Combined FO Rewritability for Conjunctive Query Answering in DL-Lite

R. Kontchakov,[1] C. Lutz,[2] D. Toman,[3] F. Wolter[4] and M. Zakharyaschev[1]

[1] School of CS and Information Systems
Birkbeck College London, UK
`{roman,michael}@dcs.bbk.ac.uk`

[2] Fachbereich Mathematik und Informatik
Universität Bremen, Germany
`clu@informatik.uni-bremen.de`

[3] D.R. Cheriton School of CS
University of Waterloo, Canada
`david@uwaterloo.ca`

[4] Department of Computer Science
University of Liverpool, UK
`frank@csc.liv.ac.uk`

1 Introduction

Standard description logic (DL) reasoning services such as satisfiability and subsumption mainly aim to support TBox design. When the design has been completed and the TBox is used in an actual application, it is usually combined with instance data stored in an ABox, and therefore query answering becomes the most important reasoning service. In this context, conjunctive queries (CQs) have become very popular, and a rich literature on the subject has emerged in recent years; see, e.g., [5, 9, 12–14, 19, 22]. Unfortunately, CQ answering in expressive DLs of the $\mathcal{ALC}$ and $\mathcal{SHIQ}$ families is often computationally harder than satisfiability and subsumption [11, 15], while in typical applications it is also much more time critical. To address this problem, it has been proposed to use standard relational database management systems (RDBMS) for DL query processing [7]. Indeed, the *DL-Lite* family of DLs was designed to allow CQ answering using standard RDBMSs while still capturing as much basic expressivity of ontology languages as possible [2, 3, 6–9, 20].

The basic idea behind CQ answering in *DL-Lite* is to store the ABox as a relational instance in the RDBMS and to rewrite the query to account for the implicit information represented in the TBox, of which the RDBMS is unaware. The foundation of this approach is captured formally by the notion of *FO rewritability* [8].[1] Recently, an alternative approach to using RDBMSs for CQ answering in DLs has been proposed by generalizing FO rewritability to *combined FO rewritability* [17, 18]. In this new approach, the TBox is incorporated mainly into the ABox rather than into the query, though limited query rewriting is still necessary. Formally, a DL enjoys combined FO rewritability if, given a knowledge base $(\mathcal{T}, \mathcal{A})$ and a CQ q, it is possible to effectively rewrite (i) the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$ into a finite FO structure $\mathfrak{M}$ (independently of q) and (ii) q and (possibly) $\mathcal{T}$ into an FO query $q^\dagger$ (independently of $\mathcal{A}$) in such a way that query answers are preserved—i.e., the answers to $q^\dagger$ over $\mathfrak{M}$ are the same as the certain answers to q over $(\mathcal{T}, \mathcal{A})$. Important members of the

[1] The term used in [8] is *FO reducibility*, but we prefer 'rewritability' to avoid confusion with the complexity class FO from descriptive complexity.

$\mathcal{EL}$ family of DLs, which is incomparable in expressive power to the *DL-Lite* family, enjoy combined FO rewritability [17, 18]. In fact, this was originally the main motivation for introducing the more general notion of rewritability: while a DL can only enjoy FO rewritability if its data complexity for CQ answering is in LogSpace [8, 9] (actually, in AC^0 [3]), PTime DLs such as those in the $\mathcal{EL}$ family can allow combined FO rewritability.

However, even for DLs such as *DL-Lite* in which the data complexity of CQ answering is in AC^0, the novel approach to CQ answering can be useful. On the one hand, this approach presupposes authority over the data to implement ABox rewriting, which makes it unsuitable for applications in which only a query interface is provided such as virtual data integration. On the other hand, though, the increased flexibility obtained by allowing to rewrite both the query *and the ABox* can be expected to enable more efficient query execution. In this context, it is interesting to note that the query rewriting for *DL-Lite* of the original approach [8, 9] involves an exponential blowup in the size of the query (the rewritten query is a union of $(|\mathcal{T}| \cdot |q|)^{|q|}$ many CQs of length $\leq |q|$), whereas the approach for $\mathcal{EL}$ [17, 18] does not involve such a blowup.

In this paper, we apply the novel approach to CQ answering pursued in [17, 18] to the *DL-Lite* family of DLs. More precisely, we concentrate on the *DL-Lite*$^{\mathcal{N}}_{horn}$ [2, 3] (which properly contains *DL-Lite*$_{\mathcal{F},\sqcap}$ of [8]) and show that it is possible to rewrite (i) every *DL-Lite*$^{\mathcal{N}}_{horn}$ TBox $\mathcal{T}$ and ABox $\mathcal{A}$ (independently of a given CQ q) to a finite FO structure of size $\mathcal{O}(|\mathcal{T}| \cdot |\mathcal{A}|)$ and (ii) every CQ q (independently of both $\mathcal{T}$ and $\mathcal{A}$) into an FO query $q^\dagger$ such that the answers are preserved. The proof is much less trivial than one might expect since, compared to $\mathcal{EL}$, the presence of inverse roles in *DL-Lite* complicates matters considerably. This also results, in general, in the rewritten query $q^\dagger$ to be of size $2^{\mathcal{O}(|q|)}$ and thus exponential in the size of q (which is commonly small), but independent of the size of $\mathcal{T}$ (which is often large). However, we identify a large and natural class of queries where the size of $q^\dagger$ is only $\mathcal{O}(|q|^2)$.

We perform an initial experimental evaluation of our approach and compare its performance to the approach of [8, 9] implemented in the QuOnto system [1, 21]. Summarized, the evaluation confirms the intuition that our combined approach should typically lead to smaller query execution times. In particular, the QuOnto rewriting often significantly blows up queries that use concepts with many subsumees, which typically leads to long query execution times. The execution times in our own approach are typically small if we stay within the identified query class that admits polynomial rewriting. On the negative side, the ABox rewriting can take long and should either be implemented offline or using (yet to be developed) incremental techniques.

