A Proof Theory for DL-Lite*

Diego Calvanese, Evgeny Kharlamov, Werner Nutt

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
{calvanese,kharlamov,nutt}@inf.unibz.it

Abstract. In this work we propose an alternative approach to inference in DL-Lite, based on a reduction to reasoning in an extension of function-free Horn Logic (EHL). We develop a calculus for EHL and prove its soundness and completeness. We also show how to achieve decidability by means of a specific strategy, and how alternative strategies can lead to improved results in specific cases. On the one hand, we propose a strategy that mimics the query-answering technique based on first computing a query rewriting and then evaluating it. On the other hand, we propose strategies that allow one to anticipate the grounding of atoms, and that might lead to better performance in the case where the size of the TBox is not dominated by the size of the data.

1 Introduction

The description logics (DLs) of the *DL-Lite* family [1, 2] have been proposed recently as DLs providing a good compromise between expressive power and computational complexity of inference. Indeed, *DL-Lite* and its variants are able to capture the fundamental features of conceptual modeling formalisms, while still keeping the basic reasoning polynomial in the size of the whole DL knowledge base (KB), and LOGSPACE in the size of the data. Notably, such reasoning services include answering conjunctive queries (CQs) formulated over a KB. Moreover, techniques have been developed to perform query answering by leveraging database technology: the ABox is actually stored in a relational database (DB), and (after suitable pre-processing) the query is answered by exploiting the relational DB engine. This approach ensures scalability of query answering over DL KBs to billions of data items. More precisely, the approach for query answering proposed in [1] is actually divided in three phases: *(1) Consistency* of the knowledge base w.r.t. functionality and (pre-processed) disjointness assertions in the TBox is verified by posing appropriate queries to the DB (i.e., the ABox) only (and independently on the actual query); *(2)* The user query is *rewritten* into a new query using the inclusion assertions in the TBox; *(3)* The rewritten query is shipped to the DB, and the returned tuples constitute the answer returned to the user.

In this work, we still rely on Phase (1), but take a closer look at Phases (2) and (3), and at the underlying formal properties of *DL-Lite*. Specifically, we exploit the similarity of TBox inclusion assertions and of ABox membership assertions to clauses in an extension of function-free Horn Logic (which we call *EHL*), in which existentially

* Research supported by the EU FET project TONES (Thinking ONtologiES, contract FP6-7603), and by the PRIN 2006 project NGS (New Generation Search), funded by MIUR.

quantified variables may appear in the clauses. We develop a sound and complete calculus for EHL that bears similarity to resolution [3], but is equipped with a specific rule to handle existentially quantified variables. In three cases, we show how to obtain complete algorithms for query answering by imposing control strategies on the calculus. The general algorithm ensures termination by using loop detection, exploiting the fact that the number of non-isomorphic clauses that can be generated from a query goal using the knowledge base is bounded. A second control strategy mimics the perfect reformulation algorithm in [4], by strictly separating the derivation steps that correspond to operations in Phases (2) and (3). Finally, the third algorithm prunes the search space in a way that is analogous to SLD-resolution in Logic Programming. Moreover, the third algorithm prunes the space by detecting failure derivations in advance.

We obtained the results for *DL-Lite*$_{\mathcal{F}}$ only. However, similar results can be obtained for other DLs in the *DL-Lite* family, such as *DL-Lite*$_{\mathcal{R}}$ [2].

2 *DL-Lite*$_{\mathcal{F}}$

Syntax, Semantics of* DL-Lite$_{\mathcal{F}}$ *and Queries. Let $AC = \{A_1, \ldots, A_{|AC|}\}$ be a set of *atomic concepts*, $AR = \{R_1, \ldots, R_{|AR|}\}$ a set of *atomic roles*, and $Const$ a countable set of *constants*. We inductively define *DL-Lite*$_{\mathcal{F}}$ concepts in the following way: *basic concept* $B \longrightarrow A \mid \exists R \mid \exists R^-$, (general) *concept* $C \longrightarrow B \mid \neg B \mid C_1 \sqcap C_2$, where $A \in AC$, $R \in AR$. With R^- we denote the *inverse* of the role R. In the following, A denotes an atomic concept, B a basic concept, C a concept, and R an atomic role.

