether the DL-atom can be derived from the DL-part. For that purpose, all DL-atoms that are already known to be true in the derivation forest are added to the ontology before calling the oracle. It may happen that only a subsequent derivation step contains the necessary information to derive the queried atom, so we would have to call the oracle over and over again. To avoid this problem, rather than

answering whether the queried atom is derivable, an oracle in **SLG**($\mathcal{O}$) returns a set (conjunction) of atoms that, if proven true, would allow us to derive the queried atom. This set is added as a child to the respective tree and treated like the result of an operation on rules.

In the end, if we achieve an unconditional answer for the initial query, the (instantiated) query is true; if all answers are conditional, then the query is undefined; otherwise it is false. The following example shall clarify the idea.

Example 2. Consider again the hybrid MKNF knowledge base $\mathcal{K}$ from Example 1 and the query $\mathtt{Student(Bill)}$. A root $\mathtt{Student(Bill)}$ is created and since no rule head is unifiable, we call the oracle. If $\mathtt{TeachesTo(x, Bill)}$ holds for some x, then $\mathtt{Student(Bill)}$ would hold. Thus a child with the goal $\mathtt{TeachesTo(x, Bill)}$ is added to the tree. Then, a new tree for $\mathtt{TeachesTo(x, Bill)}$ is created whose root unifies with $\mathtt{TeachesTo(Mary, Bill)}$. We obtain an unconditional answer, and, after resolving $\mathtt{TeachesTo(x, Bill)}$ in the tree with root $\mathtt{Student(Bill)}$, an unconditional answer for $\mathtt{Student(Bill)}$, thus finishing the derivation.

Alternatively, consider the query $\mathtt{hUT(P)}$ (with abbreviations as in Example 1). Since the root of the corresponding tree is a non-DL-atom, only rules can be considered, and, in fact, rule (3) is applicable. The resulting child contains two goals, namely $\mathtt{S(P)}$ and **not** $\mathtt{HT(P, y)}$. Note that implicitly $\mathtt{o(y)}$ also occurs so that $\mathtt{y}$ is ground when querying for **not** $\mathtt{HT(P, y)}$. A separate tree is created for $\mathtt{S(P)}$ and easily resolved with the given fact. Then we consider any of the meanwhile grounded **not** $\mathtt{HT(P, y)}$. A new tree is created for $\mathtt{HT(P, y)}$ (with ground $\mathtt{y}$) but neither the rules nor the oracle allow us to derive an unconditional answer and eventually each such tree is failed, since there is no known tutor for $\mathtt{P}$. If the tree for $\mathtt{HT(P, y)}$ fails, then **not** $\mathtt{HT(P, y)}$ holds and can be resolved, which yields an unconditional answer for $\mathtt{hUT(P)}$.

As already said, we only want to consider specifically the mechanism that provides the oracle for *DL-Lite*$_\mathcal{R}$, and here we only recall the relevant notions for that task. We start by defining the reasoning task in consideration, namely DL-safe conjunctive queries.

Definition 6. *A* (DL-safe) conjunctive query q *is a non-empty set, i.e. conjunction, of literals where each variable in* q *occurs in at least one non-DL atom in* q*. We also write* q *as a rule* $q(X_i) \leftarrow A_1, \ldots, A_n, \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$ *where* X_i *is the (possibly empty) set of variables, appearing in the body, which are requested.*

This guarantees that **SLG**($\mathcal{O}$) does always pose ground queries to the oracle, avoiding problems with DL where conjunctive query answering is undecidable in general, and, in particular, with inconsistent *DL-Lite*$_\mathcal{R}$ KB that would simply return arbitrary solutions [5] in opposite to our intentions (see Definition 8).

Now, we recall the definition of a complete oracle that provides the relation for the intended derivation. We point out that $\mathcal{F}_n$ is the current derivation forest, where n is increased with each applied **SLG**($\mathcal{O}$) operation. In it, $I_{\mathcal{F}_n}$ corresponds to all already derived (true or false) atoms in a concrete derivation forest.