2 *DL-Lite*$^{\mathcal{N}}_{horn}$

The *DL-Lite* family of DLs was originally introduced in [7–9]. We consider *DL-Lite*$^{\mathcal{N}}_{horn}$, the most expressive dialect of *DL-Lite* that does not include role inclusions and for which query answering is in AC^0 for data complexity; for relations to other logics in the *DL-Lite* family see [3, 2]. Let $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ be countably infinite sets of *individual names*, *concept names*, and *role names*.

Roles R and *concepts* C of *DL-Lite*$^{\mathcal{N}}_{horn}$ are defined as follows, where $P \in \mathsf{N_R}$, $A \in \mathsf{N_C}$ and $m > 0$:

$$R ::= P \mid P^- \qquad \text{and} \qquad C ::= \top \mid \bot \mid A \mid \geq m\, R.$$

As usual, we write $\exists R$ for $\geq 1\, R$ and identify $(R^-)^-$ with R. A *DL-Lite*$^{\mathcal{N}}_{horn}$ *TBox* is a finite set of *concept inclusions* (*CIs*) of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C$, where $C_1, \ldots, C_n$ and C are concepts. For example, the CIs $\geq 2\, R \sqsubseteq \bot$ and $A \sqcap B \sqsubseteq \bot$ say that the role R is functional and that concepts A and B are disjoint, respectively. An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $R(a, b)$, where A is a concept *name*, R a role and $a, b \in \mathsf{N_I}$. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names occurring in $\mathcal{A}$. A *DL-Lite*$^{\mathcal{N}}_{horn}$ *knowledge base* (*KB*) is a pair $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox. The semantics of the *DL-Lite*$^{\mathcal{N}}_{horn}$ constructs is defined in the standard way [4], with the unique names assumption for ABox individuals. For $\mathcal{I}$ an interpretation, C_1, C_2 concepts, R a role, and a, b individual names, we use the notation $\mathcal{I} \models C_1 \sqsubseteq C_2$, $\mathcal{I} \models C_1(a)$, and $\mathcal{I} \models R(a, b)$ with the usual meaning. A KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is *consistent* if there is an interpretation $\mathcal{I}$ with $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. In this case, we write $\mathcal{I} \models \mathcal{K}$ and say that $\mathcal{I}$ is a *model of* $\mathcal{K}$.

Let $\mathsf{N_V}$ be a countably infinite set of *variables.* Taken together, the sets $\mathsf{N_V}$ and $\mathsf{N_I}$ form the set $\mathsf{N_T}$ of *terms.* A *first-order* (*FO*) *query* is a first-order formula $q = \varphi(\boldsymbol{v})$ in the signature $\mathsf{N_C} \cup \mathsf{N_R}$ with terms from $\mathsf{N_T}$, where the concept and role names are treated as unary and binary predicates, respectively, and the sequence $\boldsymbol{v} = v_1, \ldots, v_k$ of variables from $\mathsf{N_V}$ contains all the free variables of φ. The free variables are called the *answer variables* of q, and q is *k-ary* if it has k answer variables. A *conjunctive query* (CQ, for short) is a first-order query of the form $q = \exists \boldsymbol{u}\, \psi(\boldsymbol{u}, \boldsymbol{v})$, where ψ is a conjunction of *concept atoms* $A(t)$ and *role atoms* $P(t, t')$ with $t, t' \in \mathsf{N_T}$. The variables in $\boldsymbol{u}$ are called the *quantified variables* of q. We denote by $\mathsf{var}(q)$ the set of all variables in $\boldsymbol{u}$ and $\boldsymbol{v}$, by $\mathsf{qvar}(q)$ the set of quantified variables, by $\mathsf{avar}(q)$ the set of answer variables, and by $\mathsf{term}(q)$ the set of terms in q.

For $\mathcal{I}$ an interpretation, $q = \varphi(\boldsymbol{v})$ a k-ary FO query, and $a_1, \ldots, a_k \in \mathsf{N_I}$, we write $\mathcal{I} \models q[a_1, \ldots, a_k]$ if $\mathcal{I}$ satisfies q with v_i assigned to $a_i^{\mathcal{I}}$, $1 \leq i \leq k$. A *certain answer* for a k-ary CQ q and a KB $\mathcal{K}$ is a tuple $(a_1, \ldots, a_k)$ such that $a_1, \ldots, a_k$ occur in $\mathcal{K}$ and $\mathcal{I} \models q[a_1, \ldots, a_k]$ for each model $\mathcal{I}$ of $\mathcal{K}$. We use $\mathsf{cert}(q, \mathcal{K})$ to denote the set of all certain answers for q and $\mathcal{K}$. This defines the query answering problem studied in this paper: given a *DL-Lite*$^{\mathcal{N}}_{horn}$ KB $\mathcal{K}$ and a CQ q, compute $\mathsf{cert}(q, \mathcal{K})$.

3 ABox Rewriting / Canonical Interpretations

The ABox rewriting part of our approach consists of expanding the ABox $\mathcal{A}$ to a *canonical interpretation* $\mathcal{I}_{\mathcal{K}}$ for the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The ABox $\mathcal{A}$ can be viewed as a 'fragment' of $\mathcal{I}_{\mathcal{K}}$, i.e., every ABox individual is an element of $\Delta^{\mathcal{I}_{\mathcal{K}}}$, but there are also additional elements that witness existential and number restrictions. Expanding $\mathcal{A}$ to $\mathcal{I}_{\mathcal{K}}$ involves adding these additional elements, connecting them to the ABox individuals and other additional elements by means of roles, and adding extra concept memberships for the ABox individuals.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a *DL-Lite*$^{\mathcal{N}}_{horn}$ knowledge base. For a role R, let

$$\mathsf{dom}_{\mathcal{A}}(R) \;=\; \{a \in \mathsf{Ind}(\mathcal{A}) \mid \exists b \in \mathsf{Ind}(\mathcal{A})\; R(a,b) \in \mathcal{A}\}.$$