A *DL-Lite*$_{\mathcal{F}}$ *knowledge base* (KB) $\mathcal{K}$ is constituted by a TBox (denoted as $\mathcal{T}$) and an ABox (denoted as $\mathcal{A}$). Each *DL-Lite* TBox consists of *inclusion assertions* of the form $B \sqsubseteq C$ and *functionality assertions* of the form $(\mathsf{funct}\ R)$ or $(\mathsf{funct}\ R^-)$. An ABox consists of *membership assertions* of the form $A(a)$, $R(a, b)$, where and a, b are constants. Note that negation can occur only on the right side of inclusion assertions, and that an inclusion assertion $B \sqsubseteq C_1 \sqcap C_2$ can be rewritten as a pair of inclusion assertions $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$. Therefore, in the following, we will assume w.l.o.g., that conjunction does not occur in the TBox, and we denote with $Pos(\mathcal{K})$ the set of all inclusion assertions in $\mathcal{K}$ without negation on the right hand side.

The semantics of *DL-Lite*$_{\mathcal{F}}$ is defined in the usual way for DLs, by resorting to interpretations $I = (\Delta, \cdot^I)$ over a fixed infinite countable *domain* Δ. We just remark that $(\mathsf{funct}\ R)$ is interpreted as functionality of role R. We assume that there is a bijection between $Const$ and Δ (i.e., we have *standard names*). Hence, we do not distinguish between the alphabet of constants $Const$ and the domain Δ. We define models for assertions and KBs in the usual way and say that a KB is *satisfiable* if it has a model.

We use the following rule based notation for defining *conjunctive queries* (CQs):

$$q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y}),$$

where $\exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$ (also denoted as $body(q)$) is a formula of the form $\exists \boldsymbol{y}\, R_1(t_1, t'_1) \wedge \cdots \wedge R_n(t_n, t'_n) \wedge A_1(t''_1) \wedge \cdots \wedge A_k(t''_k)$, where all R_i are binary and A_i unary predicate symbols, all t_i are either variables or constants and each variable that occurs in the conjunction is from $\boldsymbol{x}$ or $\boldsymbol{y}$. The vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the *distinguished* and *non-distinguished* variables of q, respectively. If $\boldsymbol{x}$ is empty, we call the query *boolean*. We denote as `True` the boolean query with no atoms in its body.

We denote the set of all constants that occur in $\mathcal{K}$ as $adom(\mathcal{K})$. We say that q is a query *over a KB* $\mathcal{K}$ if all the predicate symbols occurring in $body(q)$ also occur in $\mathcal{K}$ and the constants are from $adom(\mathcal{K})$. For a given KB $\mathcal{K}$, model I of $\mathcal{K}$ and query $q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$ over $\mathcal{K}$, the set $q(I)$ of *answers* of q over I is defined as: $q(I) = \{\gamma\boldsymbol{x} \mid I \models \exists \boldsymbol{y}\, body(\gamma\boldsymbol{x}; \gamma\boldsymbol{y}),\ \gamma : Var \rightarrow \Delta\}$.

The definition above says how to answer a query over a given model of a KB. Now we define how to answer a query over a KB itself. For this purpose we use the so-called *certain answers semantics*, i.e., for a given query q and a KB $\mathcal{K}$, the set $q(\mathcal{K})$ of *certain answers* (or the *answer set*) of q over $\mathcal{K}$ is defined as

$$q(\mathcal{K}) = \{\boldsymbol{c} \mid \boldsymbol{c} \in adom(\mathcal{K})^{|\boldsymbol{c}|} \text{ and } \boldsymbol{c} \in q(I), \text{ for every model } I \text{ of } \mathcal{K}\}.$$

Reasoning. Here we define query answering and discuss ways to perform it.

