Conjunctive Query Answering with *OWL 2 QL*

Stanislav Kikot and **Roman Kontchakov** and **Michael Zakharyaschev**
Department of Computer Science and Information Systems
Birkbeck, University of London, U.K.
{kikot,roman,michael}@dcs.bbk.ac.uk

Abstract

We present a novel rewriting technique for conjunctive query answering over *OWL 2 QL* ontologies. In general, the obtained rewritings are not necessarily correct and can be of exponential size in the length of the query. We argue, however, that in most, if not all, practical cases the rewritings are correct and of polynomial size. Moreover, we prove some sufficient conditions, imposed on queries and ontologies, that guarantee correctness and succinctness. We also support our claim by experimental results.

Introduction

OWL 2 QL, one of three profiles of the Web Ontology Language *OWL 2*, was designed with the aim of supporting ontology-based data access (OBDA). The key idea is that data, 'stored in a standard relational database management system (RDBMS), can be queried through an *OWL 2 QL* ontology via a simple rewriting mechanism, i.e., by rewriting the query into an SQL query that is then answered by the RDBMS, without any changes to the data' (www.w3.org/TR/owl2-profiles). The rewritability property ensures, in particular, that the data complexity of answering queries over *OWL 2 QL* ontologies matches the complexity of database query answering, which is in AC^0.

It has been observed, however, that the available 'rewriting mechanisms' for *OWL 2 QL* (Calvanese et al. 2007a; Pérez-Urbina, Motik, and Horrocks 2009; Rosati and Almatelli 2010; Chortaras, Trivela, and Stamou 2011; Gottlob, Orsi, and Pieris 2011) are actually not so 'simple.' In fact, the rewritten queries are often too long to be executed by modern RDBMSs, and the question whether 'short' rewritings exist has attracted considerable attention over the last two years. For example, Kikot, Kontchakov, and Zakharyaschev (2011) showed that no polynomial algorithm can construct a 'pure rewriting' of a conjunctive query (CQ) $\boldsymbol{q}$ over an *OWL 2 QL* ontology $\mathcal{T}$. Here by a *pure rewriting* we mean any first-order (FO) rewriting with the same signature (predicates and constants) as $\boldsymbol{q}$ and $\mathcal{T}$, possibly with equality. On the other hand, Gottlob and Schwentick (2011) gave a polynomial-time ('impure') rewriting using additional constants and predicates. Optimised (but still exponential) pure rewritings to nonrecursive Datalog were suggested by Rosati and Almatelli (2010) and Gottlob, Orsi, and Pieris (2011). The combined approach of Kontchakov et al. (2010) was developed for *OWL 2 QL* without role inclusions; it uses a simple polynomial rewriting over the data expanded by applying the ontology axioms and introducing a small number of new individuals.

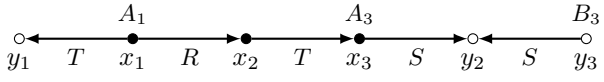
The diversity of approaches to query rewriting prompts another question: what is the type/shape/size of rewritings we should aim at to make OBDA with *OWL 2 QL* efficient? When trying to answer this question, we should bear in mind that (*i*) the OBDA paradigm relies on the proven efficiency of RDBMSs, but (*ii*) database query answering is not tractable in the size of queries (PSPACE-complete for FO queries and NP-complete for CQs). High efficiency of RDBMSs in practice appears to indicate only that answering real-world queries over real-world databases turns out to be tractable. As rewritings can turn a standard query to something 'out of this world,' a first rule of thumb could be as follows: *the rewritten query should look similar to the original one*. In this respect, as Gottlob and Schwentick (2011) remark, their polynomial rewriting is of rather 'theoretical nature' (it uses the extra constants, predicates and existentially quantified variables to encode, by making nondeterministic guesses, a relevant part of the chase, aka the canonical model; see the discussion in Conclusions for more details).

The aim of this paper is to investigate what causes exponentially long pure rewritings of CQs over *OWL 2 QL* ontologies and to check experimentally whether those 'bad guys' occur in real-world queries and ontologies. As a result of this analysis, we suggest some short (polynomial-size) rewritings that cover most, if not all, practical cases.

