Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

Query Rewriting in *DL-Lite*$_{horn}^{(\mathcal{HN})}$ *

Elena Botoeva, Alessandro Artale, and Diego Calvanese

KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
lastname@inf.unibz.it

Abstract. In this paper we present practical algorithms for query answering and knowledge base satisfiability checking in *DL-Lite*$_{horn}^{(\mathcal{HN})}$, a logic from the extended *DL-Lite* family that contains horn concept inclusions and number restriction. This logic is the most expressive DL that is shown to be FOL-rewritable. The algorithms we present are based on the rewriting technique so that reasoning over the TBox and over the ABox can be done independently of each other, and the inference problems are reduced to first order query evaluation. This allows for employing relational database technology for the final query evaluation and gives optimal data complexity.

1 Introduction

Query answering is the main reasoning task in the setting of ontology based data access [1,2] and data integration [3], where large amounts of data are stored in external databases, and accessed through a conceptual layers provided by an ontology (expressed in terms of a description logic knowledge base). Query answering in this case requires reasoning, and to perform it efficiently in practice the underlying description logic has to be 'lite' enough, to be more precise it should enjoy first-order rewritability: it should be possible to rewrite a query q posed over the ontology in terms of a new query that can be directly evaluated over the data, and that provides the same answers as those provided by q.

The *DL-Lite* family [4] is a family of description logics that enjoy such nice computational properties, i.e., data complexity of query answering is in AC^0. For *DL-Lite*$_{core}$ and *DL-Lite*$_{core}^{\mathcal{F}}$ [1], the basic logics of the *DL-Lite* family, a polynomial query answering algorithm was established in [4], and later extended to *DL-Lite*$_{\mathcal{A}}$ [1]. The idea of the algorithm is to first rewrite the query by taking into account the assertions in the TBox and then to evaluate the rewritten query over the ABox. Based on this approach, several systems were implemented, notably QuOnto [6,7].

DL-Lite$_{horn}^{(\mathcal{HN})}$ is a more expressive logic than *DL-Lite*$_{\mathcal{A}}$, which contains number restrictions (as opposed to global functionality assertions), allows for conjunction of basic concepts on the left-hand side of concept inclusions, and for role inclusions that

* This work has been partially supported by the EU project Ontorule (ICT-231875).

[1] Notice that we adopt here the naming convention for logics introduced in [5].

however cannot interact with maximum number restriction. As shown in [5], also *DL-Lite*$_{horn}^{(\mathcal{HN})}$ is FOL-rewritable, i.e., a similar approach to the one discussed above can be used for query answering over ontologies. However, the algorithm presented in [5] is not immediately implementable, since it would generate queries that would be extremely difficult to process and optimize by a DBMS. Indeed as demonstrated by recent experiments with ontology based data access systems [2], commercial relational DBMSs are not designed and optimized to process complex queries (where, e.g., joins are performed over unions), and for such kinds of queries performance degrades dramatically when the size of the data increases.

Therefore, in this paper we address the problem of devising an algorithm for answering unions of conjunctive queries in *DL-Lite*$_{horn}^{(\mathcal{HN})}$ that is based on rewriting, and where the rewriting step generates again a union of conjunctive queries. Such an algorithm can be directly implemented in a system like QUONTO by extending the current algorithm for *DL-Lite*$_{\mathcal{A}}$. Moreover, since current DBMSs are optimized for the evaluation of conjunctive queries, they can process the queries generated by our rewriting algorithm more efficiently than queries generated by an algorithm that tries to delegate complex operations to the DBMS.

Summing up, our contributions are the following:

- We present an algorithm for answering unions of conjunctive queries posed to *DL-Lite*$_{horn}^{(\mathcal{HN})}$ knowledge bases that is in AC^0 for data complexity. We employ the query rewriting approach, that is, query answering is performed in two steps. First, the initial query is rewritten using the TBox. Then, the rewritten query is evaluated over the ABox. The main advantage of this approach is that the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine.
- We provide an algorithm that checks satisfiability of *DL-Lite*$_{horn}^{(\mathcal{HN})}$ knowledge bases by evaluating a first-order query over the ABox and that is in AC^0 for data complexity. Thus, the knowledge base satisfiability problem is reduced to query evaluation and again the TBox and the ABox are processed independently of each other.

2 The Description Logic *DL-Lite*$_{horn}^{(\mathcal{HN})}$

In this section we present the logic *DL-Lite*$_{horn}^{(\mathcal{HN})}$ and give other preliminary definitions.

The language of *DL-Lite*$_{horn}^{(\mathcal{HN})}$ contains atomic concept and role names, respectively denoted by A and P, possibly with subscripts. *Basic roles* and *concepts*, denoted respectively by R and B, possibly with subscripts, are defined as $R ::= P \mid P^-$ and $B ::= \bot \mid A \mid \geq k\, R$, where k is a positive integer. $\geq k\, R$ is called a *number restriction*.

A *DL-Lite*$_{horn}^{(\mathcal{HN})}$ *TBox*, $\mathcal{T}$, is a finite set of *concept* and *role inclusion axioms* of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ and $R_1 \sqsubseteq R_2$, respectively, and *role constraints* $\mathsf{Dis}(R_1, R_2)$, $\mathsf{Asym}(P)$, $\mathsf{Sym}(P)$, $\mathsf{Irr}(P)$ and $\mathsf{Ref}(P)$. The TBox $\mathcal{T}$ may also contain occurrences of qualified number restrictions $\geq k\, R.B$ on the right-hand side of concept inclusions. The TBox assertions must satisfy the following conditions:

(inter) if R has a proper sub-role in $\mathcal{T}$ (i.e., $R' \sqsubseteq R$, $R \not\sqsubseteq R'$ for some R'), then $\mathcal{T}$ does not contain occurrences of number restrictions $\geq k\, R$ or $\geq k\, R^-$ with $k \geq 2$ on the left-hand side of concept inclusions.