Conjunctive query (CQ) answering for a CQ q and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the task of finding $q(\mathcal{K})$. One can easily see that CQ answering is equivalent to finding all tuples $\boldsymbol{c}$ of constants from $adom(\mathcal{K})$ such that the entailment $\mathcal{K} \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ holds. It turns out (see the separation theorem in [4]) that $\mathcal{K} \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ holds if and only if $\mathcal{K}$ is satisfiable and $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ holds. In order to check satisfiability of $\mathcal{K}$ it is enough to verify that the minimal model of $\mathcal{A}$ (the intersection of all models of $\mathcal{A}$) satisfies all functional assertions of $\mathcal{K}$ and all negative inclusion assertions entailed by $\mathcal{K}$ [4].

In order to decide $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$, one needs to check that all models of the premises satisfy the conclusion. It turns out there may be infinitely many (possibly infinite) "different" models of $\mathcal{K}$ [5]. Hence, at a first glance, it is not clear at all whether query answering is decidable.

It turns out [5] that any satisfiable *DL-Lite*$_{\mathcal{F}}$ KB meets the so-called *universal model property*. That is, there exists a model UI (called a *universal model*) of $\mathcal{K}$ that can be homomorphically embedded in any another model of $\mathcal{K}$. Due to this fact, the entailment checking for $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ is equivalent to model checking for $UI \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$. For instance, a *chase* [6] of the minimal model of $\mathcal{A}$ with $Pos(\mathcal{K})$ "produces" a universal model of $\mathcal{K}$ (denoted as $chase(\mathcal{K})$). The constructive nature of $chase(\mathcal{K})$ allows one to model check $chase(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ in finite time, even if $chase(\mathcal{K})$ is infinite [1]. In [1] an algorithm is presented (called *perfect reformulation*) to decide whether $chase(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$ holds.

Perfect reformulation allows one even more, while deciding whether $chase(\mathcal{K}) \models \exists \boldsymbol{x}.\boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$ holds, it returns all the vectors $\boldsymbol{c}$ from the answer set $q(\mathcal{K})$. The algorithm works in two stages as follows: *(1)* it rewrites a CQ $\exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$ to a set S of CQs using assertions from $Pos(\mathcal{K})$ and *(2)* it evaluates S over $\mathcal{A}$ stored as an RDB. The evaluation returns precisely $q(\mathcal{K})$.

In this work we propose an orthogonal (proof theoretical) approach for deciding the entailment $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$, based on a deductive system (calculus). Moreover, while verifying the entailment $\mathcal{K} \models \exists \boldsymbol{x}.\boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$, the calculus allows one to construct deductions that return precisely $q(\mathcal{K})$. We will show later that the perfect reformulation algorithm can be obtained from the calculus by putting a specific control strategy over it.

3 *DL-Lite*$_{\mathcal{F}}$ vs Extended Horn Logic

The idea to make a deductive system for CQ answering has arisen from the observation that *DL-Lite*$_{\mathcal{F}}$ in fact is a syntactic variation of a fragment of slightly extended Horn Clause Logic, such that existentially quantified variables are allowed to occur in positive literals of Horn clauses (heads of horn rules). Using this observation we adopted the resolution calculus in order to deal with the extension to Horn Logic. In this section we present the extension of Horn Logic and the calculus.

Extended Horn Logic. In *Extended Horn Logic* (EHL) formulas (called *e-clauses*) are of the form

$$\forall \boldsymbol{x}\, \exists \boldsymbol{y}\, (L_1(\boldsymbol{x}, \boldsymbol{y}) \vee \cdots \vee L_m(\boldsymbol{x}, \boldsymbol{y})),$$

where each L_i is a literal over $AC \cup AR$, the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ contain all variables occurring in $L_1, \ldots, L_m$, and at most one literal is positive. As usual we use the terms *goal* and *fact* to refer to e-clauses with no positive literal and no negative literal, respectively.

In [7], natural translations π_y and π respectively from $\mathcal{SHIQ}$ concepts and assertions to FOL were presented. In fact π maps positive inclusion and membership assertions of *DL-Lite* to EHL. We extend π to conjunctive queries and present both π_y and π in the following table. The variable y in $\pi_y(B, x)$ indicates that an implicit existential variable in the the DL expression B will be explicitly denoted as y in the FOL version.