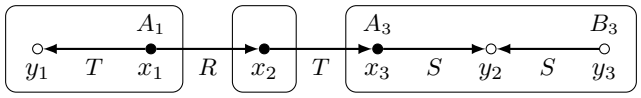
We think of a CQ as a labelled directed multigraph. For example, the query 'find x_1, x_2, x_3 for which

$$\exists y_1, y_2, y_3 \big(A_1(x_1) \land A_3(x_3) \land B_3(y_3) \land T(x_1, y_1) \land R(x_1, x_2) \land T(x_2, x_3) \land S(x_3, y_2) \land S(y_3, y_2)\big)$$

holds' can be represented as the graph



To answer a CQ $\boldsymbol{q}(\vec{x})$ over data $\mathcal{A}$ and ontology $\mathcal{T}$, we seek homomorphisms from $\boldsymbol{q}$ to the structure, called the canonical model and obtained by expanding $\mathcal{A}$ (extensional data) with knowledge in $\mathcal{T}$ (intensional data). Thus, we have to find possible cuts of $\boldsymbol{q}$ into a number of pieces: some of them—say, of type (A)—are mapped to individuals in $\mathcal{A}$, while the others—of type (B)—may only have some of their terms ('roots') in $\mathcal{A}$, whereas the remaining part is implied by the knowledge in $\mathcal{T}$. For example, the query above can be cut, for a suitable $\mathcal{T}$, into 3 pieces of which only the middle one is of type (A), and the right piece has two roots x_3, y_3:



If $\boldsymbol{q}(\vec{x})$ does not have existentially quantified variables then the whole $\boldsymbol{q}$ forms the only possible cut of type (A), because the answer variables $\vec{x}$ must be mapped to individuals in $\mathcal{A}$. If $\mathcal{T}$ does not contain role inclusion axioms then, as shown by Kontchakov et al. (2010; 2011), every root determines a unique piece of type (B). However, in general, $\boldsymbol{q}$ may have exponentially many pieces of type (B). One can encode the intuition above as a pure positive existential rewriting $\boldsymbol{q}_e$, which is, roughly speaking, a disjunction over all possible cuts of $\boldsymbol{q}$, and so is exponential in $|\boldsymbol{q}|$ in the worst case.

A slightly different approach to checking possible cuts of $\boldsymbol{q}$ is to consider, for every edge in $\boldsymbol{q}$, whether it belongs to an (A) piece, or generates a (B) piece itself, or lies inside the (B) piece generated by some other edge. This 'local' view gives another rewriting, a conjunction of disjunctions $\boldsymbol{q}_c$, which may still be exponential as the same edge may generate exponentially many distinct (B) pieces. (Two (B) pieces are different if their domains or roots are different.) Moreover, the new rewriting is not necessarily correct because some (B) pieces may not be realised together in the canonical model; we call such pieces conflicting.

We analyse—both theoretically and experimentally—conditions under which the number of (B) pieces is polynomial and they are not in conflict with each other. In particular, we develop techniques to ensure that rewritings are not affected by conflicting pieces. For example, one simple—but impure—approach involves a single fresh constant to represent all intensional objects in the canonical model (labelled nulls in the chase), but no extra variables.

We show that the number of (B) pieces generated by one edge in the query $\boldsymbol{q}$ is largely determined by a sophisticated interaction between role inclusions and inverse roles in the ontology $\mathcal{T}$, which can produce canonical models with very complex intensional parts. We give a sufficient condition (on $\boldsymbol{q}$ and $\mathcal{T}$) which guarantees that one edge in $\boldsymbol{q}$ can generate at most one piece of type (B) (though the number of ways this piece can be matched in the canonical model can still be exponential). This leads to our shortest pure rewriting, $\boldsymbol{q}_p$, which can be constructed in polynomial time, but is correct only if $\boldsymbol{q}$ and $\mathcal{T}$ satisfy the sufficient condition, which can also be checked in polynomial time. Trivial examples where this condition holds are ontologies without inverse roles, ontologies without role inclusions and qualified existential quantification, or those without positive occurrences of existential quantifiers. Our experiments with a number of standard *OWL 2 QL* ontologies and queries demonstrate that the sufficient conditions always apply, the queries normally contain very few pieces of type (B), if any, and moreover, these pieces are never in conflict. Thus, in practice the rewritings $\boldsymbol{q}_c$ and $\boldsymbol{q}_p$ are both short and correct.

