The Combined Approach to Ontology-Based Data Access

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

[1] Department 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@cs.uwaterloo.ca

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

Abstract

The use of ontologies for accessing data is one of the most exciting new applications of description logics in databases and other information systems. A realistic way of realising sufficiently scalable ontology-based data access in practice is by reduction to querying relational databases. In this paper, we describe the *combined approach*, which incorporates the information given by the ontology into the data and employs query rewriting to eliminate spurious answers. We illustrate this approach for ontologies given in the *DL-Lite* family of description logics and briefly discuss the results obtained for the $\mathcal{EL}$ family.

1 Ontology-Based Data Access

The paradigm of ontology-based data access (OBDA) has recently emerged as an exciting application of knowledge representation and reasoning technologies in information management systems [Dolby *et al.*, 2008; Heymans *et al.*, 2008; Poggi *et al.*, 2008a]. In a nutshell, the underlying idea is to facilitate access to data by separating the user from the raw data sources using an ontology that provides a user-oriented view of the data and makes it accessible via queries formulated solely in the language of the ontology without any knowledge of the actual structure of the data.

To make this idea more precise, let us assume that the ontology, $\mathcal{T}$, is given by a finite set of sentences of (a suitable fragment of) first-order logic (FO) and the data $\mathcal{D}$ by a finite set of ground atoms $P(a_1, \dots, a_n)$ of FO, where the a_i are individual names (constants) and P is an n-ary predicate symbol. A query $q(\vec{x})$ is an FO-formula with free variables $\vec{x}$, called the *answer variables*. At its core, the OBDA scenario is typical of the logic-based approach to knowledge representation, where logical theories are employed to represent knowledge, and reasoning is required to unlock that knowledge for applications. In OBDA, the ontology $\mathcal{T}$ is typically used to enrich the data with additional vocabulary for querying, to translate between different data and query vocabularies, and for reconciling the different vocabularies of multiple data sources. The data $\mathcal{D}$ is assumed to be *incomplete* to allow for the inference of additional data by means of reasoning. This requires, at least in principle, reasoning over a potentially infinite set of possible models of $\mathcal{T}$ and $\mathcal{D}$. Thus, the fundamental *query-answering problem* we are facing is to decide whether a tuple $\vec{a}$ of individual names from $\mathcal{D}$ is a *certain answer* to q over $\mathcal{T}$ and $\mathcal{D}$—i.e., whether $q(\vec{a})$ is true in every FO-model $\mathcal{M}$ of $\mathcal{T}$ and $\mathcal{D}$. In contrast, the relational database paradigm presupposes that the data is complete, so answering a query means checking whether it holds in the *single* model given by $\mathcal{D}$.

As a simple illustration of OBDA, consider the query

$$\varphi(x) = \exists y, z\, (\mathit{city}(x) \land \mathit{has\_airport}(x, y) \land \mathit{located\_in}(x, \mathit{US}) \land \mathit{named\_for}(y, z) \land \mathit{ww2\_hero}(z)),$$

asking for US cities with an airport named after a WW2 hero. Let us assume that we have a database $\mathcal{DB}$ with tables for all the relations mentioned in $\varphi(x)$, except the more abstract concept $\mathit{ww2\_hero}$, and with additional tables for $\mathit{ww2\_deco}$ (WW2 decoration) and $\mathit{recipient\_of}$. Thus, $\mathcal{DB}$ contains atoms such as

$$\mathit{city}(\mathit{Chicago}), \quad \mathit{has\_airport}(\mathit{Chicago}, \mathit{ORD}),$$
$$\mathit{located\_in}(\mathit{Chicago}, \mathit{US}), \quad \mathit{named\_for}(\mathit{ORD}, \mathit{O'Hare}),$$
$$\mathit{recipient\_of}(\mathit{O'Hare}, \mathit{ww2\_medal\_of\_honor}),$$
$$\mathit{ww2\_deco}(\mathit{ww2\_medal\_of\_honor}).$$

As $\mathcal{DB}$ contains no data for the relation $\mathit{ww2\_hero}$, no answer to $\varphi(x)$ over $\mathcal{DB}$ can be found. However, if we describe WW2 heros by means of an ontology $\mathcal{H}$ with sentences such as

$$\forall x, y\, (\mathit{recipient\_of}(x, y) \land \mathit{ww2\_deco}(y) \to \mathit{ww2\_hero}(x)),$$

$\mathit{Chicago}$ becomes an answer to $\varphi(x)$ over $\mathcal{H}$ and $\mathcal{DB}$.

To be useful in practice, OBDA should scale to large amounts of data and preferably be as efficient as standard relational database management systems (RDBMSs), where decades of research have been invested to make them scalable. Realistically, this means that we are interested in such ontology and query languages for which OBDA is *efficiently reducible* to tasks that can be executed using existing RDBMSs. Thus, given $\mathcal{T}$, $\mathcal{D}$ and $q(\vec{x})$ as above, we want to compute a finite FO model $\mathcal{D}'$ and an FO query $q'(\vec{x})$ such that

(ans) $\vec{a}$ is an answer to $q'(\vec{x})$ over $\mathcal{D}'$ if, and only if, $\vec{a}$ is a certain answer to $q(\vec{x})$ over $\mathcal{T}$ and $\mathcal{D}$.