$\pi_y(A, x) = A(x)$	$\pi_y(\exists R, x) = \exists y R(x, y)$
	$\pi_y(\exists R^-, x) = \exists y R(y, x)$
$\pi(A(a)) = A(a)$	$\pi(B \sqsubseteq B') = \pi_y(B', x) \vee \neg \pi_z(B, x)$
$\pi(R(a, b)) = R(a, b)$	$\pi(\exists \boldsymbol{y}\ body(\boldsymbol{x}; \boldsymbol{y})) = \neg body(\boldsymbol{x}; \boldsymbol{y})$

where $A \in AC$; $R \in AR$; a, b are constants; B, B' are basic concepts;

We call *CQ goal* a goal corresponding to a CQ. We say that $\boldsymbol{x}$ is the *vector of distinguished variables of a CQ goal* Γ if $\Gamma = \pi(\exists \boldsymbol{y}\ body(\boldsymbol{x}; \boldsymbol{y}))$. Note, that CQ goals do not contain existential variables and e-clauses that correspond to membership assertions are facts.

The following are examples of applying π_y and π: $\pi(A \sqsubseteq \exists R) = \pi_y(\exists R, x) \vee \neg \pi_z(A, x) = \exists y R(x, y) \vee \neg A(x)$, and $\pi(\exists R^- \sqsubseteq \exists R') = \pi_y(\exists R', x) \vee \neg \pi_z(\exists R^-, x) = \exists y R'(x, y) \vee \neg \exists z R(z, x)$, and $\pi(\texttt{True}) = \bot$.

Calculus for Extended Horn Logic. Our calculus consists of three rules. Rules should be read from top to bottom. We assume that the order of literals in the goals is irrelevant.

1. **Factorization Rule:**

$$fr: \ \frac{\Gamma \vee L_1 \vee L_2}{\Gamma\theta \vee L_1\theta}\ [\theta]$$

   where the literals L_1 and L_2 are unifiable, and θ is an mgu that is the identity on distinguished variables. If $L_1 = L_2$, then $\theta = id$.
2. $\exists$**-resolution rule:**

$$erl: \ \frac{\Gamma \vee \neg \pi_x(B, t) \qquad \pi_v(B, y) \vee \neg \pi_z(B', y)}{\Gamma \vee \neg \pi_w(B', t)}\ [id]$$

where x is a non-distinguished variable that occurs only once in $\Gamma \vee \neg\pi_x(B,t)$ and w is a fresh variable.

3. **Resolution rule:**

$$rl: \ \frac{\Gamma \vee L \qquad D}{\Gamma\theta}\ [\theta]$$

where D is a ground atom of the from $A(a)$ or $R(a,b)$, i.e., a membership assertion, and θ is an mgu of the literals L and $\neg D$.

We say that in the derivation rules *fr*, *erl* and *rl*, the literals L_2, $\neg\pi_x(B,t)$ and L are the *leading* ones, respectively. Since each time a derivation step is performed the leading literal is either eliminated or substituted with another literal, we say the leading literal is *processed* by the derivation rule with the e-clause on the right in the premisses of the rule (which is assumed to be $\bot$ in the case of the *fr* rule).

Note, that each time a derivation step is performed the leading literal is either eliminated or substituted with another literal. Because of this reason we say that the leading literal is *processed* by the derivation rule with the e-clause that occurs on the right in the premisses of the rule (in the case of the fr rule the e-clause is assumed to be $\bot$).

As usual, we say that a goal Γ' is *directly derived* from a goal Γ and an e-clause g by a rule r with a substitution θ, denoted as $\Gamma \vdash_{g,\theta} \Gamma'$, if r is either an *fr*, or an *erl*, or an *rl* rule, Γ and g are respectively on the left and on the right in the premisses of r, Γ' is in the conclusion of r, and θ *participates* in r (it occurs in r). If a sequence $\Gamma_1 \cdots \Gamma_n$ of goals is such that for each $i < n$ the direct derivation $\Gamma_i \vdash_{g_i,\theta_i} \Gamma_{i+1}$ holds, then we say that Γ_n is *derived* from Γ_1 with the substitution $\Theta = \theta_1 \circ \cdots \circ \theta_{n-1}$ and the set of e-clauses $L = \{g_1, \ldots, g_{n-1}\}$, and denote it as $\Gamma_1 \vdash^*_{L,\Theta} \Gamma_n$.