Definition 7. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base, S a ground goal, L a set of ground atoms that appear in at least one rule head in $\mathcal{P}_G$, and $I^+_{\mathcal{F}_n} = I_{\mathcal{F}_n} \setminus \{\mathbf{not}\, A \mid \mathbf{not}\, A \in I_{\mathcal{F}_n}\}$. The* complete oracle *for $\mathcal{O}$, denoted $compT_{\mathcal{O}}$, is defined by $compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ iff $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models S$.*

This notion of a complete oracle is used to define the **SLG($\mathcal{O}$)** operation that handles the oracle calls to the DL (see [1] for notation of trees in **SLG($\mathcal{O}$)**).

– Oracle Resolution: Let $\mathcal{F}_n$ contain a tree with root node $N = S$:- $|S$, where S is ground. Assume that $compT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, Goals)$. If N does not have a child $N_{child} = S$:- $|Goals$ in $\mathcal{F}_n$ then add N_{child} as a child of N.

It is shown (in Theorem 5.3 of [1]) that answers to queries in **SLG($\mathcal{O}$)** correspond to the hybrid MKNF well-founded model as in [11].

The complete oracle is unfortunately not efficient, in that it potentially creates a lot of superfluous answers, such as supersets of correct minimal answers. This problem is tackled with the introduction of a partial oracle [1].

Definition 8. *Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a hybrid MKNF knowledge base, S a ground goal, and L a set of atoms that are unifiable with at least one rule head in $\mathcal{P}_G$. A* partial oracle *for $\mathcal{K}_G$, denoted $pT_{\mathcal{O}}$, is a relation $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ such that if $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$, then $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models S$ and $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ consistent. A partial oracle $pT_{\mathcal{O}}$ is* correct *w.r.t. $compT_{\mathcal{O}}$ iff, for all MKNF-consistent $\mathcal{K}_G$, replacing $compT_{\mathcal{O}}$ in* **SLG($\mathcal{O}$)** *with $pT_{\mathcal{O}}$ succeeds for exactly the same set of queries.*

There are three main differences between complete and partial oracles. First, in the latter we do not have to consider all possible answers, thus restricting the number of returned answers for a given query. This is important, because, as pointed out in [1], the favorable computational properties of the well-founded MKNF model are only maintained if the number of returned answers is appropriately bounded. Second, only derivations based on a consistent $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ are considered. This removes all pointless attempts to derive an inconsistency for an MKNF-consistent KB $\mathcal{K}$. It also removes derivations based on explosive behavior w.r.t. inconsistencies, which is why the correctness result of the partial oracle is limited to MKNF-consistent KB. This may not be a problem, since it has already been conjectured in [1] that the resulting paraconsistent behavior should be beneficial in practice. Finally, instead of requiring ground sets L, we admit sets of atoms that are unifiable with heads. This simplifies notation and postpones the necessary grounding to be handled in **SLG($\mathcal{O}$)** w.r.t. the rules.

How such a partial oracle is defined for a concrete DL is not specified in [1] and, in Section 5, we present a concrete oracle for $DL\text{-}Lite_{\mathcal{R}}$.

4 Satisfiability in $DL\text{-}Lite_{\mathcal{R}}$

For defining an oracle for $DL\text{-}Lite_{\mathcal{R}}$ we rely on the reasoning algorithms provided for $DL\text{-}Lite_{\mathcal{R}}$ in [5]. The basic reasoning service to which all others are reduced is satisfiability. Satisfiability of a $DL\text{-}Lite_{\mathcal{R}}$ KB is checked by evaluating a suitable

Boolean first-order logic query w.r.t. a canonical model of its ABox. We recall the construction and evaluation of such a formula here specifically, since our work is based on an adaptation of that algorithm. First, we recall the definition of a canonical interpretation of the ABox from [5].

Definition 9. *Let $\mathcal{A}$ be a DL-Lite$_\mathcal{R}$ ABox. By $db(\mathcal{A}) = (\Delta^{db(\mathcal{A})}, \cdot^{db(\mathcal{A})})$ we denote the interpretation defined as follows:*

- *$\Delta^{db(\mathcal{A})}$ is the nonempty set consisting of all constants occurring in $\mathcal{A}$;*
- *$a^{db(\mathcal{A})} = a$ for each constant a;*
- *$A^{db(\mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$ for each atomic concept A; and*
- *$R^{db(\mathcal{A})} = \{(a, b) \mid R(a, b) \in \mathcal{A}\}$ for each atomic role R.*

It is easy to see that $db(\mathcal{A})$ is in fact a model of $\mathcal{A}$ and a minimal one [5].