To illustrate the type of reduction we have in mind, observe that querying $(\mathcal{H}, \mathcal{DB})$ with $\varphi(x)$ can be reduced to asking $\varphi(x)$ over the extension $\mathcal{DB}'$ of $\mathcal{DB}$ with a table for $ww2\_hero$ containing all names in $\mathcal{DB}$ who are recipients of a WW2 decoration (which can be easily computed). Another possible reduction is to use the same database $\mathcal{DB}$ but rewrite the query $\varphi(x)$ to a new query $\varphi'(x)$, which results from $\varphi(x)$ by replacing the conjunct $ww2\_hero(z)$ with

$$ww2\_hero(z) \lor \exists v\,(recipient\_of(z, v) \land ww2\_deco(v)).$$

As the size of data is normally large and many different queries can be posed to the same database, at least the following two requirements should supplement **(ans)**:

(dat) $\mathcal{D}'$ is computable in polynomial time in $\mathcal{D}$ and does not depend on $q(\vec{x})$;

(que) $q'(\vec{x})$ does not depend on $\mathcal{D}$.

There are various possible refinements of these conditions. The *query-rewriting* approach of [Calvanese *et al.*, 2007] does not allow modifications of the data, so that **(dat)** is replaced with $\mathcal{D}' = \mathcal{D}$. As a result, this approach is only applicable to description logics for which query-answering belongs to the class AC^0 for *data complexity* (that is, if only the data is regarded as input, whereas the ontology and the query are regarded as fixed). Although guaranteeing the same data complexity as in RDBMSs, the rewriting approach does not impose any restrictions on the size of the 'rewritten' queries $q'(\vec{x})$, which may be exponential in the size of q and so prohibitive for efficient execution by RDBMSs.

In this paper, we suggest a different refinement of conditions **(dat)** and **(que)** by taking account of the size of $\mathcal{T}$:

(dat′) $\mathcal{D}'$ is computable in polynomial time in both $\mathcal{T}$ and $\mathcal{D}$, preferably using RDBMSs;

(que′) $q'(\vec{x})$ is polynomial in $\mathcal{T}$ and $q(\vec{x})$.

These conditions emerge from the *combined approach* to OBDA suggested in [Lutz *et al.*, 2009; Kontchakov *et al.*, 2010] and aim at scenarios where it is allowed to manipulate the source data (which is not always the case in information integration). The motivation for this approach is twofold. First, by allowing $\mathcal{D}' \neq \mathcal{D}$, we gain the advantage of much smaller and transparent rewritings $q'(\vec{x})$ of the query. The experimental data we discuss below indicates that this leads to significant performance improvements. Second, the approach advocated here is not confined to the languages for which OBDA is in AC^0 for data complexity. As shown in [Lutz *et al.*, 2009], it can be successfully applied to ontology languages for which query-answering is PTIME-complete for data complexity.

Over many years now, the most popular and successful ontology languages are based on description logics. In fact, the *DL-Lite* family of description logics [Calvanese *et al.*, 2007; Artale *et al.*, 2009] was specifically designed for the rewriting approach to OBDA and is the logical underpinning of the profile *OWL 2 QL* of the *OWL 2* Web Ontology Language. In our exposition of the combined approach below, we focus on a simple member of the *DL-Lite* family, but also briefly discuss its applicability to the $\mathcal{EL}$ family of description logics, which underlies the *OWL 2 EL* profile of *OWL 2*.

2 *DL-Lite*$_{horn}$

To discuss the main ideas behind the combined approach, we consider the description logic *DL-Lite*$_{horn}$ [Artale *et al.*, 2009] designed to represent relationships between concepts (unary predicates in FO or classes in *OWL*) and the domains and ranges of roles (binary relations in FO or properties in *OWL*). Ontologies in *DL-Lite*$_{horn}$, as well as most other description logics, are called *TBoxes* (T for terminology) and consist of inclusions between concepts. The expressive power of the logic depends then on the constructors available to build concepts. In the case of *DL-Lite*$_{horn}$, *roles* R and *concepts* C are built from concept names A_i and role names P_i, $i \geq 0$, according to the following syntax rules:

$$R ::= P_i \mid P_i^-, \qquad C ::= \top \mid \bot \mid A_i \mid \exists R,$$

and a *DL-Lite*$_{horn}$ TBox $\mathcal{T}$ is a finite set of *concept inclusions* (*CIs*) $C_1 \sqcap \dots \sqcap C_n \sqsubseteq C$, where the C_i and C are concepts. Every CI can be regarded as a first-order Horn sentence. For example, $\exists P \sqcap \exists P^- \sqsubseteq A$ has exactly the same meaning as $\forall x\,(\exists y\,P(x, y) \land \exists y\,P(y, x) \to A(x))$.

Thus, TBoxes are interpreted in standard FO structures $\mathcal{I} = (\Delta^{\mathcal{I}}, A_i^{\mathcal{I}}, P_i^{\mathcal{I}})_{i\geq 0}$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain*, $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We set $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$, $(P_i^-)^{\mathcal{I}} = \{(x, y) \mid (y, x) \in P_i^{\mathcal{I}}\}$, and

$$(\exists R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{there is } y \in \Delta^{\mathcal{I}} \text{ with } (x, y) \in R^{\mathcal{I}}\};$$

so $\exists P$ is interpreted as the domain of P and $\exists P^-$ as its range. We write $\mathcal{I} \models \bigsqcap_{i=1}^n C_i \sqsubseteq C$ and say that $\bigsqcap_{i=1}^n C_i \sqsubseteq C$ *is satisfied in* $\mathcal{I}$ if $\bigcap_{i=1}^n C_i^{\mathcal{I}} \subseteq C^{\mathcal{I}}$.