We say that a query q' is *derived from* a query q and a KB $\mathcal{K}$ with a substitution θ, denoted as $q \vdash^*_{\mathcal{K},\theta} q'$, if there is a derivation of a CQ goal $\pi(q')$ from $\pi(q)$ with θ and a set of e-clauses L, where each $l \in L$ corresponds either to an assertion from $Pos(\mathcal{K})$ or to a membership assertion of $\mathcal{K}$.

4 CQ Answering as Deduction in EHL

We motivated the e-calculus as a general instrument to answer conjunctive queries over satisfiable *DL-Lite*$_\mathcal{F}$ knowledge bases. In this section we state several formal properties of the general calculus and of some control strategies for it. The proofs are contained in a forthcoming technical report [8].

As a first result, we can show that the calculus can be used to verify that a boolean CQ is entailed by a satisfiable knowledge base.

Theorem 1 (Soundness and Completeness). *Let $\mathcal{K}$ be a satisfiable KB, and $q() \leftarrow \exists \boldsymbol{y}\, body(\boldsymbol{y})$ a boolean CQ over $\mathcal{K}$. Then q is entailed by $\mathcal{K}$, i.e., $\mathcal{K} \models \exists \boldsymbol{y}\, body(\boldsymbol{y})$, if and only if there exists a derivation from $\pi(body(q))$ to the boolean query* `True`.

As a consequence of this theorem, we can use the calculus to verify that a tuple of constants $\boldsymbol{c}$ is a certain answer of a CQ $q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$, since $\boldsymbol{c} \in q(\mathcal{K})$ if and only if $\mathcal{K} \models \exists \boldsymbol{y}\, body(\boldsymbol{c}; \boldsymbol{y})$. The next theorem shows that the calculus can be used to generate all certain answers.

Theorem 2 (Answer Completeness). *Let $\mathcal{K}$ be a satisfiable KB, $q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}\, body(\boldsymbol{x}; \boldsymbol{y})$ a CQ over $\mathcal{K}$, and $\mathbf{c}$ a vector of constants. Then $\mathbf{c}$ is in the answer set of q over $\mathcal{K}$, that is $\mathbf{c} \in q(\mathcal{K})$, if and only if there is a derivation from $\pi(body(q))$ to the boolean query* `True` *with the substitution θ such that $\boldsymbol{x}\theta = \mathbf{c}$.*

General Algorithm. We are now in a position to formulate a non-deterministic algorithm to compute the answer set $q(\mathcal{K})$. For the algorithm, we specify the states of the computation, transitions between such states, and the subset of final states.

Let $\boldsymbol{x}$ be the vector of distinguished variables of q. A *granule* is a pair $\Gamma.\sigma$, where Γ is a CQ goal and σ is a substitution that maps the variables in $\boldsymbol{x}$ to constants or to themselves. The states of the computation are sets of granules, denoted by the letter $\mathcal{G}$. When computing $q(\mathcal{K})$, the initial state is the set $\{\pi(body(q)).id\}$.

Suppose a goal Γ is derived from $\pi(body(q))$ by our calculus. Then, in addition to $\boldsymbol{x}$, the goal Γ may contain non-distinguished variables and new variables that are introduced by the erl-rule. We refer to both these new and non-distinguished variables as the *existential* variables of Γ. We say that two granules $\Gamma.\sigma$ and $\Gamma'.\sigma'$ are *similar* if $\sigma = \sigma'$ and if Γ and Γ' are identical up to renaming of their existential variables.