This forms the basis for checking satisfiability in DL-Lite$_\mathcal{R}$, i.e., we are able to reduce satisfiability to evaluating a Boolean FOL query.

Definition 10. *Satisfiability in DL-Lite$_\mathcal{R}$ is* FOL-reducible *if, for every TBox $\mathcal{T}$ expressed in DL-Lite$_\mathcal{R}$, there exists a Boolean FOL query q, over the alphabet of $\mathcal{T}$, such that, for every nonempty ABox $\mathcal{A}$, $(\mathcal{T}, \mathcal{A})$ is satisfiable if and only if q evaluates to false in $db(\mathcal{A})$.*

Now, a query is constructed, by first splitting all GCI into those with $\neg$ on the right hand side (called *negative inclusions* (NI)) and those without (called *positive inclusions* (PI)) and by considering the negative inclusions as a starting point to compute all derivable negative inclusions.

Definition 11. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox. We call* NI-closure *of $\mathcal{T}$, denoted by $cln(\mathcal{T})$, the set defined inductively as follows:*

1. *All negative inclusion assertions in $\mathcal{T}$ are also in $cln(\mathcal{T})$.*
2. *If $B_1 \sqsubseteq B_2$ is in $\mathcal{T}$ and $B_2 \sqsubseteq \neg B_3$ or $B_3 \sqsubseteq \neg B_2$ is in $cln(\mathcal{T})$, then also $B_1 \sqsubseteq \neg B_3$ is in $cln(\mathcal{T})$.*
3. *If $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $\exists R_2 \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists R_2$ is in $cln(\mathcal{T})$, then also $\exists R_1 \sqsubseteq \neg B$ is in $cln(\mathcal{T})$.*
4. *If $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $\exists R_2^- \sqsubseteq \neg B$ or $B \sqsubseteq \neg \exists R_2^-$ is in $cln(\mathcal{T})$, then also $\exists R_1^- \sqsubseteq \neg B$ is in $cln(\mathcal{T})$.*
5. *If $R_1 \sqsubseteq R_2$ is in $\mathcal{T}$ and $R_2 \sqsubseteq \neg R_3$ or $R_3 \sqsubseteq \neg R_2$ is in $cln(\mathcal{T})$, then also $R_1 \sqsubseteq \neg R_3$ is in $cln(\mathcal{T})$.*
6. *If one of the assertions $\exists R \sqsubseteq \neg \exists R$, $\exists R^- \sqsubseteq \neg \exists R^-$, or $R \sqsubseteq \neg R$ is in $cln(\mathcal{T})$, then all three such assertions are in $cln(\mathcal{T})$.*

A translation function from assertions in $cln(\mathcal{T})$ to FOL formulas is defined [5].

Definition 12. *Let $\mathcal{O}$ be a DL-Lite$_\mathcal{R}$ KB and $cln(\mathcal{T})$ the NI-closure of $\mathcal{T}$. The translation function δ from axioms in $cln(\mathcal{T})$ to first-order formulas is:*

$$\delta(B_1 \sqsubseteq \neg B_2) = \exists x.\gamma_1(x) \wedge \gamma_2(x)$$

$$\delta(R_1 \sqsubseteq \neg R_2) = \exists x, y.\rho_1(x, y) \wedge \rho_2(x, y)$$

where $\gamma_i(x) = A_i(x)$ if $B_i = A_i$, $\gamma_i(x) = \exists y_i.R_i(x, y_i)$ if $B_i = \exists R_i$, and $\gamma_i(x) = \exists y_i.R_i(y_i, x)$ if $B_i = \exists R^-$; and $\rho_i(x, y) = P_i(x, y)$ if $R_i = P_i$, and $\rho_i(x, y) = P_i(y, x)$ if $R_i = P_i^-$.

Require: *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$
Ensure: true if $\mathcal{O}$ is satisfiable, false otherwise
  $q_{unsat} = \bot$
  for all $\alpha \in cln(\mathcal{T})$ **do**
    $q_{unsat} = q_{unsat} \vee \delta(\alpha)$
  end for
  if $q_{unsat}^{db(\mathcal{A})} = \emptyset$ **then**
    return **true**
  else
    return **false**
  end if

Fig. 1. Algorithm Consistent

The algorithm in Fig. 1 checks satisfiability of a *DL-Lite*$_\mathcal{R}$ knowledge base $\mathcal{O}$ by testing if q_{unsat} is not satisfied if evaluated over $db(\mathcal{A})$. Of course, if $\mathcal{O}$ is free of negative inclusions, then the algorithm succeeds automatically.