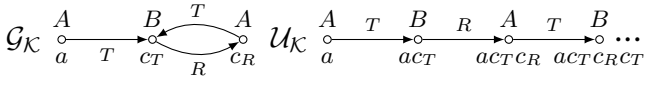
In description logic, ground atoms of the form $A(a)$ and $P(a, b)$, where A is concept name, P a role name, and a, b individual names, are called *concept assertions* and *role assertions*, respectively. An *ABox*, $\mathcal{A}$, is a finite set of concept and role assertions, which is used to store instance data. In interpretations $\mathcal{I}$, $a^{\mathcal{I}}$ is the domain element interpreting the individual name a in $\mathcal{I}$. As usual, $\mathcal{I} \models A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and $\mathcal{I} \models P(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$.

We denote by $\mathsf{Ind}(\mathcal{A})$ the set of individual names occurring in $\mathcal{A}$, and often write $P^-(a, b) \in \mathcal{A}$ instead of $P(b, a) \in \mathcal{A}$. A *DL-Lite*$_{horn}$ *knowledge base* (*KB*) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. We write $\mathcal{K} \models \alpha$ whenever $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $\mathcal{K}$. $\mathcal{K}$ is *consistent* if it has a model. Consistency of *DL-Lite*$_{horn}$ KBs is known to be PTIME-complete [Artale *et al.*, 2009] for combined complexity.

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

$$\mathcal{T} = \{A \sqsubseteq \exists T,\ \exists T^- \sqsubseteq B,\ B \sqsubseteq \exists R,\ \exists R^- \sqsubseteq A\}.$$

Two models of $\mathcal{K}$, called $\mathcal{G}_{\mathcal{K}}$ and $\mathcal{U}_{\mathcal{K}}$, are depicted below:



First-order (*FO*) *queries* $q(\vec{x})$ to a KB $\mathcal{K}$ are constructed from concept and role names (treated as unary and binary predicates) using individual names and variables (called, as usual, *terms*). We say that a tuple $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$ (of the same length as $\vec{x}$) is an *answer to* $q(\vec{x})$ *in an interpretation* $\mathcal{I}$ if

$\mathcal{I} \models q[\vec{a}]$, where $q[\vec{a}]$ results from replacing the (free) variables $\vec{x}$ in $q(\vec{x})$ with constants $\vec{a}$. The set of all answers to q in $\mathcal{I}$ is denoted by $\mathsf{ans}(q, \mathcal{I})$. A tuple $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is a *certain answer to* $q(\vec{x})$ *over* $\mathcal{K}$ if $\mathcal{I} \models q[\vec{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$. The set of all certain answers to q over $\mathcal{K}$ is denoted by $\mathsf{cert}(q, \mathcal{K})$.

Example 2. Let $q(x) = \exists y, z\,(T(x, y) \land R(y, z) \land T(z, y))$ and $\mathcal{K}$ be as in Example 1. Then a is an answer to $q(x)$ in $\mathcal{G}_\mathcal{K}$, but not a certain answer to $q(x)$ over $\mathcal{K}$ as $\mathcal{U}_\mathcal{K} \not\models q[a]$.

Conditions **(ans)** and **(dat)** cannot be satisfied for arbitrary FO queries—simply because FO is undecidable. Although such queries are allowed in standard database languages such as SQL, most commonly used in practice are *conjunctive queries* (*CQ*s) of the form $q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$, where φ is constructed from atomic predicates using only conjunction (in databases, CQs are known as SPJ queries). In more general *positive existential queries*, disjunction is also allowed in φ.

Thus, the problem we are facing now is how to compute, ideally in polynomial time, given a *DL-Lite*$_{horn}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a CQ $q(\vec{x})$, a finite FO-structure $\mathcal{G}_\mathcal{K}$, independently from $q(\vec{x})$, and an FO-query $q'(\vec{x})$, independently from $\mathcal{A}$, such that **(ans)** holds: for every tuple $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$ if, and only if, $\vec{a} \in \mathsf{ans}(q', \mathcal{G}_\mathcal{K})$. The key to a solution is the well-known property of Horn theories: the existence of *least Herbrand*, or *canonical*, *models* which give all correct answers to CQs [Apt, 1990]. (We note here that all ontology languages used for OBDA so far are sub-languages of the *Horn fragment* of FO, which is not able to express any kind of disjunctive information.)

3 Canonical Models

To construct the *canonical model*, $\mathcal{U}_\mathcal{K}$, for a *DL-Lite*$_{horn}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we start with $\mathcal{A}$ and then exhaustively apply the CIs from $\mathcal{T}$, always introducing *new* elements in role domain and ranges if necessary (as in Example 1). Formally, the domain of $\mathcal{U}_\mathcal{K}$ consists of *paths* of the form $a c_{R_1} \cdots c_{R_n}$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$ and the following conditions hold:

(agen) $\mathcal{K} \models \exists R_1(a)$ but $R_1(a, b) \notin \mathcal{A}$ for all $b \in \mathsf{Ind}(\mathcal{A})$, in which case we write $a \rightsquigarrow c_{R_1}$;

(rgen) for $i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$, in which case we write $c_{R_i} \rightsquigarrow c_{R_{i+1}}$.