We call a role R *generating w.r.t.* $\mathcal{K}$ if there are $a \in \mathsf{Ind}(\mathcal{A})$ and $R_1, \ldots, R_n$ such that $R_n = R$, $\mathcal{K} \models \exists R_1(a)$, $a \notin \mathsf{dom}_{\mathcal{A}}(R_1)$, and $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ for $1 \leq i < n$. In other words, R is generating w.r.t. $\mathcal{K}$ if every model $\mathcal{I}$ of $\mathcal{K}$ must contain a $d \in \Delta^{\mathcal{I}}$ with an incoming R-arrow but there is an $\mathcal{I}$ where no such d is identified by an individual in the ABox. The *canonical interpretation* $\mathcal{I}_{\mathcal{K}}$ for $\mathcal{K}$ is defined now as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_{\mathcal{K}}} &= \mathsf{Ind}(\mathcal{A}) \cup \{x_R \mid R \text{ is generating w.r.t. } \mathcal{K}\},\\
A^{\mathcal{I}_{\mathcal{K}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{x_R \in \Delta^{\mathcal{I}} \mid \mathcal{K} \models \exists R^- \sqsubseteq A\},\\
P^{\mathcal{I}_{\mathcal{K}}} &= \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid P(a,b) \in \mathcal{A}\} \cup\\
&\qquad \{(a, x_P) \mid \mathcal{K} \models \exists P(a) \text{ and } a \notin \mathsf{dom}_{\mathcal{A}}(P)\} \cup\\
&\qquad \{(x_{P^-}, a) \mid \mathcal{K} \models \exists P^-(a) \text{ and } a \notin \mathsf{dom}_{\mathcal{A}}(P^-)\} \cup\\
&\qquad \{(x_S, x_P) \mid \mathcal{K} \models \exists S^- \sqsubseteq \exists P \text{ and } P \neq S^-\} \cup\\
&\qquad \{(x_{P^-}, x_S) \mid \mathcal{K} \models \exists S^- \sqsubseteq \exists P^- \text{ and } P \neq S\},\\
a^{\mathcal{I}_{\mathcal{K}}} &= a.
\end{aligned}$$

In the above definition, P ranges over role names, while R and S range over role names and their inverses. We do not discuss in detail how $\mathcal{I}_{\mathcal{K}}$ can be constructed in the context of database systems here; for a similar construction see [17]. Note that $\mathcal{I}_{\mathcal{K}}$ is finite by definition, and that $\Delta^{\mathcal{I}_{\mathcal{K}}} \setminus \mathsf{Ind}(\mathcal{A})$ can typically be expected to be relatively small.

Example 1. Consider the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with

$$\begin{aligned}
\mathcal{T} &= \{A_1 \sqsubseteq A,\;\; A_2 \sqsubseteq A,\;\; \exists P^- \sqsubseteq \exists S,\;\; \exists S^- \sqsubseteq \exists R,\;\; A \sqsubseteq \exists P\},\\
\mathcal{A} &= \{A_1(a),\;\; A_2(b),\;\; S(a,b)\}.
\end{aligned}$$

Then the canonical interpretation $\mathcal{I}_{\mathcal{K}}$ is as follows:

$$\begin{aligned}
&\Delta^{\mathcal{I}_{\mathcal{K}}} = \{a, b, x_P, x_S, x_R\},\\
&A^{\mathcal{I}_{\mathcal{K}}} = \{a, b\}, \qquad A_1^{\mathcal{I}_{\mathcal{K}}} = \{a\}, \qquad A_2^{\mathcal{I}_{\mathcal{K}}} = \{b\},\\
&P^{\mathcal{I}_{\mathcal{K}}} = \{(a, x_P), (b, x_P)\}, \quad S^{\mathcal{I}_{\mathcal{K}}} = \{(a,b), (x_P, x_S)\}, \quad R^{\mathcal{I}_{\mathcal{K}}} = \{(b, x_R), (x_S, x_R)\}.
\end{aligned}$$

We note that $\mathcal{I}_{\mathcal{K}}$ is *not* always a model of $\mathcal{K}$. Indeed, this cannot be expected since *DL-Lite*$^{\mathcal{N}}_{horn}$ does not enjoy the finite model property whereas $\mathcal{I}_{\mathcal{K}}$ needs to be finite to be stored as a relational instance. As a concrete example, consider $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{A \sqsubseteq \exists P,\; \geq 2\, P^- \sqsubseteq \bot\}$ and $\mathcal{A} = \{A(a), A(b)\}$. Then $x_P \in (\geq 2\, P^-)^{\mathcal{I}_{\mathcal{K}}}$, and so $\mathcal{I}_{\mathcal{K}} \not\models \mathcal{K}$. From the perspective of query answering, this is not a problem. What is more problematic is that $\mathcal{I}_{\mathcal{K}}$ does not give the right answers to queries, i.e., it is not the case that $\mathcal{I}_{\mathcal{K}} \models q[\mathbf{a}]$ iff $\mathcal{K} \models q[\mathbf{a}]$ for all k-ary CQs q and k-tuples $\mathbf{a} \subseteq \mathsf{Ind}(\mathcal{A})$. In contrast, the 'unravelling' of $\mathcal{I}_{\mathcal{K}}$ into an (infinite) interpretation $\mathcal{U}_{\mathcal{K}}$ as introduced in the following *does* give the right answers to queries.

A *path* in $\mathcal{I}_{\mathcal{K}}$ is a finite sequence $ax_{R_1} \cdots x_{R_n}$, for $n \geq 0$, where

- $a \in \mathsf{Ind}(\mathcal{A})$;
- $(a^{\mathcal{I}_{\mathcal{K}}}, x_{R_1}) \in R_1^{\mathcal{I}_{\mathcal{K}}}$;
- $(x_{R_i}, x_{R_{i+1}}) \in R_i^{\mathcal{I}_{\mathcal{K}}}$, for $1 \leq i < n$;
- $R_{i+1} \neq R_i^-$, for $1 \leq i < n$.