With this, instance checking is straightforwardly obtained in [5] as a reduction to satisfiability checking.

Theorem 1. *Let $\mathcal{O}$ be DL-Lite$_\mathcal{R}$ KB, and H either a general concept (with ground argument t_i) appearing in $\mathcal{O}$ where $\hat{A}$ an atomic concept not appearing in $\mathcal{O}$ or a role name or its inverse (with ground arguments t_i) appearing in $\mathcal{O}$ and $\hat{A}$ an atomic role not appearing in $\mathcal{O}$. Then $\mathcal{O} \models H(t_i)$ iff $\mathcal{O} \cup \{\hat{A} \sqsubseteq \neg H, \hat{A}(t_i)\}$ is unsatisfiable.*

Note that this theorem is a generalization of two separate theorems for concepts and roles in [5] joined here for reasons of notation.

5 An Oracle for *DL-Lite*$_\mathcal{R}$

The material presented in Section 4 suffices to handle the bottom-up computation of the well-founded MKNF model w.r.t. *DL-Lite*$_\mathcal{R}$. In the subsequent example, which we recall from [5], and which is a modification of Example 1 now without rules and with the facts turned into ABox assertions, we not only present how satisfiability and instance checking work, but also intuitively sketch the solution for defining an oracle.

Example 3. Consider the *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O}$ consisting of the axioms in the TBox:

$$\texttt{Professor} \sqsubseteq \exists \texttt{TeachesTo} \quad (4) \qquad \exists \texttt{HasTutor}^- \sqsubseteq \texttt{Professor} \quad (7)$$

$$\texttt{Student} \sqsubseteq \exists \texttt{HasTutor} \quad (5) \qquad \texttt{Professor} \sqsubseteq \neg \texttt{Student} \quad (8)$$

$$\exists \texttt{TeachesTo}^- \sqsubseteq \texttt{Student} \quad (6) \qquad \texttt{HasTutor}^- \sqsubseteq \texttt{TeachesTo} \quad (9)$$

and the simple ABox:

$$\texttt{Student}(\texttt{Paul}) \quad \texttt{HasTutor}(\texttt{Jane}, \texttt{Mary}) \quad \texttt{TeachesTo}(\texttt{Mary}, \texttt{Bill}). \quad (10)$$

For checking satisfiability, we consider $db(\mathcal{A})$ with $\Delta^{db(\mathcal{A})} = \{\texttt{Paul}, \texttt{Jane}, \texttt{Mary}, \texttt{Bill}\}$ whose elements are all mapped to themselves, and the interpretation of all concept and role names according to $\mathcal{A}$, e.g., $\texttt{Student}^{\texttt{db}(\mathcal{A})} = \{\texttt{Paul}\}$ and $\texttt{Professor}^{\texttt{db}(\mathcal{A})} = \emptyset$.

Then, we compute $cln(\mathcal{T})$ as follows.

$$\texttt{Professor} \sqsubseteq \neg\texttt{Student} \quad (11) \qquad \exists\texttt{TeachesTo} \sqsubseteq \neg\texttt{Student} \quad (14)$$
$$\exists\texttt{HasTutor}^- \sqsubseteq \neg\texttt{Student} \quad (12) \qquad \exists\texttt{HasTutor} \sqsubseteq \neg\texttt{Professor} \quad (15)$$
$$\exists\texttt{TeachesTo}^- \sqsubseteq \neg\texttt{Professor} \quad (13)$$

Axiom 11 occurs in $\mathcal{T}$, Axiom 12 follows from (11) and (7), Axiom 13 follows from (11) and (6), and (14) and (15) follow from (9) and (12), respectively (13).

The translation function δ can be applied to each negative inclusion in $cln(\mathcal{T})$.

$$\delta(\texttt{Professor} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.\texttt{Professor}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}) \quad (16)$$
$$\delta(\texttt{HasTutor}^- \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{y},\texttt{x})) \wedge \texttt{Student}(\texttt{x}) \quad (17)$$
$$\delta(\texttt{TeachesTo}^- \sqsubseteq \neg\texttt{Professor}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \texttt{Professor}(\texttt{x}) \quad (18)$$
$$\delta(\texttt{TeachesTo} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{x},\texttt{y})) \wedge \texttt{Student}(\texttt{x}) \quad (19)$$
$$\delta(\texttt{HasTutor} \sqsubseteq \neg\texttt{Professor}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \texttt{Professor}(\texttt{x}) \quad (20)$$

Considering $db(\mathcal{A})$ and the disjunction of first-order formulas resulting from the translation yields a successful test for satisfiability.