We denote the last element in a path σ by $\mathsf{tail}(\sigma)$, and define $\mathcal{U}_\mathcal{K}$ by taking:

$$
\begin{aligned}
\Delta^{\mathcal{U}_\mathcal{K}} &= \{a c_{R_1} \cdots c_{R_n} \mid a \in \mathsf{Ind}(\mathcal{A}), a \rightsquigarrow c_{R_1} \rightsquigarrow \cdots \rightsquigarrow c_{R_n}\},\\
a^{\mathcal{U}_\mathcal{K}} &= a, \text{ for } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{U}_\mathcal{K}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup\\
&\quad \{\sigma c_R \in \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\},\\
P^{\mathcal{U}_\mathcal{K}} &= \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid P(a, b) \in \mathcal{A}\} \cup\\
&\quad \{(\sigma, \sigma c_P) \in \Delta^{\mathcal{U}_\mathcal{K}} \times \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathsf{tail}(\sigma) \rightsquigarrow c_P\} \cup\\
&\quad \{(\sigma c_{P^-}, \sigma) \in \Delta^{\mathcal{U}_\mathcal{K}} \times \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathsf{tail}(\sigma) \rightsquigarrow c_{P^-}\}.
\end{aligned}
$$

The following theorem, reflecting the fact that any model of $\mathcal{K}$ contains a homomorphic image of $\mathcal{U}_\mathcal{K}$, provides a basis for reductions of OBDA to query-answering in RDBMSs:

Theorem 3. *For every consistent DL-Lite*$_{horn}$ *KB* $\mathcal{K}$ *and every positive existential query* q, $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{U}_\mathcal{K})$.

The consistency check $\mathcal{K} \models \bot$ required in this theorem can be easily encoded as an FO-query over $\mathcal{K}$. So the real problem here is how to cope with the fact that $\mathcal{U}_\mathcal{K}$ can be large, even infinite.

Note first that, although $\mathcal{U}_\mathcal{K}$ can be infinite, its structure is quite regular: we have a 'kernel' given by the ABox $\mathcal{A}$, each element a of which is a root of a tree with branches of the form $a \rightsquigarrow c_{R_1} \rightsquigarrow \cdots \rightsquigarrow c_{R_n}$, where the c_{R_i} represent 'witnesses' for the existential restrictions $\exists R_i$. The concepts from $\mathcal{T}$ to which the elements of $\mathcal{U}_\mathcal{K}$ belong are computed (in polynomial time) by applying the 'rules' in $\mathcal{T}$: for instance, $a c_{R_1} \cdots c_{R_n} \in A^{\mathcal{U}_\mathcal{K}}$ if, and only if, $\mathcal{T} \models \exists R_n^- \sqsubseteq A$.

This suggests two ways of encoding $\mathcal{U}_\mathcal{K}$. The rewriting approach of [Calvanese *et al.*, 2007], illustrated by the query $\varphi'(x)$ from the introduction, 'builds' $\mathcal{T}$ into the rewritten query, independently of $\mathcal{A}$. We give another example demonstrating the underlying idea:

Example 4. Take again the TBox $\mathcal{T}$ from Example 1 and let $q(x) = \exists y\, R(x, y)$. An element $a \in \mathsf{Ind}(\mathcal{A})$ is an answer to $q(x)$ in $\mathcal{U}_\mathcal{K}$ if either $R(a, y)$ is in $\mathcal{A}$, for some y, or $R(a, y)$ is entailed by $\mathcal{A}$ and $\mathcal{T}$. By inspecting $\mathcal{T}$, we see that $R(a, y)$ can only be entailed by either $B(a)$ or $T(z, a)$, for some z. Thus, a is an answer to $q(x)$ over $(\mathcal{T}, \mathcal{A})$ if, and only if, a is an answer to $q(x) \lor B(x) \lor \exists z\, T(z, x)$ over $\mathcal{A}$.

This rewriting technique reduces the OBDA problem '$(\mathcal{T}, \mathcal{A}) \models q(\vec{x})$?' for *DL-Lite*$_{horn}$ KBs to the RDBMS problem '$\mathcal{A} \models q'(\vec{x})$?' where q' depends on q and $\mathcal{T}$ only. Thus, it complies with conditions **(ans)**, **(dat)** and **(que)** and shows that answering positive existential queries over *DL-Lite*$_{horn}$ KBs is in AC^0 for data complexity [Calvanese *et al.*, 2007], the same as RDBMS query answering. However, the size of $q'(\vec{x})$ in all reductions known so far is exponential in $|q|$, $\mathcal{O}((|\mathcal{T}| \cdot |q|)^{|q|})$ in the worst case, and even modern RDBMSs struggle with such queries.

The combined approach of [Kontchakov *et al.*, 2010] encodes $\mathcal{U}_\mathcal{K}$ in a finite FO structure, $\mathcal{G}_\mathcal{K}$, by identifying all paths σ of $\mathcal{U}_\mathcal{K}$ with the same $\mathsf{tail}(\sigma)$—all of them belong to the same concepts by the definition of $\mathcal{U}_\mathcal{K}$—as illustrated in Example 1. Thus, the *generating model* $\mathcal{G}_\mathcal{K}$ of $\mathcal{K}$ is defined as follows:

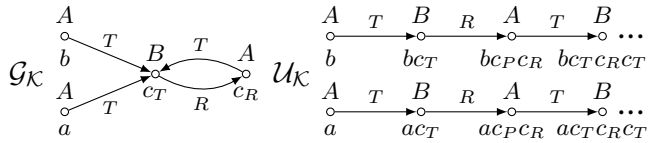
$$
\begin{aligned}
\Delta^{\mathcal{G}_\mathcal{K}} &= \mathsf{Ind}(\mathcal{A}) \cup \{c_R \mid a \in \mathsf{Ind}(\mathcal{A}), a \rightsquigarrow \cdots \rightsquigarrow c_R\},\\
a^{\mathcal{G}_\mathcal{K}} &= a, \text{ for } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{G}_\mathcal{K}} &= \{\mathsf{tail}(\sigma) \mid \sigma \in A^{\mathcal{U}_\mathcal{K}}\},\\
P^{\mathcal{G}_\mathcal{K}} &= \{(\mathsf{tail}(\sigma), \mathsf{tail}(\sigma')) \mid (\sigma, \sigma') \in P^{\mathcal{U}_\mathcal{K}}\}.
\end{aligned}
$$