Omitted proofs can be found in the full version of the paper at `www.dcs.bbk.ac.uk/~kikot`.

OWL 2 QL

The language of *OWL 2 QL* contains *individual names* a_i, *concept names* A_i, and *role names* P_i ($i \geq 1$). *Roles* R, *basic concepts* B and *concepts* C are defined by the grammar:

$$\begin{array}{lcl} R & ::= & P_i \mid P_i^-, \\ B & ::= & \bot \mid A_i \mid \exists R, \\ C & ::= & B \mid \exists R.B \end{array}$$

A *TBox*, $\mathcal{T}$, is a finite set of *inclusions* of the form

$$B \sqsubseteq C, \qquad R_1 \sqsubseteq R_2,$$
$$B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot.$$

(Note that concepts of the form $\exists R.B$ can only occur in the right-hand side of concept inclusions in *OWL 2 QL*. An inclusion $B' \sqsubseteq \exists R.B$ can be regarded as an abbreviation for three inclusions: $B' \sqsubseteq \exists R_B$, $\exists R_B^- \sqsubseteq B$ and $R_B \sqsubseteq R$, where R_B is a fresh role name.) An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $A_k(a_i)$ and $P_k(a_i, a_j)$. $\mathcal{T}$ and $\mathcal{A}$ together constitute the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The semantics for *OWL 2 QL* is defined in the usual way based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$; consult (Baader et al. 2003) for details. The set of individual names in $\mathcal{A}$ will be denoted by $\mathsf{ind}(\mathcal{A})$. For concepts or roles E_1 and E_2, we write $E_1 \sqsubseteq_{\mathcal{T}} E_2$ if $\mathcal{T} \models E_1 \sqsubseteq E_2$; and we set $[E] = \{E' \mid E \sqsubseteq_{\mathcal{T}} E' \text{ and } E' \sqsubseteq_{\mathcal{T}} E\}$.

A *conjunctive query* (CQ) $\boldsymbol{q}(\vec{x})$ is a first-order formula $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$, where φ is constructed, using $\wedge$, from atoms of the form $A_k(t_1)$ and $P_k(t_1, t_2)$, where each t_i is a *term* (an individual or a variable from $\vec{x}$ or $\vec{y}$). Variables in $\vec{x}$ are called *answer variables*, and those in $\vec{y}$ *bound* variables. A tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer* to $\boldsymbol{q}(\vec{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}[\vec{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$. To simplify notation, we will often identify $\boldsymbol{q}$ with the set of its atoms and use $P^-(t, t') \in \boldsymbol{q}$ as a synonym of $P(t', t) \in \boldsymbol{q}$; $\mathsf{term}(\boldsymbol{q})$ is the set of terms in $\boldsymbol{q}$. We call $\boldsymbol{q}$ *tree-shaped* if its primal graph $(\mathsf{term}(\boldsymbol{q}), \{\{t, t'\} \mid R(t, t') \in \boldsymbol{q}\})$ is a tree.

Remark 1 Although the official *OWL 2 QL* contains the concept $\top$ (for the whole domain), we do not consider it here as $\top$ makes OBDA with *OWL 2 QL* *domain dependent* (Abiteboul, Hull, and Vianu 1995): take, for example, the query $A(x)$ over the ontology $\{\top \sqsubseteq A\}$. (But we shall use $\top$ as an auxiliary symbol as it is convenient to regard $\exists R$ as an abbreviation for $\exists R.\top$.) To simplify presentation, we omit data properties and (ir)reflexivity constraints for roles.