(exists) if $\geq k\, R.B$ occurs in $\mathcal{T}$, then $\mathcal{T}$ does not contain occurrences of $\geq k'\, R$ or $\geq k'\, R^-$, for $k' \geq 2$, on the left-hand side of concept inclusions.

Note that disjointness between concepts B_1 and B_2 is expressed as $B_1 \sqcap B_2 \sqsubseteq \bot$. Also, *DL-Lite*$_{horn}^{(\mathcal{HN})}$ allows for expressing local cardinality constraints, e.g., the assertion $A \sqcap \geq 3\, R \sqsubseteq \bot$ is equivalent to $A \sqsubseteq \leq 2\, R$.

An *ABox* $\mathcal{A}$ is a finite set of *membership assertions* of the form $A(a)$, $\neg A(a)$, $P(a, b)$, and $\neg P(a, b)$. $\mathcal{T}$ and $\mathcal{A}$ constitute the *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

In the following, we use R^- to denote P^- if $R = P$ and P if $R = P^-$, $R(x, y)$ to denote $P(x, y)$ if $R = P$, and $P(y, x)$ if $R = P^-$. For a TBox $\mathcal{T}$, let $\sqsubseteq^{\pm}$ denote the closure under inverses of the subrole relation: $R \sqsubseteq^{\pm} R' \in \mathcal{T}$ iff $R \sqsubseteq R' \in \mathcal{T}$ or $R^- \sqsubseteq R'^- \in \mathcal{T}$.

The formal semantics relies on the standard notion of interpretation [8]. Here we adopt the *unique name assumption* (UNA). In the following we assume that TBoxes do not contain role constraints $\mathsf{Asym}(P)$, $\mathsf{Sym}(P)$, $\mathsf{Irr}(P)$, $\mathsf{Ref}(P)$ and qualified number restrictions: we can get rid of them as described in [5].

In this work we concentrate on two reasoning tasks for *DL-Lite*$_{horn}^{(\mathcal{HN})}$, to which other reasoning tasks can be reduced: knowledge base satisfiability and query answering. The *KB satisfiability problem* is to check, given a KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model. To define the query answering problem, we first provide some definitions.

A *conjunctive query (CQ)* q *over a KB* $\mathcal{K}$ is a first-order formula of the form: $q(\boldsymbol{x}) = \exists \boldsymbol{y}.conj(\boldsymbol{x}, \boldsymbol{y})$, where $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $A(t)$ and $P(t_1, t_2)$, and t, t_1, t_2 are either constants in $\mathcal{K}$ or variables in $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x}$ are the free variables of q, also called *distinguished* variables. A *union of conjunctive queries (UCQ)* q is a formula of the form $q(\boldsymbol{x}) = \bigvee_{i=1,\ldots,n} \exists \boldsymbol{y}_i.conj_i(\boldsymbol{x}, \boldsymbol{y}_i)$, with $conj_i(\boldsymbol{x}, \boldsymbol{y}_i)$ as before. Given a query q (either a CQ or a UCQ) and an interpretation $\mathcal{I}$, we denote by $q^{\mathcal{I}}$ the set of tuples of elements of $\Delta^{\mathcal{I}}$ obtained by evaluating q in $\mathcal{I}$. The answer to q over a KB $\mathcal{K}$ is the set $ans(q, \mathcal{K})$ of tuples $\boldsymbol{a}$ of constants appearing in $\mathcal{K}$ such that $\boldsymbol{a}^{\mathcal{I}} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{K}$. Each such tuple is called *certain answer*. Observe that, if $\mathcal{K}$ is unsatisfiable, then $ans(q, \mathcal{K})$ is trivially the set of all possible tuples of constants in $\mathcal{K}$ whose arity is the one of the query. We denote such a set by $AllTup(q, \mathcal{K})$. The *query answering problem* is defined as follows: given a KB $\mathcal{K}$ and a query q (either a CQ or a UCQ) over $\mathcal{K}$, compute the set $ans(q, \mathcal{K})$.

3 Knowledge Base Satisfiability

To check KB satisfiability, we exploit the notions of canonical interpretation and closure of negative inclusions. First, we define negative and positive inclusion assertions, and the database interpretation.

We call *negative inclusions (NI)* assertions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot$ and $\mathsf{Dis}(R_1, R_2)$. Other assertions are called *positive inclusions (PI)*. $\mathcal{T}_{\mathcal{N}}$ denotes the set of all negative inclusions in $\mathcal{T}$, and $\mathcal{T}_{\mathcal{P}}$ the set of positive inclusions in $\mathcal{T}$. Obviously,

$\mathcal{T} = \mathcal{T}_{\mathcal{N}} \cup \mathcal{T}_{\mathcal{P}}$. Note that NIs include also those assertions that express functionality of roles, e.g. $\geq 2\, R \sqsubseteq \bot$, and more in general maximum number restrictions.

The *database interpretation* $db(\mathcal{A}) = \langle \Delta^{db(\mathcal{A})}, \cdot^{db(\mathcal{A})} \rangle$ of an ABox $\mathcal{A}$ is 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; $P^{db(\mathcal{A})} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$, for each atomic role P.