The generating model $\mathcal{G}_\mathcal{K}$ can be constructed in polynomial time in $|\mathcal{K}|$, thus complying with condition **(dat′)**. In Section 5, we discuss how this can be done using RDBMSs. The canonical model $\mathcal{U}_\mathcal{K}$ can be viewed as the 'unraveling' of $\mathcal{G}_\mathcal{K}$, with the map $\mathsf{tail}\colon \Delta^{\mathcal{U}_\mathcal{K}} \to \Delta^{\mathcal{G}_\mathcal{K}}$ being a homomorphism from $\mathcal{U}_\mathcal{K}$ onto $\mathcal{G}_\mathcal{K}$. As positive existential queries are preserved under homomorphisms, we obtain from Theorem 3:

Theorem 5. *For every consistent DL-Lite*$_{horn}$ *KB* $\mathcal{K}$ *and every positive existential query* q, $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{U}_\mathcal{K}) \subseteq \mathsf{ans}(q, \mathcal{G}_\mathcal{K})$.

As we saw in Example 2, the converse inclusion does not hold because by identifying elements in $\mathcal{U}_\mathcal{K}$ we ruin its tree structure. Here is another example:

Example 6. Suppose that $\mathcal{K} = (\mathcal{T}, \{A(a), A(b)\})$, where $\mathcal{T}$ is as in Example 1. Consider the 'fork-shaped' query $q(x_1, x_2) = \exists y\,(T(x_1, y) \land T(x_2, y))$. Then $\mathcal{G}_\mathcal{K}$ and $\mathcal{U}_\mathcal{K}$ look as depicted below:



So we have $\mathcal{G}_\mathcal{K} \models q_2[a, b]$, while $\mathcal{U}_\mathcal{K} \not\models q_2[a, b]$.

The domain of the generating model $\mathcal{G}_\mathcal{K}$ consists of $\mathsf{Ind}(\mathcal{A})$ and those elements of the 'witness set'

$$\mathsf{R}_\mathcal{T} = \{c_R \mid R \text{ or } R^- \text{ occurs in } \mathcal{T}\}$$

that are accessible from $\mathsf{Ind}(\mathcal{A})$ via the relation $\rightsquigarrow$. In the combined approach, $\mathcal{G}_\mathcal{K}$ is computed in a preprocessing step—independently of any queries—and stored in the database. We shall discuss preprocessing in Section 5. But before that let us see how to get rid of the spurious answers to queries which can be given by $\mathcal{G}_\mathcal{K}$ as in Examples 2 and 6.

4 Conjunctive Query Answering

Suppose we are given a CQ $q(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$. Our aim is to rewrite q to an FO query $q^\dagger(\vec{x})$ in such a way that (*i*) for every *DL-Lite*$_{horn}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q^\dagger, \mathcal{G}_\mathcal{K})$ and (*ii*) the size of $q^\dagger$ is polynomial in the size of q. We define the rewriting $q^\dagger$ as a conjunction

$$q^\dagger(\vec{x}) = \exists \vec{y}\,(\varphi \land \varphi_1 \land \varphi_2 \land \varphi_3),$$

where φ_1, φ_2 and φ_3 are Boolean combinations of equalities $t_1 = t_2$, and each of the t_i is either a term in q or a constant $c_R \in \mathsf{R}_\mathcal{T}$. The purpose of the conjunct

$$\varphi_1 = \bigwedge_{x \in \vec{x}} \bigwedge_{c_R \in \mathsf{R}_\mathcal{T}} (x \neq c_R)$$

is to ensure that the answer variables $\vec{x}$ receive their values from $\mathsf{Ind}(\mathcal{A})$ only, as the witnesses c_R are 'implicit' elements of unknown identity and hidden from the user (labelled nulls, in the database parlance).

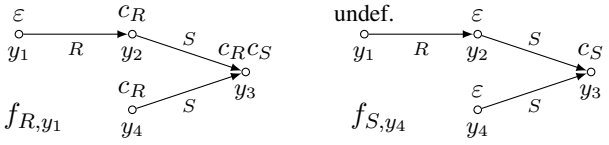
The conjuncts φ_2 and φ_3 are closely related and serve the purpose of filtering out the spurious answers discussed in Examples 2 and 6. The idea is to prevent any assignments that cannot be 'reproduced' in the canonical model $\mathcal{U}_\mathcal{K}$. Due to the forest structure of $\mathcal{U}_\mathcal{K}$, this requirement imposes strong constraints on the way how variables can be matched to the $\mathsf{R}_\mathcal{T}$ part of $\mathcal{G}_\mathcal{K}$. Essentially, the part of q that is mapped to $\mathsf{R}_\mathcal{T}$ must be homomorphically embeddable into a forest. This intuition is captured by the following definition. Let $(\mathsf{R}_\mathcal{T})^*$ be the set of all finite words over $\mathsf{R}_\mathcal{T}$ (including the empty word ε). To simplify notation, we identify q with the set of its atoms and use $P^-(t, t') \in q$ as a synonym of $P(t', t) \in q$.

Definition 7. Let q be a CQ and $R(t, t') \in q$. A partial function f from the terms of q to $(\mathsf{R}_\mathcal{T})^*$ is a *tree witness for* (R, t) in q if its domain is minimal (with respect to set-theoretic inclusion) such that $f(t) = \varepsilon$ and, for all atoms $S(s, s') \in q$,

- if $f(s) = \varepsilon$ then $f(s') = c_R$ provided that $R = S$, and
- if $f(s) = \sigma c_T$ then $f(s') = \begin{cases} \sigma, & \text{if } T = S^-, \\ \sigma c_T c_S, & \text{otherwise.} \end{cases}$

A tree witness for (R, t) in q does not exist if, starting from $R(t, t')$, we can reach some term s in q via two different paths, which would give different values for $f_{R,t}(s)$ (see Example 8). But if one exists then it is unique, and we denote it by $f_{R,t}$. Note that, by **(rgen)**, $\mathcal{G}_\mathcal{K}$ has no paths of the form $\sigma c_{S^-} c_S$, which is reflected by the two cases in the second item of the definition.