Query answering over *OWL 2 QL* KBs is based on the fact that, for any consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, there is an interpretation $\mathcal{C}_{\mathcal{K}}$ such that, for all CQs $\boldsymbol{q}(\vec{x})$ and $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$,

we have $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{C}_\mathcal{K} \models \boldsymbol{q}[\vec{a}]$. The interpretation $\mathcal{C}_\mathcal{K}$, called the *canonical model* of $\mathcal{K}$, can be constructed as follows. For each pair $[R], [B]$ with $\exists R.B$ in $\mathcal{T}$ (recall that $\exists R.\top$ is another way of writing $\exists R$), we introduce a fresh symbol $w_{[RB]}$ and call it the *witness for* $\exists R.B$. We write $\mathcal{K} \models C(w_{[RB]})$ if $\exists R^- \sqsubseteq_\mathcal{T} C$ or $B \sqsubseteq_\mathcal{T} C$. Define a *generating relation*, $\rightsquigarrow$, on the set of these witnesses together with $\mathsf{ind}(\mathcal{A})$ by taking:

- $a \rightsquigarrow w_{[RB]}$ if $a \in \mathsf{ind}(\mathcal{A})$, $[R]$ and $[B]$ are $\sqsubseteq_\mathcal{T}$-minimal such that $\mathcal{K} \models \exists R.B(a)$ and there is no $b \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{K} \models R(a,b) \land B(b)$;
- $w_{[R'B']} \rightsquigarrow w_{[RB]}$ if $u \rightsquigarrow w_{[R'B']}$, for some u, $[R]$ and $[B]$ are $\sqsubseteq_\mathcal{T}$-minimal such that $\mathcal{K} \models \exists R.B(w_{[R'B']})$ and it is not the case that $R' \sqsubseteq_\mathcal{T} R^-$ and $\mathcal{K} \models B(u)$.

If $a \rightsquigarrow w_{[R_1B_1]} \rightsquigarrow \cdots \rightsquigarrow w_{[R_nB_n]}$, $n \geq 0$, then we say that a *generates* the path $aw_{[R_1B_1]} \cdots w_{[R_nB_n]}$. Denote by $\mathsf{path}_\mathcal{K}(a)$ the set of paths generated by a, and by $\mathsf{tail}(\pi)$ the last element in $\pi \in \mathsf{path}_\mathcal{K}(a)$. $\mathcal{C}_\mathcal{K}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{C}_\mathcal{K}} &= \bigcup_{a \in \mathsf{ind}(\mathcal{A})} \mathsf{path}_\mathcal{K}(a), \quad a^{\mathcal{C}_\mathcal{K}} = a, \text{ for } a \in \mathsf{ind}(\mathcal{A}),\\
A^{\mathcal{C}_\mathcal{K}} &= \{\pi \in \Delta^{\mathcal{C}_\mathcal{K}} \mid \mathcal{K} \models A(\mathsf{tail}(\pi))\},\\
P^{\mathcal{C}_\mathcal{K}} &= \{(a,b) \in \mathsf{ind}(\mathcal{A}) \times \mathsf{ind}(\mathcal{A}) \mid \mathcal{K} \models P(a,b)\} \cup\\
&\quad \{(\pi, \pi \cdot w_{[RB]}) \mid \mathsf{tail}(\pi) \rightsquigarrow w_{[RB]},\ R \sqsubseteq_\mathcal{T} P\} \cup\\
&\quad \{(\pi \cdot w_{[RB]}, \pi) \mid \mathsf{tail}(\pi) \rightsquigarrow w_{[RB]},\ R \sqsubseteq_\mathcal{T} P^-\}.
\end{aligned}$$

$\Delta^{\mathcal{C}_\mathcal{K}} \setminus \mathsf{ind}(\mathcal{A})$ is called the *tree part* of $\mathcal{C}_\mathcal{K}$. The following result is proved in a standard way (Kontchakov et al. 2010):

Theorem 2 *For every OWL 2 QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, every CQ $\boldsymbol{q}(\vec{x})$ and every $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{C}_\mathcal{K} \models \boldsymbol{q}[\vec{a}]$.*

Given a TBox $\mathcal{T}$, define a KB $\mathcal{K}_\mathcal{T} = (\mathcal{T}, \mathcal{A}_\mathcal{T})$ by taking

$$\begin{aligned}
\mathcal{A}_\mathcal{T} = \{&R(a_{RB}, b_{RB}), B(b_{RB}) \mid\\
&\exists R.B \text{ is a } \mathcal{T}\text{-consistent concept in } \mathcal{T}\}.
\end{aligned}$$