The canonical interpretation is an interpretation constructed according to the notion of chase [9]. Following [4] we can construct the chase of a KB starting from the ABox, and applying positive inclusions to sets of membership assertions. In the definition of chase, we concentrate here on the differences with respect to the definition of chase given in [4]. We remark, however, that for the application of the chase rules we assume to have a total (lexicographic) ordering on the assertions and on all the constants (including the newly introduced ones). The *chase* of $\mathcal{K}$ is the set of membership assertions $chase(\mathcal{K}) = \bigcup_{j \in \mathbb{N}} \mathcal{S}_j$, where $\mathcal{S}_0 = \mathcal{A}$, $\mathcal{S}_{j+1} = \mathcal{S}_j \cup S_j^{new}$, and S_j^{new} is the set of new membership assertions obtained from $\mathcal{S}_j$ according to the chase rules cr1, cr2, cr3:

cr1 if $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq A$, $S' = S_{B_1}(a) \cup \cdots \cup S_{B_n}(a)$, and $A(a) \notin \mathcal{S}_j$, then $S_j^{new} = \{A(a)\}$,

cr2 if $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \geq k\, R$ and $S' = S_{B_1}(a) \cup \cdots \cup S_{B_n}(a)$, $k_1 = \sharp\{b \mid R(a,b) \in S\} < k$, then $S_j^{new} = \{R(a, b_{k_1+1}), \ldots, R(a, b_k)\}$, where $b_{k_1+1}, \ldots, b_k$ are $k - k_1$ new constants in Γ_N that follow lexicographically the constants introduced in the previous steps,

cr3 if $I = R_1 \sqsubseteq^{\pm} R_2$, $S' = \{R_1(a,b)\}$, and $R_1(a,b) \notin \mathcal{S}_j$ then $S_j^{new} = \{R_2(a,b)\}$,

where S' is the first (in lexicographic order) set of membership assertions in $\mathcal{S}_j$ s.t. there exists a PI applicable[2] to it, I is the first such PI, and we use $S_B(a)$ to denote $\{A(a)\}$ if $B = A$ and $\{R(a, b_1), \ldots, R(a, b_k)\}$ if $B = \geq k\, R$, with $b_1, \ldots, b_k$ some constants. We denote by $chase_i(\mathcal{K})$ the portion of the chase obtained after i applications of the chase rules.

The canonical interpretation is the interpretation $can(\mathcal{K}) = \langle \Delta^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})} \rangle$ where $\Delta^{can(\mathcal{K})}$ is the set of constants occurring in $chase(\mathcal{K})$, $a^{can(\mathcal{K})} = a$, for each constant a, $A^{can(\mathcal{K})} = \{a \mid A(a) \in chase(\mathcal{K})\}$, for each atomic concept A, and $P^{can(\mathcal{K})} = \{(a_1, a_2) \mid P(a_1, a_2) \in chase(\mathcal{K})\}$, for each atomic role P. The canonical interpretation is constructed in such a way that all the positive inclusions of the TBox are satisfied, so a knowledge base where the TBox contains only positive inclusions is always satisfiable. The following lemma establishes a notable property of $can(\mathcal{K})$.

Lemma 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite*$_{horn}^{(\mathcal{HN})}$ *KB, and let $\mathcal{T}_{\mathcal{P}}$ be the set of positive inclusion assertions in $\mathcal{T}$. Then $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_{\mathcal{P}}, \mathcal{A} \rangle$.*

In order to check satisfiability of *DL-Lite*$_{horn}^{(\mathcal{HN})}$ KBs, negative inclusions must be considered. Thus, if a negative inclusion in the TBox is violated by membership assertions of the ABox, then the knowledge base is inconsistent and, therefore, unsatisfiable. Besides, an interaction of positive and negative inclusions may cause inconsistency. So we need to consider all NIs implied by the TBox.

[2] The notion of *applicability* of a PI suitably extends the one in [4].

Definition 2. *Let $\mathcal{T}$ be a TBox. We call* NI-closure of $\mathcal{T}$*, denoted by $cln(\mathcal{T})$, the following set of assertions defined inductively:*

1. $\mathcal{T}_{\mathcal{N}} \subseteq cln(\mathcal{T})$.
2. *if $B_1 \sqcap \cdots \sqcap B_n \sqcap A \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ and $B'_1 \sqcap \cdots \sqcap B'_m \sqsubseteq A$ is in $\mathcal{T}$, then also $B_1 \sqcap \cdots \sqcap B_n \sqcap B'_1 \sqcap \cdots \sqcap B'_m \sqsubseteq \bot \in cln(\mathcal{T})$.*
3. *if $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq k\, R \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ and $B'_1 \sqcap \cdots \sqcap B'_m \sqsubseteq \geq k'\, R$ is in $\mathcal{T}$ for $k' \geq k$, then also $B_1 \sqcap \cdots \sqcap B_n \sqcap B'_1 \sqcap \cdots \sqcap B'_m \sqsubseteq \bot \in cln(\mathcal{T})$.*
4. *if $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq 1\, R \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ and $R' \sqsubseteq^{\pm} R$ is in $\mathcal{T}$, then also $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq 1\, R' \sqsubseteq \bot \in cln(\mathcal{T})$.*
5. *if $\mathsf{Dis}(R, R_1)$ or $\mathsf{Dis}(R_1, R) \in cln(\mathcal{T})$ and $R' \sqsubseteq^{\pm} R$ is in $\mathcal{T}$, then also $\mathsf{Dis}(R', R_1) \in cln(\mathcal{T})$.*
6. *if either $\geq 1\, R \sqsubseteq \bot$, or $\geq 1\, R^- \sqsubseteq \bot$, or $\mathsf{Dis}(R, R)$ is in $cln(\mathcal{T})$, then all three assertions are in $cln(\mathcal{T})$.*

Note, that in rule 4 it is enough to consider $k = 1$ because the condition (inter) ensures that concepts of the form $\geq k\, R$, for $k \geq 2$, do not occur in the left-hand side of concept inclusions when R appears in the right-hand side of a role inclusion. Also we don't need to add both inclusions $\mathsf{Dis}(R, R_1)$ and $\mathsf{Dis}(R_1, R)$ to $cln(\mathcal{T})$, since to trigger rule 5 one of them is sufficient.