Example 8. Let $q = \{R(y_1, y_2), S(y_2, y_3), S(y_4, y_3)\}$. Then the tree witnesses for (R, y_1) and (S, y_4) in q are:



For the cyclic query $q = \{T(x, y), R(y, z), T(z, y)\}$ from Example 2, there exists no tree witness for (R, y) because we must have both $f_{R,y}(y) = \varepsilon$ and $f_{R,y}(y) = c_R c_{T^-}$, contrary to $f_{R,y}$ being a function. Similarly, no tree witnesses exist for (R^-, z), (T, z) or (T^-, y) in q.

Intuitively, the tree witness $f_{R,t}$ addresses those assignments π where $\pi(t)$ is an ABox element but $\pi(t')$, for $R(t, t') \in q$, is not (and so $\pi(t') = c_R$), by recording consequences of the fact that we must be able to reproduce π in $\mathcal{U}_\mathcal{K}$: if $f_{R,t}(s) = \sigma c_S$, then π has no other choice than to map s to c_S, and if $f_{R,t}(s) = \varepsilon$, then π must map s to $\pi(t)$.

The conjunct φ_2 implements the matching dictated by the tree witnesses. It turns out that the matches determined by $f_{R,t}$ also apply when t is *not* mapped to an ABox element. This, in turn, means that it suffices to enforce such matches for the terms s with $f_{R,t}(s) = \varepsilon$:

$$\varphi_2 \quad = \bigwedge_{\substack{R(t,t') \in q \\ \text{tree witness for } (R,t) \text{ exists}}} \big((t' = c_R) \to \bigwedge_{f_{R,t}(s) = \varepsilon} (s = t)\big).$$

Example 9. We illustrate φ_2 using the 'fork-shaped' query $q(x_1, x_2) = \exists y\,(T(x_1, y) \land T(x_2, y))$ from Example 6. As $f_{T,x_1}(x_2) = \varepsilon$, we have $(y = c_T) \to (x_1 = x_2)$ in φ_2, which prevents the spurious match $\{x_1 \mapsto a, y \mapsto c_T, x_2 \mapsto b\}$ of Example 6.

If the tree witness for (R, t) in q does not exist, then there are two paths from $R(t, t')$ to some term s in q. As any element σ of $\mathcal{U}_\mathcal{K}$ may have only one R-successor of the form σc_R, this means that t' in every $R(t, t') \in q$ must be mapped to an ABox element. This is ensured by the conjunct

$$\varphi_3 \quad = \bigwedge_{\substack{R(t,t') \in q \\ \text{no tree witness for } (R,t) \text{ exists}}} \big(t' \neq c_R\big).$$

Example 10. We illustrate φ_3 using the 'cyclic' query $q(x) = \exists y, z\,(T(x, y) \land R(y, z) \land T(z, y))$ from Example 2. As shown in Example 8, there exist no tree witnesses for (R, y) (R^-, z), (T, z) and (T^-, y). This gives four conjuncts $(z \neq c_R)$, $(y \neq c_{R^-})$, $(y \neq c_T)$ and $(z \neq c_{T^-})$ of φ_3, which prevent the spurious matches of Example 2.

Theorem 11. *For every DL-Lite$_{horn}$ KB $\mathcal{K}$ and every CQ q,* $\mathsf{ans}(q^\dagger, \mathcal{G}_\mathcal{K}) = \mathsf{ans}(q, \mathcal{U}_\mathcal{K})$.

Given a CQ $q(\vec{x})$ and $R(t, t') \in q$, it can be decided in time $\mathcal{O}(|q|)$ whether a tree witness for (R, t) in q exists. If it does exist, the tree witness can be computed in time $\mathcal{O}(|q|)$. It follows that $q^\dagger$ can be computed in polynomial time in the size of the query q and the set $\mathsf{R}_\mathcal{T}$. The length of $q^\dagger$ is $\mathcal{O}(|q| \cdot |\mathcal{T}|)$ since φ_1 is of length $\mathcal{O}(|q| \cdot |\mathcal{T}|)$, φ_3 is of length $\mathcal{O}(|q|)$ and φ_2 can be made of length $\mathcal{O}(|q|)$ as well by observing that, for each R, the condition $f_{R,t}(s) = \varepsilon$ induces an equivalence relation on terms. If we add a unary relation aux identifying exactly the elements of $\mathsf{R}_\mathcal{T}$, then we can replace φ_1 with $\varphi_1' = \bigwedge_{x \in \vec{x}} \neg \mathsf{aux}(x)$ and obtain $q^\dagger$ of size $\mathcal{O}(|q|)$.

5 Generating Model by Views

To make the combined approach to OBDA work in practice, it is important to find a feasible way of extending the ABox $\mathcal{A}$ stored in the RDBMS to the generating model $\mathcal{G}_\mathcal{K}$. This is all the more important as we have to reflect the changes to $\mathcal{G}_\mathcal{K}$ whenever facts are added to or deleted from $\mathcal{A}$. Our solution is to use FO-queries to define views in the RDBMS that *compute* the extensions of the concept and role names in $\mathcal{G}_\mathcal{K}$ when executed on $\mathcal{A}$. We illustrate this idea by an example and refer the reader to [Kontchakov *et al.*, 2010] for details.