In other words, $\mathcal{C}_{\mathcal{K}_\mathcal{T}}$ is the disjoint union of the canonical models for consistent $(\mathcal{T}, \{R(a_{RB}, b_{RB}), B(b_{RB})\})$. To simplify notation, we use $\mathcal{C}_\mathcal{T}$ in place of $\mathcal{C}_{\mathcal{K}_\mathcal{T}}$. We extend the generating relation $\rightsquigarrow$ in $\mathcal{C}_\mathcal{T}$ by adding to it the pair $a_{RB} \rightsquigarrow b_{RB}$. The *$RB$-subtree* of $\mathcal{C}_\mathcal{T}$ has root a_{RB} and consists of the full subtree of $\mathcal{C}_\mathcal{T}$ with root b_{RB} extended with the edge (a_{RB}, b_{RB}). (Note that there may be other branches starting from a_{RB} in $\mathcal{C}_\mathcal{T}$, for example, if $\exists R \sqsubseteq_\mathcal{T} \exists S$ and $R \not\sqsubseteq_\mathcal{T} S$.)

In the next section, we introduce the naïve exponential rewriting of CQs over *OWL 2 QL* ontologies sketched in the introduction, and analyse whether it can be made shorter.

Two Rewritings

Let $\boldsymbol{q}(\vec{x})$ be a CQ and $\mathcal{T}$ an *OWL 2 QL* ontology. Our task is to construct an FO query $\boldsymbol{q}'(\vec{x})$, using the predicates and individuals of $\boldsymbol{q}$, $\mathcal{T}$ and $=$, such that, for any ABox $\mathcal{A}$ and any $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}'[\vec{a}]$, where $\mathcal{I}_\mathcal{A}$ is the interpretation with domain $\mathsf{ind}(\mathcal{A})$ given by the atoms in $\mathcal{A}$. Such a query $\boldsymbol{q}'$ is called a (*pure*) *rewriting of $\boldsymbol{q}$ and $\mathcal{T}$*. As argued in the introduction, we are interested in rewritings $\boldsymbol{q}'$ that are (*a*) as short as possible (ideally, polynomial in $|\boldsymbol{q}|$ and $|\mathcal{T}|$), (*b*) look similar to the original CQ $\boldsymbol{q}$ (in particular, built from the same predicates and terms as $\boldsymbol{q}$ and $\mathcal{T}$), and (*c*) can be constructed in reasonable time.

Without loss of generality, we will assume that the primal graph of $\boldsymbol{q}$ is connected. (If this is not the case, we consider each of the connected components separately.)

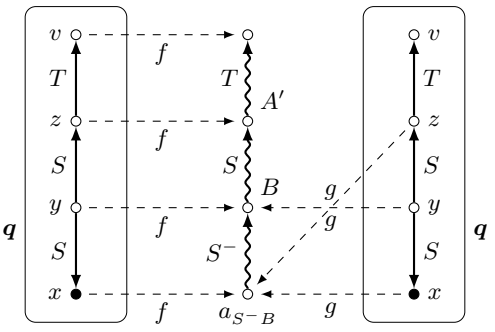
To understand the ingredients required for such a rewriting, suppose $\boldsymbol{q}(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $\mathcal{C}_\mathcal{K} \models^\mathfrak{a} \varphi$, where $\mathfrak{a}$ is an assignment of elements of $\Delta^{\mathcal{C}_\mathcal{K}}$ to the variables in φ under which $\mathfrak{a}(x) \in \mathsf{ind}(\mathcal{A})$ for all $x \in \vec{x}$. Consider an atom $P(z, z') \in \boldsymbol{q}$ with bound variables z, z' and assume that $\mathfrak{a}(t) \in \mathsf{ind}(\mathcal{A})$, for some $t \in \mathsf{term}(\boldsymbol{q})$. The assignment $\mathfrak{a}$ can send z and z' to four different locations in $\mathcal{C}_\mathcal{K}$: (A) $\mathfrak{a}(z), \mathfrak{a}(z') \in \mathsf{ind}(\mathcal{A})$; (B) $\mathfrak{a}(z) \in \mathsf{ind}(\mathcal{A})$, $\mathfrak{a}(z') \notin \mathsf{ind}(\mathcal{A})$; (B$^-$) $\mathfrak{a}(z) \notin \mathsf{ind}(\mathcal{A})$, $\mathfrak{a}(z') \in \mathsf{ind}(\mathcal{A})$; (O) $\mathfrak{a}(z), \mathfrak{a}(z') \notin \mathsf{ind}(\mathcal{A})$. Let us see how these alternatives can be reflected in our rewriting. In all of these cases, we need formulas of the form