If we want to verify, e.g., $\texttt{Student}(\texttt{Paul})$, then we extend $\mathcal{O}$ with $\hat{\texttt{A}}(\texttt{Paul})$ and $\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}$ resulting in $\mathcal{O}'$, update $db(\mathcal{A}')$ appropriately, and add three more negative inclusions to $cln(\mathcal{T})$ resulting in $cln(\mathcal{T}')$:

$$\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student} \qquad \exists\texttt{TeachesTo}^- \sqsubseteq \neg\hat{\texttt{A}} \qquad \exists\texttt{HasTutor} \sqsubseteq \neg\hat{\texttt{A}}$$

These axioms can again be translated, and it can be verified that the resulting check yields unsatisfiability. From this, we derive that $\texttt{Student}(\texttt{Paul})$ holds.

$$\delta(\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}) = \exists\texttt{x}.\hat{\texttt{A}}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}) \quad (21)$$
$$\delta(\texttt{TeachesTo}^- \sqsubseteq \neg\hat{\texttt{A}}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \hat{\texttt{A}}(\texttt{x}) \quad (22)$$
$$\delta(\texttt{HasTutor} \sqsubseteq \neg\hat{\texttt{A}}) = \exists\texttt{x}.(\exists\texttt{y}\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \hat{\texttt{A}}(\texttt{x}) \quad (23)$$

If we want to incorporate this into the top-down query procedure **SLG($\mathcal{O}$)**, then there are two possible ways for the concrete example. First, we may, e.g., query for $\texttt{Student}(\texttt{Paul})$ and the previously presented steps would derive this from $\mathcal{O}$ alone, so that we would expect the empty answer for **SLG($\mathcal{O}$)**. I.e., nothing needs to be added to $\mathcal{O}$ to derive the queried atom from $\mathcal{O}$ and an unconditional answer is created in the tree for $\texttt{Student}(\texttt{Paul})$.

Alternatively, consider that the ABox is not present, but that, for simplicity, the corresponding statements occur as rule facts as in Example 1. In this case, we want the oracle to return a set of atoms, which if resolved prove the original

query. Clearly, we can derive Student(Paul) if the satisfiability test for $\mathcal{O}$ fails. This is the case if one of the disjuncts in $q_{unsat}^{db(\mathcal{A})}$ is satisfiable, e.g., if there is an x such that Professor(x) $\wedge$ Student(x). Of course, it is counterintuitive to prove that Paul is a student by showing that there is some other individual that is a professor and a student, i.e., by deriving some inconsistency in the interaction of $\mathcal{O}$ and the rules. Thus, all the disjuncts resulting from (16)–(20), do not yield meaningful derivations. Instead they yield derivations based on some general MKNF-inconsistency, which is not possible in a partial oracle (cf. Definition 8).

However, if we resolve the disjuncts resulting from (21)–(23) with $\hat{\texttt{A}}$(Paul), then we obtain more meaningful answers that can be used in the derivation tree for Student(Paul). Namely, Student(Paul) itself is obtained, which is immediately discarded in **SLG($\mathcal{O}$)** since Student(Paul) is already a child in this tree, and ($\exists$yTeachesTo(y, Paul)) and ($\exists$yHasTutor(Paul, y)) are also obtained as possible children. Both do not contribute to the derivation of Student(Paul) itself, which is in fact obtained from the rule fact, but if we query for Student(Jane) or Student(Bill), then in each case one of the two goals unifies with a fact in the given rules.

The insights gained with this example can be formalized in the algorithm (Fig. 2) that provides an oracle for *DL-Lite*$_{\mathcal{R}}$. We only have to formalize the resolution step of the newly introduced query atom with each of the results of applications of δ. The result of such a resolution step is either a ground (unary or binary) atom or a binary atom with one existentially quantified variable. To check whether adding this atom to $\mathcal{O}$ and the already derived information remains consistent, we additionally introduce a uniform notion that turns the new atom into DL notation.