Example 12. Consider the TBox $\mathcal{T}$ from Example 1. We have to construct views $q_A(x)$, $q_B(x)$, $q_T(x, y)$ and $q_R(x, y)$ the answers to which over $\mathcal{A}$ are $A^{\mathcal{G}_\mathcal{K}}$, $B^{\mathcal{G}_\mathcal{K}}$, $T^{\mathcal{G}_\mathcal{K}}$ and $R^{\mathcal{G}_\mathcal{K}}$, respectively. For example, a first attempt to design $q_B(x)$ might be the following query:

$$q_B'(x) = B(x) \lor \exists y\, T(y, x) \lor (x = c_T).$$

The first two disjuncts reflect the ways concept B can be entailed given $\mathcal{T}$; the last disjunct is meant to reflect the fact that $c_T \in B^{\mathcal{G}_\mathcal{K}}$ when c_T is in the domain of $\mathcal{G}_\mathcal{K}$. However, c_T does not belong to $\mathcal{G}_\mathcal{K}$ if there is no element a of the ABox $\mathcal{A}$ that 'generates' c_T, i.e., $a \rightsquigarrow \cdots \rightsquigarrow c_T$, and thus the answer returned by q_B' may be too large. Fortunately, the condition 'c_T is generated by an element a of the ABox' can be expressed by another FO-query, which in this case is simply $g_T(a) = (A(a) \lor \exists y\, R(y, x)) \land \neg \exists y\, T(a, y)$; cf. $\mathcal{G}_\mathcal{K}$ in Example 1. Thus we can refine the last disjunct of q_B' and obtain the desired query:

$$q_B(x) = B(x) \lor \exists y\, T(y, x) \lor ((x = c_T) \land \exists z\, g_T(z)).$$

The remaining views are constructed analogously.

In general, FO-queries such as $q_A(x)$ and $q_R(x, y)$ in the example above can be of exponential size in $|\mathcal{T}|$. However, they always contain only polynomially many distinct subqueries, and therefore can be represented by means of polynomial non-recursive Datalog programs or a polynomial number of views of an RDBMS.

In *DL-Lite$_{core}$* [Calvanese *et al.*, 2007], which results from *DL-Lite$_{horn}$* by disallowing conjunctions in CIs apart from those in disjointness constraints of the form $A \sqcap B \sqsubseteq \bot$ and roughly corresponds to *OWL 2 QL* without role inclusions, the FO-queries $q_A(x)$ and $q_R(x, y)$ are of linear size even without structure sharing. Since these queries can be 'plugged' into the query $q^\dagger$ defined in Section 4, this yields a new implementation of the rewriting approach to OBDA. Remarkably, for the first time we achieve pure query rewriting with only a polynomial blowup of the query (which also works for the extension of *DL-Lite$_{core}$* with role functionality constraints under the unique name assumption).

6 Experimental Evidence

Pure query rewriting for the extension of *DL-Lite$_{core}$* with role inclusions has been implemented in the systems QuOnto [Acciarri *et al.*, 2005; Poggi *et al.*, 2008b], REQUIEM [Pérez-Urbina *et al.*, 2009], Presto [Rosati and Almatelli, 2010] and Nyaya [Gottlob *et al.*, 2011]. They all rewrite CQs q into *unions* Q of CQs with exponentially many components (in the length of q) in the worst case, and use various query optimisation techniques to reduce the size of Q. The reduction is in some cases quite substantial, but none of the rewritings avoids an exponential blowup in the worst case.

In the combined approach, we separate inference from query processing and construct the generating models, which take care of the ontology. As a result, the rewritings of CQs over *DL-Lite$_{horn}$* ontologies (without role inclusions) are always linear. Moreover, their simple structure ensures that answers to queries can be computed by first evaluating the *original* query in the generating model, and then filtering out (a small number of) tuples violating one of $\varphi_1, \varphi_2, \varphi_3$. This provides true scalability comparable to RDBMSs, which is confirmed by experiments in [Kontchakov *et al.*, 2010] showing that CQ answering with our approach is competitive in performance with executing the original queries over the data. The price we have to pay for this is the construction and maintenance of the generating model $\mathcal{G}_\mathcal{K}$, which, as discussed above, can be implemented by using materialised views and then relying on the RDBMS to maintain them and propagate ABox updates. In the worst case, the size of $\mathcal{G}_\mathcal{K}$ is $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{T}|)$. In practice, the experiments of [Kontchakov *et al.*, 2010] with Galen-Lite indicate that materialization of $\mathcal{G}_\mathcal{K}$ results in a five-fold increase of the size of the data, and the tests of [Calvanese and Rodríguez-Muro, 2011] on Stanford's BioPortal Resource Index show even a tenfold increase. In both cases, the main cause of the increase is the depth of the concept hierarchies in the considered ontologies. Interestingly, [Calvanese and Rodríguez-Muro, 2011] demonstrate that the increase can be curbed, at least for *DL-Lite$_{core}$*, by using an encoding of concept hierarchies as nested intervals (see also [DeHaan *et al.*, 2003]).