We denote by $\mathsf{paths}(\mathcal{I}_{\mathcal{K}})$ the set of all paths in $\mathcal{I}_{\mathcal{K}}$ and by $\mathsf{tail}(p)$ the last element in $p \in \mathsf{paths}(\mathcal{I}_{\mathcal{K}})$. Then $\mathcal{U}_{\mathcal{K}}$ is defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{U}_{\mathcal{K}}} &= \mathsf{paths}(\mathcal{I}_{\mathcal{K}}); \\
a^{\mathcal{U}_{\mathcal{K}}} &= a \text{ for all } a \in \mathsf{Ind}(\mathcal{A}); \\
A^{\mathcal{U}_{\mathcal{K}}} &= \{p \mid \mathsf{tail}(p) \in A^{\mathcal{I}_{\mathcal{K}}}\}; \\
P^{\mathcal{U}_{\mathcal{K}}} &= \{(a, b) \mid a, b \in \mathsf{Ind}(\mathcal{A}) \wedge P(a, b) \in \mathcal{A}\} \cup \\
&\quad\ \{(s, s \cdot x_P) \mid s \cdot x_P \in \Delta^{\mathcal{U}_{\mathcal{K}}}\} \cup \{(s \cdot x_{P^-}, s) \mid s \cdot x_{P^-} \in \Delta^{\mathcal{U}_{\mathcal{K}}}\},
\end{aligned}$$

where '$\cdot$' denotes concatenation. Just like $\mathcal{I}_{\mathcal{K}}$, $\mathcal{U}_{\mathcal{K}}$ is in general not a model of $\mathcal{K}$. A simple example is given by $\mathcal{T} = \{A \sqsubseteq\ \geq 2\, P\}$ and $\mathcal{A} = \{A(a)\}$, where we have $\mathcal{U}_{\mathcal{K}} \not\models A \sqsubseteq\ \geq 2P$ because a is P-related only to x_P in $\mathcal{U}_{\mathcal{K}}$. Nevertheless, $\mathcal{U}_{\mathcal{K}}$ gives the right answers to all conjunctive queries. A proof of the following theorem can be found in the full version of this paper [16].

Theorem 1. *For every satisfiable DL-Lite$^{\mathcal{N}}_{horn}$ KB $\mathcal{K}$, every k-ary conjunctive query q, and every k-tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\mathcal{K} \models q[\boldsymbol{a}]$ iff $\mathcal{U}_{\mathcal{K}} \models q[\boldsymbol{a}]$.*

In the next section, we show how to rewrite the query q such that the answers to q in $\mathcal{U}_{\mathcal{K}}$ (which is infinite and cannot be stored in a database) are identical to the answers to the rewritten query in $\mathcal{I}_{\mathcal{K}}$, thus paving the way to using $\mathcal{I}_{\mathcal{K}}$ as a database instance for query answering. In the rewriting, we exploit several structural properties of $\mathcal{I}_{\mathcal{K}}$ and $\mathcal{U}_{\mathcal{K}}$ that follow immediately from the definitions. Of particular interest are the following two properties:

(p1) If $(d, e), (d, e') \in R^{\mathcal{I}_{\mathcal{K}}}$ with $e \neq e'$, then $d \in \mathsf{Ind}(\mathcal{A})$ or $d = x_{R^-}$.
(p2) If $(d, e), (d, e') \in R^{\mathcal{U}_{\mathcal{K}}}$ with $e \neq e'$, then $d \in \mathsf{Ind}(\mathcal{A})$.

We also use an additional concept name aux with the interpretations $\mathsf{aux}^{\mathcal{I}_{\mathcal{K}}} = \Delta^{\mathcal{I}_{\mathcal{K}}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{aux}^{\mathcal{U}_{\mathcal{K}}} = \Delta^{\mathcal{U}_{\mathcal{K}}} \setminus \mathsf{Ind}(\mathcal{A})$. Thus, aux distinguishes the elements of $\mathcal{I}_{\mathcal{K}}$ used to satisfy existential restrictions and number restrictions from those that correspond to individual names.

4 Query Rewriting

We present a procedure that rewrites a k-ary CQ q into an FO query $q^\dagger$ such that $\mathcal{I}_{\mathcal{K}} \models q^\dagger[\boldsymbol{a}]$ iff $\mathcal{U}_{\mathcal{K}} \models q[\boldsymbol{a}]$, for all KBs $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and all k-tuples $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$. Using Theorem 1 we obtain the intended answers to $\mathcal{K}$ and q by using an RDBMS to execute $q^\dagger$ over $\mathcal{I}_{\mathcal{K}}$ stored as a relational instance.

Our construction of $q^\dagger$ proceeds in two steps. First, we consider only queries q satisfying a syntactic condition of being *free of bad spikes*. For queries of this sort, we construct a rewriting q^* of size $\mathcal{O}(|q|^2)$. In the second step, we show how an unrestricted query q can be transformed into a query $q^\dagger$ that is a disjunction of queries free of bad spikes and such that the answers to q and $q^\dagger$ coincide on interpretations $\mathcal{U}_\mathcal{K}$. The size of $q^\dagger$ is $2^{\mathcal{O}(|q|)}$. Applying the first step to each disjunct, we thus obtain a rewriting of q of size $2^{\mathcal{O}(|q|)^2}$. We note that the first step is quite useful by itself as it establishes queries free of bad spikes as a large and natural class of queries that can be rewritten with only a polynomial blowup.

4.1 Polynomial Rewriting

To simplify the notation, we sometimes identify a CQ with the set of its atoms and do not distinguish atoms $P(u,v)$ and $P^-(v,u)$ in queries. To prepare for the rewriting of arbitrary queries, which uses the rewriting presented here as a sub-procedure, we allow the input query q to contain atoms $\neg\mathsf{aux}(v)$. Note that this is a purely technical issue and in applications $\mathsf{aux}(v)$ can be completely hidden from the user.

Definition 1. Let q be a conjunctive query. A subquery

$$q^c = \{R_0(t_0,t_1), R_1(t_1,t_2), \ldots, R_{n-1}(t_{n-1},t_n)\} \subseteq q$$

is said to be a *cycle* if $t_0 = t_n$, $t_i \neq t_j$ for $0 \leq i < j < n$, and $R_0 \neq R_1^-$ if $n = 2$.