Definition 13. *Let $\mathcal{O}$ be a DL-Lite$_{\mathcal{R}}$ KB, α an axiom in cln($\mathcal{T}$), and $\delta(\alpha) = \exists \boldsymbol{x}.(C_1 \wedge C_2)$ such that H is unifiable with mgu[5] θ with C_i, for some i, in $\delta(\alpha)$. Then $res(\delta(\alpha), H)$ is defined as $(C_2)\theta$ if $i = 1$, and $(C_1)\theta$ otherwise. The DL representation $res_{DL}(\delta(\alpha), H)$ of $res(\delta(\alpha), H)$ is defined depending on the form of $res(\delta(\alpha), H)$:*

$$res_{DL}(\delta(\alpha), H) = \begin{cases} res(\delta(\alpha), H) & \text{if } res(\delta(\alpha), H) \text{ is a ground atom} \\ (\exists R)(a) & \text{if } res(\delta(\alpha), H) = \exists y.R(a, y) \text{ for ground } a \\ (\exists R^-)(a) & \text{if } res(\delta(\alpha), H) = \exists y.R(y, a) \text{ for ground } a \end{cases}$$

We recall that assertions for complex concepts such as $(\exists R)(a)$ are represented by $A(a)$ and $A \sqsubseteq \exists R$ for a new concept name A. This encoding may also affect atoms appearing in $I^+_{\mathcal{F}_n}$ which is why we directly incorporate $I^+_{\mathcal{F}_n}$ into $\mathcal{O}$ in Fig. 2 to avoid a more complicated notation.

The algorithm itself proceeds as outlined in the example. It checks first whether $\mathcal{O}$ together with the already proven true knowledge yields a satisfiable knowledge base. If not, the algorithm stops and returns the empty set; thus, $\mathcal{O}$ is not

[5] most general unifier

Require: *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, which already contains $I^+_{\mathcal{F}_n}$, and a ground atomic query $q = H(t_i)$
Ensure: a set $\mathcal{L}$ of L_i such that $\mathcal{O} \cup L_i \models H(t_i)$ with $\mathcal{O} \cup L_i$ consistent

```
L = ∅
q_unsat = ⊥
for all α ∈ cln(T) do
    q_unsat = q_unsat ∨ δ(α)
end for
if q_unsat^{db(A)} ≠ ∅ then
    L = ∅
else
    q_inst = q_unsat
    T' = T ∪ {Â ⊑ ¬H}
    A' = A ∪ {Â(t_i)}
    O' = (T', A')
    for all α ∈ cln(T') \ cln(T) do
        q_inst = q_inst ∨ δ(α)
    end for
    if q_inst^{db(A')} ≠ ∅ then
        L = {∅}
    else
        for all α ∈ cln(T') \ cln(T) do
            O'' = O ∪ {res_DL(δ(α), Â(t_i))}
            q_uns = q_unsat
            for all β ∈ cln(T'') \ cln(T) do
                q_uns = q_uns ∨ δ(β)
            end for
            if q_uns^{db(A'')} = ∅ then
                L = L ∪ {res(δ(α), Â(t_i))}
            end if
        end for
    end if
    return L
end if
```

Fig. 2. Algorithm *DL-Lite*$_\mathcal{R}$ Oracle

used for further derivations. Otherwise, it proceeds with an instance check for the query, i.e., by checking for unsatisfiability of the extended knowledge base, and, in the case of success, returns a set containing only the empty answer, hence, an unconditional answer in the respective tree of **SLG**($\mathcal{O}$). If the instance check fails, then, for all newly introduced axioms in $cln(\mathcal{T}')$, it is verified whether $res_{DL}(\delta(\alpha), \hat{A}(t_i))$ does not cause an inconsistency if added to $\mathcal{O}$ and the already derived knowledge. If this is successful then the corresponding atom $res(\delta(\alpha), \hat{A}(t_i))$ is included in the set of returned answers, which if proven true, allow us to derive the considered query.

We show that this algorithm provides a correct partial oracle for *DL-Lite*$_\mathcal{R}$ w.r.t. **SLG**($\mathcal{O}$).