The canonical interpretation can also be exploited for checking satisfiability of a KB containing negative inclusions. To establish that they are satisfied by $can(\mathcal{K})$, it suffices to verify that the interpretation $db(\mathcal{A})$ satisfies $cln(\mathcal{T})$.

Lemma 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB. Then $can(\mathcal{K})$ is a model of $\mathcal{K}$ if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

The proof follows the line of that of Lemma 12 in [4], but we need to take into account the modified definition of chase, and hence of $can(\mathcal{K})$. Now we can show that to check satisfiability of a KB it is is sufficient (and necessary) to look at $db(\mathcal{A})$ (provided we have computed $cln(\mathcal{T})$). More precisely, the next theorem shows that a contradiction on a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB may hold only if a membership assertion in the ABox contradicts a negative inclusion in the closure $cln(\mathcal{T})$.

Theorem 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB. Then $\mathcal{K}$ is satisfiable if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

Having these results, we can formulate the satisfiability problem in terms of evaluation of a first order query over the database (interpretation). In order to do so we define a translation δ from assertions in $cln(\mathcal{T})$ to FOL formulas encoding their violation:

$$\delta(B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot) = \exists x(\gamma_{B_1}(x) \wedge \cdots \wedge \gamma_{B_n}(x))$$
$$\delta(\mathsf{Dis}(R_1, R_2)) = \exists x, y(\rho_{R_1}(x, y) \wedge \rho_{R_2}(x, y))$$

where:

$$\gamma_{B_i}(x) = A(x), \text{ if } B_i = A;$$
$$\gamma_{B_i}(x) = \exists y_1, \ldots, y_k(P(x, y_1) \wedge \cdots \wedge P(x, y_k) \wedge \textstyle\bigwedge_{j<l} y_j \neq y_l), \text{ if } B_i = \geq k\, P;$$
$$\gamma_{B_i}(x) = \exists y_1, \ldots, y_k(P(y_1, x) \wedge \cdots \wedge P(y_k, x) \wedge \textstyle\bigwedge_{j<l} y_j \neq y_l), \text{ if } B_i = \geq k\, P^-;$$
$$\rho_{R_i}(x, y) = P(x, y), \text{ if } R_i = P; \qquad \rho_{R_i}(x, y) = P(y, x), \text{ if } R_i = P^-.$$

Algorithm Consistent($\mathcal{K}$)
Input: *DL-Lite*$_{horn}^{(\mathcal{HN})}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: $true$ if $\mathcal{K}$ is satisfiable, $false$ otherwise
$q_{unsat} = \bot$;
for each $I \in cln(\mathcal{T})$ **do** $q_{unsat} = q_{unsat} \vee \delta(I)$;
if ${q_{unsat}}^{db(\mathcal{A})} = \emptyset$ **then return** $true$; **else return** $false$;

Fig. 1. The algorithm Consistent

The algorithm Consistent, depicted in Fig. 1, takes as input a *DL-Lite*$_{horn}^{(\mathcal{HN})}$ KB, computes $db(\mathcal{A})$ and $cln(\mathcal{T})$, and evaluates over $db(\mathcal{A})$ the Boolean FOL query obtained by taking the union of all FOL formulas returned by the application of the above function δ to every assertion in $cln(\mathcal{T})$. In the algorithm, the symbol $\bot$ indicates a predicate whose evaluation is false in every interpretation. Therefore, when $\mathcal{K}$ contains no negative inclusions, ${q_{unsat}}^{db(\mathcal{A})} = \bot^{db(\mathcal{A})}$, and Consistent($\mathcal{K}$) returns true.

One can show that the algorithm Consistent terminates and a KB $\mathcal{K}$ is satisfiable if and only if Consistent($\mathcal{K}$) $= true$. As a direct consequence, we get the following theorem, which provides an alternative proof of an analogous result shown in [5].

Theorem 5. *In DL-Lite*$_{horn}^{(\mathcal{HN})}$*, knowledge base satisfiability is FOL-rewritable.*

Now we can characterize the computational complexity of the algorithm Consistent.

Theorem 6. *The algorithm* Consistent *is* AC^0 *in the size of the ABox and runs in exponential time in the size of the TBox.*

The following example demonstrates the worst case behaviour of the algorithm, where the size of $cln(\mathcal{T})$ is exponential in the size of $\mathcal{T}$.

Example 7. Let us consider the TBox $\mathcal{T}$ consisting of the assertions: $A'_1 \sqsubseteq A_1$, ..., $A'_n \sqsubseteq A_n$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \bot$. Then $cln(\mathcal{T})$ contains 2^n negative inclusion assertions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot$, where each B_i is either A_i or A'_i. Moreover, none of the assertions can be omitted: for every negative inclusion $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot$ there is an ABox $\mathcal{A}$ such that $\delta(I)^{db(\mathcal{A})} = \emptyset$. It is enough to take $\mathcal{A} = \{B_1(a), \ldots, B_n(a)\}$ with a a fresh object name. Therefore, in order to ensure that the algorithm detects unsatisfiability of $\langle \mathcal{T}, \mathcal{A} \rangle$, all the possible negative inclusions must be present in $cln(\mathcal{T})$ and its size is exponential in the size of $\mathcal{T}$.

Note that the algorithm Consistent can be turned into a non-deterministic PTIME algorithm for checking unsatisfiability of *DL-Lite*$_{horn}^{(\mathcal{HN})}$ KBs.

4 Query Answering

The aim of this section is to devise an algorithm for answering unions of CQs in *DL-Lite*$_{horn}^{(\mathcal{HN})}$ that is based on query reformulation, and hence can be easily implemented: in the rewriting step the assertions of the TBox are compiled into the query, then the resulting query is evaluated over the ABox without considering the TBox.