For $n \geq 3$ each 5-tuple $(t_i, R_i, t_{i+_n 1}, R_{i+_n 1}, t_{i+_n 2})$ with $i < n$ and $R_i = R_{i+_n 1}^-$ is called a *spike* in q^c, where $+_n$ denotes addition modulo n. The spike is *bad w.r.t. q* if $t_{i+_n 1}$ is a quantified variable and $\neg\mathsf{aux}(t_{i+_n 1}) \notin q$. Now we call q^c *free of bad spikes* if no spike in q^c is bad w.r.t. q, and q is *free of bad spikes* if every cycle $q^c \subseteq q$ is.

The condition that $R_0 \neq R_1^-$ for $n = 2$ in the definition of cycles ensures that the query $\{P(u,v)\}$, which is identified with $\{P(u,v), P^-(v,u)\}$, is not regarded as a cycle. Fix a query $q = \exists \boldsymbol{u}\, \varphi$ that is free of bad spikes. Define inductively sets $\mathsf{Id}_q^{(i)}$ by taking $\mathsf{Id}_q^{(0)} = \{(t,t,\emptyset) \mid t \in \mathsf{term}(q)\}$ and, for $i > 0$,

$$\begin{aligned}\mathsf{Id}_q^{(i+1)} \;=\; & \mathsf{Id}_q^{(i)} \cup \\ & \{(t_1,t_2,\{R\}) \mid \exists (s_1,s_2,\mathcal{R}) \in \mathsf{Id}_q^{(i)}\ \exists R(t_1,s_1), R(t_2,s_2) \in q\ (R^- \notin \mathcal{R})\} \cup \\ & \qquad \{(t_1,t_2,\mathcal{R}_1 \cup \mathcal{R}_2)) \mid \exists s\ (t_1,s,\mathcal{R}_1) \in \mathsf{Id}_q^{(i)} \wedge (s,t_2,\mathcal{R}_2) \in \mathsf{Id}_q^{(i)}\}.\end{aligned}$$

Set $\mathsf{Id}_q = \bigcup_{i \geq 0} \mathsf{Id}_q^{(i)}$ and let $\mathsf{Cyc}(q)$ be the set of quantified variables v occurring on some cycle in q. We define the *rewriting* of q as $q^* \;=\; \exists \boldsymbol{u}\, (\varphi \wedge \varphi_1 \wedge \varphi_2)$, where

$$\varphi_1 \;=\; \bigwedge_{v \in \mathsf{avar}(q) \cup \mathsf{Cyc}(q)} \neg\mathsf{aux}(v),$$

$$\varphi_2 \;=\; \bigwedge_{\substack{R(t,s), R(t',s') \in q,\\ (s,s',\mathcal{R}) \in \mathsf{Id}_q, R^- \notin \mathcal{R}}} \big((s = x_R) \rightarrow (t = t')\big).$$

The main result of this paper is the following theorem whose rather non-trivial proof can be found in [16]:

Theorem 2. *Let $q = \exists \boldsymbol{u}\, \varphi$ be a k-ary CQ free of bad spikes. Then, for every knowledge base $\mathcal{K}$ and every k-tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\mathcal{I}_\mathcal{K} \models q^*[\boldsymbol{a}]$ iff $\mathcal{U}_\mathcal{K} \models q[\boldsymbol{a}]$.*

Together, Theorems 1 and 2 show that the rewritten query q^* is as required. Clearly, the size of q^* is $\mathcal{O}(|q|^2)$. We now give an intuitive explanation of this rewriting whose general purpose is to allow only matches in $\mathcal{I}_\mathcal{K}$ that can be 'reproduced' in $\mathcal{U}_\mathcal{K}$.

The conjunct φ_1 of q^* ensures that answer variables and variables in cycles are matched to ABox individuals. The former is required independently of whether we use $\mathcal{U}_\mathcal{K}$ or $\mathcal{I}_\mathcal{K}$ and just reflects the fact that query answers can contain ABox individuals only. The latter is more interesting. First, it is readily seen that no variable in a cycle can be matched to an element of $\mathsf{aux}^{\mathcal{U}_\mathcal{K}}$ if the cycle is free of bad spikes and thus, completeness is not compromised. To see why variables in cycles *must* be matched to ABox individuals to ensure soundness, consider the query $q_0 = \exists v_1, v_2, v_3 (P_1(v_1, v_2) \land P_2(v_2, v_3) \land P_3(v_3, v_1))$. As q_0 is a cycle,

$$q_0^* = \exists v_1 v_2 v_3 (P_1(v_1, v_2) \land P_2(v_2, v_3) \land P_3(v_3, v_1) \land \neg\mathsf{aux}(v_1) \land \neg\mathsf{aux}(v_2) \land \neg\mathsf{aux}(v_3)),$$

where the $\neg\mathsf{aux}(v_i)$ are introduced since the v_i are in a cycle. To see that without this rewriting we would not get correct answers, consider $\mathcal{K}_0 = (\mathcal{T}_0, \mathcal{A}_0)$ with

$$\mathcal{T}_0 = \{A \sqsubseteq \exists P_1,\ \ \exists P_1^- \sqsubseteq \exists P_2,\ \ \exists P_2^- \sqsubseteq \exists P_3,\ \ \exists P_3^- \sqsubseteq \exists P_1\}, \quad \mathcal{A}_0 = \{A(a)\}.$$

Then $\mathcal{K} \not\models q_0$ and thus $\mathcal{U}_\mathcal{K} \not\models q_0$, but $\mathcal{I}_\mathcal{K} \models q_0$. On the other hand, $\mathcal{I}_\mathcal{K} \not\models q_0^*$.