Theorem 2. *The algorithm DL-Lite$_\mathcal{R}$ Oracle is sound and complete, i.e., the returned answers in L correspond to the definition of a partial oracle for DL-Lite$_\mathcal{R}$ and the algorithm allows the computation of all the minimal sets L according to the partial oracle for DL-Lite$_\mathcal{R}$.*

Proof. We consider $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models q$ where $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L$ is consistent (Definition 8) and q is the queried atom.

We show soundness, i.e., we show that the returned answers in L correspond to the definition of a partial oracle for *DL-Lite*$_\mathcal{R}$.

If the algorithm returns $\mathcal{L} = \emptyset$, then $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ is not consistent, which means that the result from the algorithm is sound for this case. Otherwise, if the algorithm returns $\mathcal{L} = \{\emptyset\}$, then the instance check for the query succeeds and in this case (for consistent $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ as checked) $\mathcal{O} \cup I^+_{\mathcal{F}_n} \models q$ holds, which is also a sound answer. Finally, if the algorithm returns $\mathcal{L} = \{L_1, \ldots, L_n\}$, then the direct instance check failed, but $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L_i$ is consistent and $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L_i \models q$ holds because the addition of L to $I^+_{\mathcal{F}_n}$ would exactly enable the instance check to succeed (see Theorem 1).

To show completeness, we have to show that the algorithm enables us to compute all the minimal sets L according to the partial oracle for *DL-Lite*$_\mathcal{R}$. First, if $\mathcal{O} \cup I^+_{\mathcal{F}_n}$ is not consistent, then the partial oracle does not return any answer and this is covered by the returned empty set $\mathcal{L}$ in the algorithm. Then, if $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L \models q$ holds for empty L, then the only minimal answer for the partial oracle is the empty set. The algorithm *DL-Lite*$_\mathcal{R}$ Oracle returns exactly only the empty set. It remains to be shown that the correctness result holds for nonempty L as well. So suppose that L' is a nonempty minimal set such that $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L' \models q$ and $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$ is consistent. First, any minimal set can only consist of one atom due to the restricted syntax of GCI in *DL-Lite*$_\mathcal{R}$. Furthermore, joining $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$ and $q = H(t_i)$ together with $\hat{A} \sqsubseteq \neg H$ and $\hat{A}(t_i)$ yields an inconsistent *DL-Lite*$_\mathcal{R}$ KB $\mathcal{O}_1$, hence a successful instance check for q (Theorem 1). If we remove L', then the KB is consistent but the check for consistency would still compute all axioms such that the boolean disjunctive query q_{uns} w.r.t. $\mathcal{O}_1$ without L' would be unsatisfiable as such, but satisfiable upon addition of L', i.e., indicate a successful instance check. This is exactly what the algorithm *DL-Lite*$_\mathcal{R}$ Oracle computes and, for consistent $\mathcal{O} \cup I^+_{\mathcal{F}_n} \cup L'$, L' is returned as one of the answers. Note that none of the $\alpha \in cln(\mathcal{T})$ is considered since if one of these succeeds, the entire knowledge base is inconsistent. Thus, considering only $\alpha \in cln(\mathcal{T}') \setminus cln(\mathcal{T})$ suffices to find all possible sets L, since *res* is applicable by construction in each such case. □

Building on the results on computational complexity in *DL-Lite*$_\mathcal{R}$ ([5]), we can show that the algorithm ensures that the oracle is polynomial.

Theorem 3. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in DL-Lite$_\mathcal{R}$. An* **SLG($\mathcal{O}$)** *evaluation of a query q in the algorithm DL-Lite$_\mathcal{R}$ Oracle is decidable with combined complexity PTime and with data complexity LogSpace.*

Proof. We know from the proof of Theorem 43 in [5] that the combined complexity for computing the disjunctive formula using δ on α in $cln(\mathcal{T})$ is polyno-

mial, while the evaluation w.r.t. $db(\mathcal{A})$ is in LogSpace. Consequently, instance checking and checking satisfiability for *DL-Lite*$_\mathcal{R}$ is in PTime and LogSpace respectively. The algorithm *DL-Lite*$_\mathcal{R}$ Oracle applies one such satisfiability check for $\mathcal{O} \cup I_{\mathcal{F}_n}$ and conditionally a further one for the instance check. Then, conditionally a set of (polynomially many in combined complexity) $\delta(\alpha)$ is processed (each in the worst case containing a further satisfiability check, which is a slight extension of the first). We conclude that the combined complexity of *DL-Lite*$_\mathcal{R}$ Oracle is in PTime and the data complexity LogSpace. □

Intuitively, this result is achieved because GCI and RI are of a particular restricted form, so that the oracle only returns single atoms, and does not need to compute minimal subsets of arbitrary size in the power set of all atoms.