Let $Q = \bigcup_i q_i$ be a UCQ. We say that an argument of an atom in a query is *bound* if it corresponds either to a constant, or to a distinguished variable, or to a shared variable, that is, a variable occurring at least twice in the query body. Instead, an argument of an atom in a query is *unbound* if it corresponds to a nondistinguished nonshared variable. The symbol '_' is used to represent unbound variables.

Let $Q^R_{\mathcal{T}}$ denote the set of natural numbers containing 1 and all the numerical parameters k for which the concept $\geq k\, R$ occurs in $\mathcal{T}$. The *extension* $ext(\mathcal{T})$ of $\mathcal{T}$ contains

- $\geq k'\, R \sqsubseteq \geq k\, R$, for all $k, k' \in Q^R_{\mathcal{T}}$ such that $k' > k$ and $k' > k'' > k$ for no $k'' \in Q^R_{\mathcal{T}}$ and R is either a direct or inverse role in $\mathcal{T}$, and
- $\geq k\, R \sqsubseteq \geq k\, R'$, for all $k \in Q^R_{\mathcal{T}}$ and $R \sqsubseteq^{\pm} R' \in \mathcal{T}$.

A positive concept inclusion I is *applicable to an atom* $B(x)$ if I has B on the right-hand side, where $B(x) = A(x)$ if $B = A$, $B(x) = E_k P(x)$ if $B = \geq k\, P$, and $B(x) = E_k P^-(x)$ if $B = \geq k\, P^-$. A positive role inclusion I is *applicable to an atom* $P(x_1, x_2)$ if I has either P or P^- on the right-hand side. We indicate with $gr(g, I)$ the atom obtained from the atom g by applying the applicable inclusion I. Formally:

Definition 8. *Let $I \in ext(\mathcal{T})$ be a positive inclusion assertion that is applicable to the atom g. Then, $gr(g, I)$ is the formula defined as follows:*

1. *if $g = A(x)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq A$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$,*
2. *if $g = E_k R(x)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \geq k'\, R$, $k' \geq k$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$,*
3. *if $g = E_1 R(\_)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \geq k\, R$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$, for a fresh variable x,*
4. *if $g = E_k R(x)$ and $I = P_1 \sqsubseteq^{\pm} R$, then $gr(g, I) = E_k P_1(x)$,*
5. *if $g = E_k R(x)$ and $I = P_1 \sqsubseteq^{\pm} R^-$, then $gr(g, I) = E_k P_1^-(x)$,*
6. *if $g = P(x_1, x_2)$ and $I = P_1 \sqsubseteq^{\pm} P$, then $gr(g, I) = P_1(x_1, x_2)$,*
7. *if $g = P(x_1, x_2)$ and $I = P_1 \sqsubseteq^{\pm} P^-$, then $gr(g, I) = P_1(x_2, x_1)$.*

We also define the *most general unifier*, mgu, between two atoms g_1, g_2 of a query q that unify. In the case where g_1 and g_2 are respectively of the form $A(x)$ and $A(z)$, or $P(x, y)$ and $P(z, w)$, the mgu is defined as usual (see [4]), taking also into account the possible presence of inequalities $x \neq z$ (and $y \neq w$). However, we allow also an atom $P(x, y)$ to unify with $E_1 P(z)$ and $E_1 P^-(w)$. Moreover, when one of the atoms is of the form $E_k R(x)$, then the mgu is defined as follows: let $g_1 = E_{k_1} R(x)$ and $g_2 = E_{k_2} R(z)$, $k = max(k_1, k_2)$, and $x \neq z$ does not occur in q; or $g_1 = E_k R(x)$ and $g_2 = E_1 R^-(\_)$, then $mgu = E_k R(x)$.

In Fig. 2 we provide the algorithm PerfectRef, which reformulates a UCQ taking into account the PIs of a TBox $\mathcal{T}$. In the algorithm, $q[g/g']$ denotes the CQ obtained from a CQ q by replacing the atom g with a new atom g'. The function $remdup$ removes from the body of a CQ duplicated atoms, or if a query contains two atoms of the form $E_{k_1} R(x)$ and $E_{k_2} R(x)$, then it removes the atom with the smaller k_i. Furthermore, τ is a function that takes as input a CQ q and returns a new CQ obtained by replacing each occurrence of an unbound variable in q with the symbol _, whereby $P(x, \_)$ becomes $E_1 P(x)$, $P(\_, x)$ becomes $E_1 P^-(x)$ (and $P(\_, \_)$ becomes $E_1 P(\_)$). Finally, $reduce$ is a function that takes as input a CQ q and two atoms g_1 and g_2 that unify and occur

Algorithm PerfectRef(Q, $\mathcal{T}$)
Input: union of conjunctive queries Q, *DL-Lite*$_{horn}^{(\mathcal{HN})}$ TBox $\mathcal{T}$
Output: union of conjunctive queries PR
$PR := \{remdup(\tau(q)) \mid q$ is a CQ in $Q\}$;
repeat (1)
  $PR' := PR$;
  for each $q \in PR'$
  (a) **for each** g in q
    for each PI I in $ext(\mathcal{T})$
      if I is applicable to g
      then $PR := PR \cup \{remdup(q[g/gr(g, I)])\}$;
  (b) **for each** g_1, g_2 in q
    if g_1 and g_2 unify
    then $PR := PR \cup \{remdup(\tau(reduce(q, g_1, g_2)))\}$;
until $PR' = PR$;
for each $q \in PR$ (2)
  for each g in q
    if g is of the form $E_k R(x)$, $k \geq 2$, **then** replace g with $\gamma_{\geq k\, R}(x)$;
    if g is of the form $E_1 R(x)$ **then** replace g with $R(x, \_)$;
return PR.