In order to define the transition between states, we first need to slightly extend our calculus so that it operates on granules. A granule $\Gamma'.\sigma'$ is derived from $\Gamma.\sigma$ if Γ' is derived from Γ with θ and $\sigma' = \sigma \circ \theta$. There is a transition from a state $\mathcal{G}$ to a state $\mathcal{G}' = \mathcal{G} \cup \{G'\}$ if G' is not similar to any granule in $\mathcal{G}$ and there is a $G \in \mathcal{G}$ such that G' can be derived from G. In this case we say that the transition *processes* G. A state $\mathcal{G}$ is final if no transition from $\mathcal{G}$ is possible.

We note that there are only finitely many different atomic concepts, atomic roles, and constants occurring in the KB. Hence, for a given maximal length of goals and set of distinguished variables, it is only possible to form finitely many non-similar granules. We also note that the rules of our calculus never increase the length of a goal. Hence, for a given granule, we can only derive finitely many non-similar granules.

The following theorem states that all certain answers of a CQ over a satisfiable KB can be obtained by computing a final state and collecting substitutions from granules with empty goals.

Theorem 3. *Let q be a CQ with distinguished variables $\boldsymbol{x}$ over a satisfiable KB $\mathcal{K}$. Then every sequence of transitions starting from $s = \{\pi(body(q)).id\}$ and using $\mathcal{K}$ terminates. Moreover, if $\mathcal{G}$ is a final state reached from s, then $q(\mathcal{K}) = \{\sigma\boldsymbol{x} \mid (\bot.\sigma) \in \mathcal{G}\}$.*

Perfect Reformulation Algorithm. In our framework, we can also show the soundness and completeness of the perfect reformulation algorithm [4]. In fact, an execution of this algorithm corresponds to a sequence of transitions where initially only such transitions are performed that employ the factorization and the $\exists$-resolution rules of our calculus. When no transitions of this kind are possible any more, then transitions corresponding to resolution steps are performed. It follows from Theorem 1 that such a strategy leads in fact to certain answers. To show completeness, we need an extra argument. This is provided by the following proposition, which shows that resolution steps can always be postponed until the end of a derivation.

Proposition 1 (Resolution Commutes). *Suppose that from $\Gamma \vee L$ we can obtain Γ' by first applying the resolution rule to the leading literal L with substitution θ and then another rule r to some leading literal $M\theta$. Then we can obtain Γ' as well by first applying r to $\Gamma \vee L$ with the leading literal M and substitution δ, and then to the result the resolution rule with leading literal $L\delta$. That is,*

$$\Gamma \vee L \vdash_{g,\theta} \Gamma\theta \vdash_{g',\theta'} \Gamma' \text{ implies } \Gamma \vee L \vdash_{g',\delta} (\Gamma''\delta \vee L\delta) \vdash_{g,\delta'} \Gamma' \text{ and } \theta \circ \theta' = \delta \circ \delta'.$$

Live-Only Algorithm. In our calculus, it is possible to construct several different derivations from a goal Γ to another goal Γ' and the transition-based algorithm will in fact compute all such derivations. Another problem is that the algorithm is unable to detect granules that will not lead to any answer and continues to process them. To avoid such unnecessary computations, we introduce a criterion to recognize when a granule needs no further processing.

We say a variable x is *critical* for a CQ goal Γ, if x occurs more than once in Γ. A literal L in Γ is *terminal* if L is unary or if it is binary and does not have a critical variable. Intuitively, if a literal L is terminal in Γ and no rules are applicable to L, then no rule will ever become applicable to L in any Γ' derived from Γ.

We say that a granule $\Gamma.\sigma$ is *exhausted* in $\mathcal{G}$, if Γ contains a terminal literal L such that the following holds: if Γ' is obtained from Γ by applying a rule of the calculus with leading literal L and substitution θ, then $(\Gamma'.\sigma \circ \theta)$ is similar to some granule in $\mathcal{G}$. Intuitively, a granule is exhausted if it contains one terminal literal that has been completely processed. A granule is *live* if it is not exhausted.