7 Extensions and Variations

To simplify presentation, we have only considered a rather small fragment of the languages investigated in [Kontchakov *et al.*, 2010]. First, by adopting the unique name assumption (UNA), the combined polynomial rewriting can be extended to *DL-Lite$_{horn}$* enriched with unqualified number restrictions. (Without the UNA, conjunctive query answering becomes coNP-complete for data complexity [Artale *et al.*, 2009]; note that in the case of *DL-Lite$_{horn}$* considered in this paper, query answers agree with and without UNA.) For the

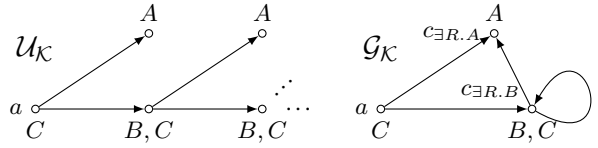
extension of *DL-Lite*$_{horn}$ with role inclusions (i.e., expressions of the form $R_1 \sqsubseteq R_2$), one can polynomially reduce conjunctive query answering to answering *positive existential* queries over TBoxes without role inclusions. At the price of a worst-case exponential blowup, the latter problem can then be reduced to answering unions of conjunctive queries; for details and side conditions, we refer the reader to [Kontchakov *et al.*, 2010]. We note that the blowup encountered in second reduction depends only on the size of the query and on the number of role inclusions, which is typically very small in real-world ontologies.

The combined approach can also be applied to the description logic $\mathcal{EL}$, which is used for large scale medical terminologies such as SNOMED CT. In contrast to *DL-Lite*, in $\mathcal{EL}$ one can qualify existential restrictions and use CIs such as $\exists R.A \sqcap C \sqsubseteq \exists R.B$; however, $\mathcal{EL}$ does not have inverse roles. Conjunctive query answering for $\mathcal{EL}$ ontologies is PTIME-complete for data complexity, and so the rewriting approach is not applicable in this case. In [Lutz *et al.*, 2009], the combined approach is developed for the extension $\mathcal{ELH}_{\bot}^{dr}$ of $\mathcal{EL}$ with the inconsistent concept, role inclusions, and domain and range restrictions. Although the structures of the generating models and rewritten queries are quite different from the *DL-Lite* case, their size is still polynomial in $\mathcal{T}$ and q, respectively. Here we only illustrate the polynomial representation of canonical models for $\mathcal{EL}$ KBs.

Example 13. Let $\mathcal{K} = (\mathcal{T}, \{C(a)\})$, where

$$\mathcal{T} \;=\; \{C \sqsubseteq \exists R.A,\; \exists R.A \sqcap C \sqsubseteq \exists R.B,\; B \sqsubseteq C\}.$$

The canonical model $\mathcal{U}_\mathcal{K}$ and the generating model $\mathcal{G}_\mathcal{K}$ are:



Note that the generating model $\mathcal{G}_\mathcal{K}$ may contain an extra element $c_{\exists R.C}$ for each existential quantifier $\exists R.C$ in the TBox.

Our results for *DL-Lite* and $\mathcal{EL}$ suggest a variety of future research problems. For instance, it would be of great interest to find out how the rewritings used for *DL-Lite* and $\mathcal{EL}$ can be combined so as to cover other Horn descriptions logics such as $\mathcal{ELI}$ or Horn-$\mathcal{SHIQ}$, which use both inverse roles and qualified existential restrictions. In this case, it seems likely that the extension of the data has to be exponential in the size of the input TBox, in the worst case. As noted in [Lutz *et al.*, 2009], DLs for which conjunctive query answering is not in PTIME for data complexity (such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$, unless PTIME = NP) necessarily involve an extension of the data that is exponential in the size of the input *ABox*, which we consider useless for practical purposes. It would also be of interest to investigate the combined approach for more expressive ABoxes and/or queries. First results on how the combined approach can be extended so as to cover ABoxes and queries with 'epistemic' negation are presented in [Lutz *et al.*, 2009]. Another interesting direction is to consider the Datalog$^{\pm}$ family of languages [Gottlob *et al.*, 2011].

Acknowledgments. This work was partially supported by the U.K. EPSRC grants EP/H05099X/1 and EP/H043594/1, by the DFG SFB/TR 8 "Spatial Cognition", and by NSERC.

References

[Acciarri *et al.*, 2005] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies. In *Proc. of AAAI*, pages 1670–1671, 2005.

[Apt, 1990] K. Apt. Logic programming. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 493–574. Elsevier, 1990.

[Artale *et al.*, 2009] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.

[Calvanese and Rodríguez-Muro, 2011] D. Calvanese and M. Rodríguez-Muro, 2011. Private communication.

[Calvanese *et al.*, 2007] 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 Aut. Reason.*, 39(3):385–429, 2007.

[DeHaan *et al.*, 2003] D. DeHaan, D. Toman, M. Consens, and M.T. Özsu. A comprehensive XQuery to SQL translation using dynamic interval encoding. In *Proc. of SIGMOD*, pages 623–634, 2003. ACM.

[Dolby *et al.*, 2008] J. Dolby, A. Fokoue, A. Kalyanpur, Li Ma, E. Schonberg, K. Srinivas, and X. Sun. Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In *Proc. of ISWC*, 2008.

[Gottlob *et al.*, 2011] G. Gottlob, G. Orsi, and A. Pieris. Ontological queries: Rewriting and optimization. In *Proc. of ICDE*, 2011.

[Heymans *et al.*, 2008] S. Heymans, *et al.* Ontology reasoning with large data repositories. In *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*, pages 89–128. Springer, 2008.

[Kontchakov *et al.*, 2010] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of KR*, 2010.

[Lutz *et al.*, 2009] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, pages 2070–2075, 2009.

[Pérez-Urbina *et al.*, 2009] H. Pérez-Urbina, B. Motik, and I. Horrocks. A comparison of query rewriting techniques for *DL-Lite*. In *Proc. of DL*, 2009.

[Poggi *et al.*, 2008a] 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.

[Poggi *et al.*, 2008b] A. Poggi, M. Rodriguez, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In *Proc. of OWLED DC*, 2008.

[Rosati and Almatelli, 2010] R. Rosati and A. Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of KR*, 2010.