Fig. 2. The algorithm PerfectRef

in the body of q, and returns a CQ q' obtained by applying to q the most general unifier between g_1 and g_2.

Informally, part (1) of the algorithm reformulates the query by replacing and unifying atoms, and accumulates the new queries. In this part all possible applications of the PIs, according to Definition 8, are exhausted. Part (2) performs unfolding of the atoms $E_k R(x)$. Notice that it is sufficient to perform unfolding of the atoms $E_k R(x)$ in the very end for the following reasons: first, if $k \geq 2$, then only role inclusions could be applied to the atoms of the form $R(x, y)$; however condition (inter) ensures that in this case there are no such inclusions. If $k = 1$, then such role inclusions have already been aplied in step (1). Second, one could reduce some of the atoms $R(x, y)$, but it would not produce new answers, since all variables in the new R-atoms are bound.

This algorithm is an extension of the PerfectRef algorithm devised for the logic *DL-Lite*$_{\mathcal{A}}$ [4]. The extended version has to deal with inequality atoms and with atoms of the form $E_k R(x)$ that later need to be unfolded. In contrast with the algorithm for *DL-Lite*$_{\mathcal{A}}$ the query may grow, so one also needs to take care of redundant atoms. Notice also that the number of CQs in PR may be exponential in the size of the TBox (and not only in the length of q), due to the fact that horn inclusions may cause a single CQ to become of length linear in the size of the TBox. Here, we assume that numbers in the TBox are coded in unary.

Now, to compute the answers to Q over the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we need to evaluate the set of conjunctive queries PR produced by the algorithm PerfectRef over the ABox $\mathcal{A}$ considered as a relational database. The algorithm computing $ans(Q, \mathcal{K})$ is exactly the same as in [4], so we do not present it here. Correctness of the above described query-answering technique is established in the following. We start by observing that,

as in [4], query answering can in principle be done by evaluating the query over the model $can(\mathcal{K})$.

Theorem 9. *Let $\mathcal{K}$ be a satisfiable DL-Lite$_{horn}^{(\mathcal{HN})}$ KB, and let Q be a union of conjunctive queries over $\mathcal{K}$. Then, $ans(Q, \mathcal{K}) = Q^{can(\mathcal{K})}$.*

Since $can(\mathcal{K})$ is in general infinite, we cannot compute it and evaluate Q over it. Instead, we compile the TBox into the query, thus simulating the evaluation of the query over $can(\mathcal{K})$ by evaluating a finite reformulation of the query over the ABox considered as a database. The proof of the following lemma is inspired by the one for *DL-Lite*$_{\mathcal{R}}$ in [4], but needs to take into account number restrictions and horn inclusion assertions.

Lemma 10. *Let $\mathcal{T}$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ TBox, Q a UCQ over $\mathcal{T}$, and PR the UCQ returned by* PerfectRef$(Q, \mathcal{T})$. *For every DL-Lite$_{horn}^{(\mathcal{HN})}$ ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = PR^{db(\mathcal{A})}$.*

Proof. We first introduce the preliminary notion of *witness of a tuple of constants* with respect to a CQ q in Q. Given a *DL-Lite*$_{horn}^{(\mathcal{HN})}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a CQ $q(\boldsymbol{x}) \leftarrow conj(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{K}$, and a tuple $\boldsymbol{t}$ of constants occurring in $\mathcal{K}$, a set of membership assertions $\mathcal{G}$ is a *witness* of $\boldsymbol{t}$ w.r.t. q if there exists a substitution σ from the variables $\boldsymbol{y}$ in $conj(\boldsymbol{t}, \boldsymbol{y})$ to constants in $\mathcal{G}$ such that the set of atoms in $\sigma(conj(\boldsymbol{t}, \boldsymbol{y}))$ is equal to $\mathcal{G}$. Then $\boldsymbol{t} \in q^{can(\mathcal{K})}$ iff there exists a witness $\mathcal{G}$ of $\boldsymbol{t}$ w.r.t. q such that $\mathcal{G} \subseteq chase(\mathcal{K})$. The cardinality of a witness $\mathcal{G}$, denoted by $|\mathcal{G}|$, is the number of membership assertions in $\mathcal{G}$.

Let us prove first the statement for a modified version of PerfectRef, where at the end of the algorithm we perform an additional step of unifications as in step (b) of part (1). Let us call PerfectRef$_u$ this extended version of PerfectRef.

We have that $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = Q^{can(\mathcal{K})} = \bigcup_{q \in Q} q^{can(\mathcal{K})}$, and $PR^{db(\mathcal{A})} = \bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})}$, where PR is the UCQ returned by PerfectRef$_u(Q, \mathcal{T})$. Hence, we need to show that $\bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})} = Q^{can(\mathcal{K})}$. For simplicity, we consider the case where Q consists of a single CQ q.

"$\Leftarrow$" We have to prove that $\hat{q}^{db(\mathcal{A})} \subseteq q^{can(\mathcal{K})}$, for each $\hat{q} \in PR$. Let q_{i+1} be obtained from q_i by some step of the algorithm PerfectRef$_u$. We can show that $q_{i+1}^{can(\mathcal{K})} \subseteq q_i^{can(\mathcal{K})}$ at any step. Since each query of PR is either q or a query obtained from q by repeatedly applying steps (a) and (b) of the algorithm PerfectRef$_u$, then by the rewriting in part (2), and the unification step in the end, it follows that for each $\hat{q} \in PR$, $\hat{q}^{can(\mathcal{K})} \subseteq q^{can(\mathcal{K})}$, by repeatedly applying the property $q_{i+1}^{can(\mathcal{K})} \subseteq q_i^{can(\mathcal{K})}$. Since $db(\mathcal{A}) \subseteq can(\mathcal{K})$ and CQs are monotonic queries, we get $\hat{q}^{db(\mathcal{A})} \subseteq \hat{q}^{can(\mathcal{K})} \subseteq q^{can(\mathcal{K})}$ for each CQ $\hat{q} \in PR$.