To explain the conjunct φ_2, consider $q_1 = \exists u\,(P(v, u) \land P(w, u))$. Then we have $(v, w, \{P\}) \in \mathsf{Id}_{q_1}^{(1)}$ and, therefore,

$$q_1^* = \exists u\,(P(v, u) \land P(w, u) \land \neg\mathsf{aux}(v) \land \neg\mathsf{aux}(w) \land (u = x_P \to v = w)),$$

where $\neg\mathsf{aux}(v) \land \neg\mathsf{aux}(w)$ is introduced by φ_1 since v and w are answer variables and the last conjunct is introduced by φ_2. Completeness is not compromised because, by **(p2)**, no non-ABox element can be in a relation $R^{\mathcal{U}_\mathcal{K}}$ to two distinct elements and $u = x_P$ implies that u is matched with a non-ABox element. To see that the last conjunct is necessary to ensure soundness, consider $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$, for $\mathcal{T}_1 = \{A \sqsubseteq \exists P\}$ and $\mathcal{A}_1 = \{A(a), A(b)\}$. Then $\mathcal{K}_1 \not\models q_1[a, b]$ and thus $\mathcal{U}_{\mathcal{K}_1} \not\models q_1[a, b]$, but $\mathcal{I}_{\mathcal{K}_1} \models q_1[a, b]$. On the other hand, $\mathcal{I}_{\mathcal{K}_1} \not\models q_1^*[a, b]$. The use of the pre-condition $u = x_P$ instead of $\mathsf{aux}(u)$ for identifying v and w is one of the main differences to the rewriting for $\mathcal{EL}$ in [17]. It reflects property **(p1)** of $\mathcal{I}_\mathcal{K}$.

4.2 Exponential Rewriting

We now show how an unrestricted query q can be transformed into a query $q^\dagger$ that is a disjunction of queries free of bad spikes and such that the answers to q and $q^\dagger$ coincide on the interpretations $\mathcal{U}_\mathcal{K}$.

Given a spike $S = (t_0, R_0, t_1, R_1, t_2)$, its *center variable* $v(S)$ is t_1 (which must, by definition, be a quantified variable). We rewrite $q = \exists \boldsymbol{u}\, \varphi$ into a disjunction of the form

$$q^\dagger \;=\; \bigvee_i \exists \boldsymbol{u}\, (\varphi_i \land \psi_i \land \chi_i), \qquad (\dagger)$$

where

1. each φ_i is a substitution instance of φ (obtained from φ by replacing terms);
2. each ψ_i is a conjunction of atoms $\mathsf{aux}(t)$;

3. each χ_i is a conjunction of negated atoms $\neg\mathsf{aux}(t)$;
4. for each φ_i and each spike S in a cycle in φ_i, either $v(S)$ is not quantified or $\neg\mathsf{aux}(v(S))$ is a conjunct of χ_i.

The last condition guarantees that each disjunct is free of bad spikes. Indeed, each disjunct is of the form discussed in the previous section and, using Theorem 2, we can apply the rewriting from the previous section to each disjunct of $q^\dagger$ and get a rewriting for q that can be passed to an RDBMS for execution.

Theorem 3. *For every k-ary CQ q, one can construct an FO query $q^\dagger$ of the form (†) such that, for every knowledge base $\mathcal{K}$ and k-tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{U}_\mathcal{K} \models q[\boldsymbol{a}]$ iff $\mathcal{U}_\mathcal{K} \models q^\dagger[\boldsymbol{a}]$.*

We now discuss the construction of $q^\dagger$ in detail. The rewriting is iterative in the following sense: a single step rewriting obtains a formula of the form $q^\dagger$, but for which Condition 4 may not yet be satisfied. Hence we reapply the rewriting to the disjuncts of the obtained formula until Condition 4 is satisfied. For this reason, we assume that the input query $q = \exists \boldsymbol{u}\, \varphi$ to the rewriting may contain atoms $\neg\mathsf{aux}(v)$. Let $\mathcal{S}(q)$ be the set of bad spikes S in cycles in q and let $\mathit{\Gamma} = \{v(S) \mid S \in \mathcal{S}(q)\}$. For every subset U of $\mathit{\Gamma}$, define an equivalence relation $\sim_U$ on $\mathsf{term}(q)$ as the reflexive and transitive closure of

$$\{(t_0, t_1) \mid (t_0, R, t, R', t_1) \in \mathcal{S}(q),\ t \in U\}.$$

Intuitively, U is a 'guess' identifying those variables in $\mathit{\Gamma}$ that are mapped to elements of $\mathsf{aux}^{\mathcal{U}_\mathcal{K}}$, whereas all variables in $\mathit{\Gamma} \setminus U$ are mapped to elements of $\mathit{\Delta}^{\mathcal{I}_\mathcal{K}}$ that are identified by individual names. Having this guess allows us to eliminate bad spikes by (i) adding atoms $\neg\mathsf{aux}(v)$ for all $v \in \mathit{\Gamma} \setminus U$ and (ii) identifying all terms t, t' with $t \sim_U t'$ as these have to be mapped to the same element in matches of q in $\mathcal{U}_\mathcal{K}$ by property **(p2)** from Section 3.

$$q^U \;=\; \exists \boldsymbol{u}\, \big(\varphi^U \wedge \bigwedge_{t \in U} \mathsf{aux}(t_\xi) \wedge \bigwedge_{t \in (\mathit{\Gamma} \setminus U)} \neg\mathsf{aux}(t_\xi)\big),$$

where t_ξ is a representative of the $\sim_U$-equivalence class ξ of t and φ^U results from φ by replacing every t in it with t_ξ.

Lemma 1. *For every k-ary CQ q, every KB $\mathcal{K}$, and every k-tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{U}_\mathcal{K} \models q[\boldsymbol{a}]$ iff $\mathcal{U}_\mathcal{K} \models \bigvee_{U \subseteq \mathit{\Gamma}} q^U[\boldsymbol{a}]$.*

Clearly, all bad spikes in q are eliminated during the construction of q^U. However, as mentioned above, this elimination can introduce new bad spikes. So we apply the rewriting to the disjuncts of q^U and repeat this process until we end up with a query satisfying Condition 4 above. This procedure must terminate since at least two terms are identified at each step. The outcome is the desired rewriting $q^\dagger$ of q. It is not hard to see that it is of size $2^{\mathcal{O}(|q|)}$.

5 Experiments

We evaluate the performance of our combined FO rewriting technique by comparing it (i) with unrewritten queries over the original ABox, which does not yield correct results but provides an 'ultimate lower bound' in the sense that queries cannot be expected to be executed any faster; and (ii) with the non-combined FO query rewriting introduced in [8, 9] and implemented in the QuOnto system [1, 21].