The *Live-Only Algorithm* is a variant of the general algorithm. It is different in that transitions cannot process arbitrary granules, but only live granules. The Live-Only Algorithm allows for a specific control strategy, which resembles SLD-Resolution in Logic Programming. Under SLD-Resolution, an arbitrary literal in a goal is chosen and resolved in all possible ways. After that, the goal is discarded. In our case, if we choose a terminal literal and process it in all possible ways, then the granule becomes exhausted and is blocked from any further rule application. We do not discard exhausted granules from a state because their presence is needed to detect loops.

The completeness of the Live-Only Algorithm (and therefore also of the SLD-like strategy) can be shown by an induction argument using the proposition below, which states that in a derivation a rule application to a terminal literal commutes with all preceding derivation steps.

Proposition 2. *Let L be a terminal literal of $\Gamma \vee L$. Suppose there is a derivation of Γ' from $\Gamma \vee L$ where an instantiation of L is processed at the last step. Then there is another derivation of Γ' from $\Gamma \vee L$ where L is processed at the first step. That is,*

$$\Gamma \vee L \vdash^*_{G,\theta_1} (\Gamma_1 \vee L\theta_1) \vdash_{g,\theta_2} \Gamma' \text{ implies } \Gamma \vee L \vdash_{g,\delta_1} \Gamma\delta_1 \vdash^*_{G,\delta_2} \Gamma' \text{ and } \theta_1 \circ \theta_2 = \delta_1 \circ \delta_2$$

5 Related Works and Conclusions

Using resolution for query answering over DL KBs was already considered by several authors. In [7] Hustadt at al. adopted resolution for CQ answering over $\mathcal{SHIQ}$ KBs

$\mathcal{K}_{\mathcal{SHIQ}}$. They presented a way to decide whether $\boldsymbol{c} \in q(\mathcal{K}_{\mathcal{SHIQ}})$ holds, but they do not propose any procedure for computing answer sets $q(\mathcal{K}_{\mathcal{SHIQ}})$. Since, our procedure for computing answer sets involves only positive inclusion and membership assertions, i.e., a fragment of $\mathcal{SHIQ}$, the results of our paper extend the ones in [7] for this fragment. Another work on using resolution for query answering is [9], where a way is proposed to check whether an $\mathcal{EL}$ KB implies a subsumption between two concepts by posing atomic queries to the KB. This work does not consider CQs and computing certain answers. To the best of our knowledge, our work is the first one that considers computing answer sets for CQs over DL KBs in the framework of resolution.

We envisage that our work will facilitate the combination of *DL-Lite* with other formalisms in data management tasks that are based on variants of Horn logic, such as mappings in data integration [10] and dependencies in data exchange [11]. It remains also to be investigated under which conditions the adoption of evaluation strategies for the calculus that are different from the one underlying the rewriting approach, may lead to improved performance. Specifically, such alternative strategies look promising for those cases where the size of the TBox is *not* negligible w.r.t. the size of the ABox, and the ABox may not be directly managed by a DBMS. In such cases, an approach based on rewriting would generate very large queries to be shipped to the database, while anticipating resolution with ground atoms may result in strong pruning.

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proc. of AAAI 2005. (2005) 602–607
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR 2006. (2006) 260–270
3. Lloyd, J.W.: Foundations of Logic Programming (Second, Extended Edition). Springer, Berlin, Heidelberg (1987)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning (2007) To appear.
5. Kharlamov, E.: Model theory and calculus for the description logic DL-Lite. Master's thesis, Faculty of Computer Science, Free University of Bozen-Bolzano (2006) Available at http://www.inf.unibz.it/ kharlamov/.
6. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
7. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: Proc. of LPAR 2004. (2004) 21–35
8. Calvanese, D., Kharlamov, E., Nutt, W.: A proof theory for DL-Lite. Technical Report KRDB07-6, KRDB Research Center, Faculty of Computer Science, Free University of Bozen-Bolzano (2007)
9. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany (2006) Available at http://www.cs.man.ac.uk/ ykazakov/publications/thesis/Kazakov06Phd.pdf.
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Linking data to ontologies: The description logic DL-Lite$_A$. In: Proc. of OWLED 2006. (2006)
11. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proc. of PODS 2005. (2005) 61–75