Consequently, we obtain the computational complexity of answering DL-safe conjunctive queries in hybrid MKNF knowledge bases with a *DL-Lite*$_\mathcal{R}$ DL part.

Theorem 4. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base with $\mathcal{O}$ in DL-Lite$_\mathcal{R}$. Answering a DL-safe conjunctive query q in* $\mathbf{SLG}(\mathcal{O})$ *is decidable with data complexity PTime and LogSpace if $\mathcal{P}$ is empty.*

Proof. This is a direct consequence of Theorem 5.4 in [1] and Theorem 3 and the fact that, for nonempty $\mathcal{P}$, PTime from the rules includes LogSpace. □

Hence, reasoning can still partially be done with relational databases.

Theorem 5. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a consistent hybrid MKNF knowledge base with $\mathcal{O}$ in DL-Lite$_\mathcal{R}$. The answer to a DL-safe conjunctive query q in* $\mathbf{SLG}(\mathcal{O})$ *corresponds to the well-founded MKNF model.*

Proof. The result follows from Theorem 5.3 in [1] and Theorem 2. □

If $\mathcal{K}$ is MKNF-inconsistent, then there is no well-founded MKNF model, but we obtain a paraconsistent approximation. Consider that the KB from Example 1 is part of a larger KB that is MKNF-inconsistent, but the inconsistency is not related to the predicates shown in the example, i.e., no rule or GCI links predicates from the example to those causing the inconsistency. Then, querying for atoms from Example 1 yields the same results as if $\mathcal{K}$ was MKNF-consistent.

6 Conclusions

In [10] we provided a concrete procedure for KB with non-monotonic rules and an ontology in the DL $\mathcal{REL}$, a fragment of the DL underlying OWL 2 EL. This slightly easier to obtain procedure relies, after preprocessing, on translating the DL to rules rather than on defining an oracle in the true sense of [1] as done here for *DL-Lite*$_\mathcal{R}$. We note that the resulting data complexity is identical, but higher than the one for *DL-Lite*$_\mathcal{R}$, simply because it has a lower complexity than $\mathcal{REL}$ in [10]. By translating the DL into rules, one can also easily obtain a procedure for DLP [8] – the DL underlying OWL 2 RL.

With the results in this paper, query-answering procedures that do not jeopardize tractability are now available for MKNF KB with rules and ontologies for the DL underlying all the three OWL2 profiles defined by W3C. As the next step, we want to provide an implementation of our work, building on XSB[6] and QuOnto/Mastro[7]. Moreover, OWL 2 QL has some expressive features not contained in $DL\text{-}Lite_{\mathcal{R}}$ and an extension is considered for future work.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to Hybrid MKNF Knowledge Bases Through Oracular Tabling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 1–16. Springer, Heidelberg (2009)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
3. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Candidate Recommendation (June 22, 2010)
4. Boley, H., Kifer, M. (eds.): RIF Framework for Logic Dialects. W3C Candidate Recommendation (June 22, 2010), `http://www.w3.org/TR/rif-fld/`
5. 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)
6. Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. J. ACM 43(1), 20–74 (1996)
7. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. J. Artif. Intell. Res. 31, 150–197 (2008)
8. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logics. In: Proc. of the World Wide Web Conference (WWW 2003), pp. 48–57. ACM (2003)
9. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (October 2009)
10. Knorr, M., Alferes, J.J.: Querying in $\mathcal{EL}^+$ with nonmonotonic rules. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) 19th European Conf. on Artificial Intelligence, ECAI 2010, pp. 1079–1080. IOS Press (2010)
11. 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)
12. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Mylopoulos, J., Reiter, R. (eds.) 12th Int. Joint Conf. on AI, IJCAI 1991, pp. 381–386 (1991)
13. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. Journal of the ACM 57(5), 93–154 (2010)
14. Patel, C., Cimino, J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching Patient Records to Clinical Trials Using Ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 816–829. Springer, Heidelberg (2007)
15. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Candidate Recommendation (January 2008)

[6] `http://xsb.sourceforge.net/`
[7] `http://www.dis.uniroma1.it/quonto/`