The experiments employ two *DL-Lite* ontologies formulated in *DL-Lite*$_{core}$, the common fragment of *DL-Lite*$^{\mathcal{N}}_{horn}$ and the logic underlying QuOnto. The first ontology, called *Galen-Lite*, is the *DL-Lite*$_{core}$ approximation of the well-known medical ontology *Galen*: it consists of exactly those *DL-Lite*$_{core}$ concept inclusions (in the vocabulary of Galen) that are logical consequences of Galen. Galen-Lite contains 2734 concept names, 207 roles, and 4890 axioms. The second ontology is called *Core*; it is a *DL-Lite*$_{core}$ representation of (a fragment of) a supply-chain management system used by the bookstore chain Ottakar's, now rebranded as Waterstone's. Core contains 84 concept names, 77 roles, and 381 axioms. Galen-Lite defines a complex concept hierarchy, but there are only few axioms that involve roles. This leads to canonical interpretations whose anonymous (i.e., non-ABox) part is relatively small. Core, in contrast, comprises more interesting statements about roles, which gives a richer anonymous part in canonical interpretations. Core's concept hierarchy, however, is not as sophisticated as the one of Galen-Lite. Both ontologies can be found at `http://www.dcs.bbk.ac.uk/~roman/query-rewriting/`.

In the experiments for the combined FO rewriting technique, we use the following representation of $\mathcal{I}_{\mathcal{K}}$. As relational database systems are not optimized to handle a large number of relatively small relations, one for each concept and role name in the ontology, only two relations have been used to represent $\mathcal{I}_{\mathcal{K}}$:

`acbox(conceptid,indid)` and `arbox(roleid,domain-indid,range-indid)`,

where `conceptid` and `roleid` are numerical identifiers for concept names and roles names, `indid`, `domain-indid`, and `range-indid` are numerical identifiers for individuals, `acbox` represents concept memberships, and `arbox` role memberships. B+tree indices were created on the attribute pairs `{conceptid,indid}`, `{roleid,domain-indid}` and `{roleid,range-indid}`. We distinguish ABox individuals from anonymous ones by positive and negative ids, and thus need not store `aux` as a relation. As an example for this representation, take the query '$q = A(x) \wedge \neg \mathsf{aux}(x)$' which, with respect to the Core ontology (that merely provides the concept identifier for A), translates to the SQL statement

```
select indid from acbox where conceptid=31 and indid > 0.
```

Data sets (ABox instances) for our experiments were generated randomly. The data was stored and the test queries were executed on PostgreSQL version 8.3.7 running on a SUN Fire-280R server with two UltraSPARC III 1.2GHz CPUs, 4GB memory and 1TB storage under Solaris 5.10.

Figures 1 and 2 summarize the running times for each of the test queries and varying ABox size. For each ABox, we report the number of individuals (Ind, in thousands), the numbers of concept assertions (CAs, in millions) and role assertions (RAs, in millions) in the original ABox and in the rewritten ABox (i.e, the canonical interpretation). For each query, we show the following execution times in the columns UN, RW and QO:

	ABox size (in M)				query											
Ind	original		canonical		*coreQ1*			*coreQ2*			*coreQ3*			*coreQ4*		
(in K)	CA	RA	CA	RA	UN	RW	QO	UN	RW	QO	UN	RW	QO	UN	RW	QO
100	0.5	0.5	1	1.8	0.4	0.8	>2h	1.5	4.5	92.8	1.4	6.0	>2h	1.4	1.8	2.6
100	1	1	1.8	3.2	0.6	2.3	>2h	1.7	7.6	241.1	1.5	11.4	>2h	3.1	3.4	3.1
100	5	5	5.8	10	8.1	16.5	>2h	9.7	27.6	2048.2	9.4	33.3	>2h	18.8	20.6	18.6
500	0.5	0.5	1.5	2.3	0.2	0.5	>2h	2.2	7.9	113.9	0.4	3.4	>2h	2.3	1.5	2.3
500	1	1	2.8	4.3	0.6	1.7	>2h	3.2	13.6	271.0	1.2	12.8	>2h	2.9	2.7	2.8
500	5	5	9	15.9	4.0	6.6	>2h	9.7	48.8	1921.3	7.9	69.4	>2h	20.6	20.5	19.7
500	10	10	15	27.7	5.8	11.2	>2h	26.9	93.5	5093.2	23.2	125.1	>2h	47.5	48.1	44.1

Fig. 1. Query processing time (in seconds) for the Core ontology.

	ABox size (in M)				query											
Ind	original		canonical		*galenQ1*			*galenQ2*			*galenQ3*			*galenQ4*		
(in K)	CA	RA	CA	RA	UN	RW	QO	UN	RW	QO	UN	RW	QO	UN	RW	QO
5	0.5	0.5	1.6	1.2	0.3	0.4	0.5	0.1	2.2	6.1	0.2	0.9	97.9	1.6	3.1	1.5
5	1	1	2.4	1.9	0.7	0.6	1.0	1.1	3.5	9.9	0.5	1.5	133.9	1.6	5.3	1.6
5	5	5	7.2	6.2	6.2	6.6	15.4	9.5	7.5	226.2	5.1	2.6	2330.8	22.0	27.1	22.6
50	0.5	0.5	3.4	3.8	0.1	0.3	0.4	0.1	1.5	6.7	0.1	0.6	136.9	1.3	8.1	1.3
50	1	1	5.5	4.8	0.2	0.7	0.5	0.2	3.5	9.0	0.2	1.3	152.8	2.6	14.9	1.6
50	5	5	15.5	12.0	1.1	1.6	2.5	1.2	15.1	40.7	16.3	6.2	3397.3	8.8	40.0	9.2
50	10	10	24.6	18.7	2.1	2.4	5.7	1.5	21.8	117.3	2.7	16.9	1318.6	23.8	72.0	19.9

Fig. 2. Query processing time (in seconds) for the Galen-Lite ontology.

UN: the unmodified query over the original ABox;
RW: the query rewritten as in Section 4 over the canonical interpretation from Section 3;
QO: the query produced by QuOnto (over the original ABox).