"$\Rightarrow$" We have to show that for each tuple $\boldsymbol{t} \in q^{can(\mathcal{K})}$, there exists $\hat{q} \in PR$ such that $\boldsymbol{t} \in \hat{q}^{db(\mathcal{A})}$. First, since $\boldsymbol{t} \in q^{can(\mathcal{K})}$, it follows that there exists a finite number h such that there is a witness $\mathcal{G}_h$ of $\boldsymbol{t}$ w.r.t. q contained in $chase_h(\mathcal{K})$. Moreover, w.l.o.g. we can assume that every rule cr1, cr2 and cr3 used in the construction of $chase(\mathcal{K})$ is necessary in order to generate such a witness $\mathcal{G}_h$. In the following, we say that a set S of membership assertions is an *ancestor* of a set S' of membership assertions in a set $\mathcal{S}$ of

membership assertions, if there exist $S_1, \ldots, S_n$ in $\mathcal{S}$, where $S_1 = S$ and $S_n = S'$, such that, for each $j \in \{2, \ldots, n\}$, S_j can be generated by applying a chase rule to a subset of S_{j-1}. We also say that S' is a successor of S. Furthermore, for each $i \in \{0, \ldots, h\}$, we denote with $\mathcal{G}_i$ the *pre-witness* of $\boldsymbol{t}$ w.r.t. q in $chase_h(\mathcal{K})$, defined as follows:

$$\mathcal{G}_i = \bigcup_{S' \subseteq \mathcal{G}_h} \{\, S \subseteq chase_i(\mathcal{K}) \mid \begin{array}{l} S \text{ is an ancestor of } S' \text{ in } chase_h(\mathcal{K}) \text{ and} \\ \text{there exists no successor of } S \text{ in } chase_i(\mathcal{K}) \\ \text{that is an ancestor of } S' \text{ in } chase_h(\mathcal{K}) \,\}. \end{array}$$

Now we prove by induction on i that, starting from $\mathcal{G}_h$, we can "go back" through the rule applications and find a query $\hat{q}$ in PR such that the pre-witness $\mathcal{G}_{h-i}$ of $\boldsymbol{t}$ w.r.t. q in $chase_{h-i}(\mathcal{K})$ is also a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$. To this aim, we prove that there exists $\hat{q} \in PR$ such that $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_{h-i}|$, where $|\hat{q}|$ indicates the number of atoms in the CQ $\hat{q}$ except inequality atoms. The claim then follows for $i = h$, since $chase_0(\mathcal{K}) = \mathcal{A}$.

Base step: There exists $\hat{q} \in PR$ such that $\mathcal{G}_h$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_h|$. This is an immediate consequence of the fact that (1) $q \in PR$ and (2) PR is closed with respect to step (b) of the algorithm $\mathsf{PerfectRef}_u$.

Inductive step: Suppose there exists $\hat{q} \in PR$ such that $\mathcal{G}_{h-i+1}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ in $chase_{h-i+1}(\mathcal{K})$ and $|\hat{q}| = |\mathcal{G}_{h-i+1}|$. Let us assume that $chase_{h-i+1}(\mathcal{K})$ is obtained by applying chase rule cr2 to $chase_{h-i}(\mathcal{K})$ (the proof for rules cr1 and cr3 is analogous). Hence, a PI of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \geq k\, R$, where B_j, $1 \leq j \leq n$, are basic concepts and R is a basic role, is applied in $chase_{h-i}(\mathcal{K})$ to a set of membership assertions $S' = S_{B_1}(a) \cup \cdots \cup S_{B_n}(a)$, such that $k_1 = \sharp\{b \mid R(a,b) \in chase_{h-i}(\mathcal{K})\} < k$. Therefore, $chase_{h-i+1}(\mathcal{K}) = chase_{h-i}(\mathcal{K}) \cup \{R(a, b_{k_1+1}), \ldots, R(a, b_k)\}$, where $b_{k_1+1}, \ldots, b_k \in \Gamma_N$ follow lexicographically all constants occurring in $chase_{h-i}(\mathcal{K})$. Since every rule used in the construction of $chase(\mathcal{K})$ is necessary for generating $\mathcal{G}_{h-i+1}$ and the set of membership assertions $\{R(a, b_{k_1+1}), \ldots, R(a, b_k)\}$ does not have successors in $chase_{h-i+1}(\mathcal{K})$, so $\{R(a, b_{k_1+1}), \ldots, R(a, b_k)\} \subseteq \mathcal{G}_{h-i+1}$.

Let $\{R(a, b_1), \ldots, R(a, b_{k_2})\}$, $0 \leq k_2 \leq k_1$, be the minimal set of membership assertions that has successors in $chase_{h-i+1}(\mathcal{K})$. Then, according to the definition of $\mathcal{G}_i$, the set $\{R(a, b_{k_2+1}), \ldots, R(a, b_{k_1})\} \subseteq \mathcal{G}_{h-i+1}$. By the inductive assumption, $|\hat{q}| = |\mathcal{G}_{h-i+1}|$, and $\{R(a, b_{k_2+1}), \ldots, R(a, b_k)\} \subseteq \mathcal{G}_{h-i+1}$, hence, $\hat{q}$ has to contain the atoms $R(x, y_{k_2+1}), \ldots, R(x, y_k)$, where $y_{k_2+1}, \ldots, y_k$ appear only in the mentioned predicate atoms and possibly inequalities. We show that in $\hat{q}$ there must be all the possible inequalities $y_{j_1} \neq y_{j_2}$, for $k_2 + 1 \leq j_1 < j_2 \leq k$.