The queries referred to in the figure are as follows, where the symbols A, B, R, etc., stand for concept and role names in Core and Galen-Lite:

$$\begin{aligned}
\mathit{coreQ1}(x) &= \exists yz\, A(x), B(y), R(x,y), R(x,z), C(z)\\
\mathit{coreQ2}(x) &= \exists yz\, R(x,y), S(x,z)\\
\mathit{coreQ3}(x) &= \exists yz\, A(x), R(x,y), S(x,z)\\
\mathit{coreQ4}(x) &= \exists xyzw\, R(x,y), R(z,y), R(x,w), R(z,w)\\
\mathit{galenQ1}(x) &= \exists y\, A(x), R(x,y), B(y)\\
\mathit{galenQ2}(x) &= \exists y\, A'(x), R'(x,y), B'(y)\\
\mathit{galenQ3}(x) &= \exists yz\, A(x), B(y), R(x,y), R(x,z), C(z)\\
\mathit{galenQ4}(x) &= \exists xyzw\, R(x,y), R(z,y), R(x,w), R(z,w)
\end{aligned}$$

Thus, the only difference between, say, *coreQ1* and *galenQ3* is that the abstract symbols have been replaced by distinct concept and role names. Note that *coreQ4* and *galenQ4* contain bad spikes, whereas all other queries are free of them. These queries show only a sample of the queries that we have tested, but the results are representative.

To evaluate the results, we first compare the query processing times for the unmodified query over the original ABox (UN) with the times for our combined FO rewriting technique (RW). The figures show that RW is almost always less than 10 times longer than UN, in many case much less than 5 times longer. In terms of absolute numbers, RW queries are all answered within one minute, and usually even within a few seconds (just like UN queries). This applies even to the queries *coreQ4* and *galenQ4*, for which an exponential rewriting is required. Finally, we observe that the performance of the combined FO rewriting approach is similar for Core and Galen-Lite, and thus does not seem to strongly depend on the structure of the TBox. We find these results extremely encouraging: while our approach results in a considerable increase in the ABox size and rewriting the ABox may take some time (up to an hour using standard RDBMSs, which are not optimized for this task), the actual query execution is usually not significantly slower than the 'ultimate lower bound'.

We now compare UN/RW with the QuOnto approach, which results in query execution times that are much less favorable for both Core and Galen. To explain this discrepancy, recall that the RW approach rewrites the query independently of the TBox, whereas QuOnto rewriting *does* depend on the TBox. To see a basic example of this dependence, consider $q(x) = A(x)$ and $\mathcal{T} = \{A_1 \sqsubseteq A, A_2 \sqsubseteq A\}$, where the QuOnto rewriting is $q'(x) = A(x) \vee A_1(x) \vee A_2(x)$, i.e., there is one disjunct for each subsumee of the concept A in the query. Galen-Lite is a TBox in which many concepts have a lot of subsumees. For example, the concepts A and B in *galenQ2* have 39 and 11 (direct and indirect) subsumees, which results in 429 subqueries; *galenQ3* even generates 3072 subqueries.

Other sources of query expansion in QuOnto stem from mandatory role participation assertions and the possibility of query folding due to multiple occurrences of the same role. Like many ontologies originating from ER design, the Core TBox contains such constraints. For the queries *coreQ1*, *coreQ2*, and *coreQ3*, this results in 180, 310, and 2100 disjuncts in the rewritten query, respectively. We note that the weak performance of QuOnto on *coreQ1* and *coreQ3* over the Core TBox can, in part, be traced to poor query optimizer choices. On queries with very many disjuncts, the probability of the optimizer choosing a bad strategy for at least one of the disjuncts is rather high, and the resulting execution time will dominate the execution time of the overall query.

6 Conclusion

We introduce a novel approach to CQ answering over *DL-Lite*$^{\mathcal{N}}_{horn}$ TBoxes using relational database systems. Our experimental evaluation suggests that, in applications where the user has authority over the data and can implement ABox rewriting, this approach may be preferable to the TBox-based rewriting [8, 9]. The work presented in this paper is only a first step: future work includes an adaptation of our approach to variants of *DL-Lite* with role hierarchies, extending the class of CQs for which a polynomial rewriting is possible, finding a more careful exponential rewriting step, and developing incremental approaches to ABox rewriting. It might also be interesting to combine the ideas underlying query rewriting in QuOnto with ABox rewriting to improve query execution times in the QuOnto approach.

References

1. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: QUerying ONTOlogies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. *DL-Lite* in the light of first-order logic. In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 361–366, 2007.
3. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. Technical Report BBKCS-09-03, SCSIS, Birkbeck College, London, 2009 (available at `http://www.dcs.bbk.ac.uk/research/techreps/2009/bbkcs-09-03.pdf`).
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
5. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS'98*, pages 149–158, New York, NY, USA, 1998. ACM.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and M. Ruzzi. Data integration through *DL-Lite*$_{\mathcal{A}}$ ontologies. In K.-D. Schewe and B. Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 2008.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.
9. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
10. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in P2P data integration: An epistemic logic approach. *Information Systems*, 33(4):360–384, 2008.
11. T. Eiter, C. Lutz, M. Ortiz, and M. Šimkus. Query answering in description logics with transitive roles. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence IJCAI09*. AAAI Press, 2009.
12. B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 399–404, 2007.
13. B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*. CEUR Workshop Proceedings, 2006.
14. I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *In AAAI/IAAI*, pages 399–404, 2000.
15. C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.

16. C. Lutz, R. Kontchakov, D. Toman, F. Wolter, and M. Zakharyaschev. Combined FO rewritability for conjunctive query answering in *DL-Lite*. Available at `http://www.informatik.uni-bremen.de/~clu/`, 2009.
17. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in $\mathcal{EL}$ using a database system. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008.
18. C. Lutz, D. Toman, and F. Wolter. Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ using a Relational Database System. In *Proc. of the 21th Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, page in press, 2009.
19. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *J. of Automated Reasoning*, 41(1):61–98, 2008.
20. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
21. A. Poggi, M. Rodriguez, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC)*, 2008.
22. H. Perez-Urbina, B. Motik, and I. Horrocks. Rewriting Conjunctive Queries over Description Logic Knowledge Bases. In *Proc. of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, number 4925 in LNCS, pages 199–214, Springer, 2008.