By contradiction, assume that not all variables are related to each other by inequalities, i.e. there are $m \geq 2$ sets of variables $y_{k_2+1}, \ldots, y_k$ such that the variables within one set are mutually unequal and variables from different sets are not constrained to be different. It means that the part of $\hat{q}$ with $y_{k_2+1}, \ldots, y_k$ looks as follows:

$$\begin{array}{c} R(x, y_{11}), \ldots, R(x, y_{1l_1}), y_{11} \neq y_{12}, \ldots, y_{1l_1-1} \neq y_{1l_1}, \\ R(x, y_{21}), \ldots, R(x, y_{2l_2}), y_{21} \neq y_{22}, \ldots, y_{2l_2-1} \neq y_{2l_2}, \\ \ldots \\ R(x, y_{m1}), \ldots, R(x, y_{ml_m}), y_{m1} \neq y_{m2}, \ldots, y_{ml_m-1} \neq y_{ml_m}, \end{array}$$

where $l_j \geq 1$, $j \in \{1, \ldots, m\}$, $\sum l_j = k - k_2$. Therefore, $\hat{q}$ has to be the result of unfolding by part (2) of the algorithm $\mathsf{PerfectRef}_u$ of a query q_1 with the corresponding

part $E_{l_1}R(x), \ldots, E_{l_m}R(x)$. However, the algorithm cannot produce such a query: the function *remdup* would keep in q_1 only one such atom $E_{l_{max}}R(x)$ with $l_{max} < k-k_2$, $l_{max} = \max_j\{l_j\}$, which unfolded would contain less than $k - k_2$ atoms of the form $R(x, y_j)$. The contradiction rises from the assumption $m \geq 2$. So, m has to be equal to 1 and all variables $y_{k_2+1}, \ldots, y_k$ appear in pairwise inequalities in $\hat{q}$.

Thus, there exists a query q_1 in PR_1 that contains the atom $E_{k-k_2}R(x)$, $k-k_2 \leq k$, and $\hat{q}$ is obtained from q_1 by part (2) of the algorithm. Then, by step (a) it follows that there exists a query $q_2 = \mathit{remdup}(\hat{q}_1[E_{k-k_2}R(x)/B_1(x) \wedge \cdots \wedge B_n(x)])$ in PR_1 such that $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. q_2. Let $\hat{q}_1$ be the result of unfolding q_2 by part (2) of the algorithm, then $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}_1$ as well and $|\hat{q}_1| \geq |\mathcal{G}_{h-i}|$.

If $|\hat{q}_1| > |\mathcal{G}_{h-i}|$ it implies that there exists at least one membership assertion f in $\mathcal{G}_{h-i}$ such that there exist at least two atoms g_1, g_2 in $\hat{q}_1$ that both unify with f. Hence g_1 and g_2 unify, and by step (b) of the algorithm it follows that in PR there exists $q_3 = \mathit{remdup}(\tau(\mathit{reduce}(\hat{q}_1, g_1, g_2)))$, $|q_3| < |\hat{q}_1|$ and $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. q_3. If $|q_3| > |\mathcal{G}_{h-i}|$ then by applying the argument consecutively there is a query $\hat{q}_2$ such that $|\hat{q}_2| = |\mathcal{G}_{h-i}|$ and $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}_2$, which proves the claim.

Finally, we observe that unification of atoms of the query at the end of $\mathsf{PerfectRef}_u$ does not add new answers to the set $ans(Q, \langle \mathcal{T}, \mathcal{A}\rangle)$ since all variables in the new R atoms introduced in part (2) are bound. Therefore, the theorem holds also for the algorithm PerfectRef. □

Based on the above property, we are finally able to establish correctness of the algorithm Answer and its computational complexity.

Theorem 11. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB, Q a UCQ over $\mathcal{T}$, and $\boldsymbol{t}$ a tuple of constants in $\mathcal{K}$. Then, $\boldsymbol{t} \in ans(Q, \mathcal{K})$ if and only if $\boldsymbol{t} \in \mathsf{Answer}(Q, \mathcal{K})$.*

Theorem 12. *The algorithm* Answer *is exponential in the size of the TBox, and* AC^0 *in the size of the ABox (data complexity).*

Note that we can modify PerfectRef to get an algorithm for the decision problem associated with the query answering problem that runs in nondeterministic polynomial time in combined complexity: it nondeterministically returns one of the CQs from the reformulation of the input query (polynomially many rewriting steps) and in polynomial time checks whether a tuple is in the answer to this CQ. The corresponding version of Answer also runs in nondeterministic polynomial time.

5 Conclusions

This paper presents practical algorithms for query answering and knowledge base satisfiability in *DL-Lite*$_{horn}^{(\mathcal{HN})}$. They are based on the same idea as those devised for the original *DL-Lite* logics. The distinguishing feature of these algorithms is a separation between TBox and ABox reasoning, which enables an interesting modularization of query answering: the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine. We showed that the algorithms are sound and complete.

The complexity of the developed algorithms is AC^0 w.r.t. data complexity, NP in the size of the TBox and NP w.r.t. combined complexity.

References

1. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics **X** (2008) 133–173
2. Rodríguez-Muro, M.: Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics. PhD thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano (2010)
3. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002). (2002) 233–246
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 **39**(3) (2007) 385–429
5. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research **36** (2009) 1–69
6. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QUONTO: QUerying ONTOlogies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005). (2005) 1670–1671
7. Poggi, A., Rodríguez-Muro, M., Ruzzi, M.: Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In Clark, K., Patel-Schneider, P.F., eds.: Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC). (2008)
8. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
9. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)