$$\begin{aligned}
\mathsf{ext}_P(x,y) &= \bigvee_{R \sqsubseteq_\mathcal{T} P} R(x,y),\\
\mathsf{ext}_C(x) &= \bigvee_{A \sqsubseteq_\mathcal{T} C} A(x) \lor \bigvee_{\exists R \sqsubseteq_\mathcal{T} C} \exists y\, R(x,y).
\end{aligned}$$

In case (A) we have $\mathcal{C}_\mathcal{K} \models^\mathfrak{a} P(z,z')$ iff $\mathcal{A} \models^\mathfrak{a} \mathsf{ext}_P(z,z')$. Case (B) is possible only if, for some concept $\exists R.B$, we have $\mathfrak{a}(z) \rightsquigarrow w_{[RB]}$, $R \sqsubseteq_\mathcal{T} P$ and the atoms of $\boldsymbol{q}$ 'linked to' z' can be mapped into the RB-subtree of $\mathcal{C}_\mathcal{T}$. To illustrate, consider an example.

Example 3 Let $\mathcal{T} = \{A \sqsubseteq \exists S^-.B, B \sqsubseteq \exists S.A', A' \sqsubseteq \exists T\}$ and $\boldsymbol{q}(x) = \{S(y,x), S(y,z), T(z,v)\}$. The answer variable x must be mapped by $\mathfrak{a}$ to an ABox element. However, y can be mapped either to an ABox element or to the point $\mathfrak{a}(x) \cdot w_{[S^-B]}$ provided that $\mathfrak{a}(x)$ is an instance of A (and so of $\exists S^-.B$). In the latter case, we have two possibilities for mapping z: either to $\mathfrak{a}(x) \cdot w_{[S^-B]} \cdot w_{[SA']}$, in which case we must further set $\mathfrak{a}(v) = \mathfrak{a}(x) \cdot w_{[S^-B]} \cdot w_{[SA']} \cdot w_{[T\top]}$, or to $\mathfrak{a}(x)$, provided that $\mathfrak{a}(x)$ is an instance of $\exists T$. In the picture below, these two (partial) maps are denoted by f and g.

S^-B-subtree of $\mathcal{C}_\mathcal{T}$



The observations made in Example 3 are formalised in our central definition. Given a pair (t, t') of adjacent terms in (the graph of) $\boldsymbol{q}$, we define a *tree witness for* (t, t') to be

a homomorphism f (with domain $\mathsf{dom}\, f$) from the query

$$\boldsymbol{q}_f = \big\{S(s,s') \in \boldsymbol{q} \mid s, s' \in \mathsf{dom}\, f\big\} \cup \big\{A(s) \in \boldsymbol{q} \mid s \in \mathsf{dom}\, f \setminus f^{-1}(\mathsf{r}_f)\big\}$$

to the RB-subtree of $\mathcal{C}_\mathcal{T}$, for some $\exists R.B$, with root $\mathsf{r}_f = a_{RB}$ such that the following conditions hold:

(t1) $\mathsf{dom}\, f$ is the smallest set containing t, t' and such that if $s \in \mathsf{dom}\, f \setminus f^{-1}(\mathsf{r}_f)$ and $S(s,s') \in \boldsymbol{q}$ then $s' \in \mathsf{dom}\, f$,

(t2) $f(t) = \mathsf{r}_f$ and if $s \in \mathsf{dom}\, f \setminus f^{-1}(\mathsf{r}_f)$ then s is bound.

Note that this notion of tree witness is different from the one used for the description logic *DL-Lite*$^{\mathcal{N}}_{horn}$ by Kontchakov et al. (2010), where the structure of the canonical models ensured uniqueness of a tree witness if it existed. In Example 3, there are two tree witnesses, f and q, for (x, y). As we shall see below, there may be exponentially many of them.

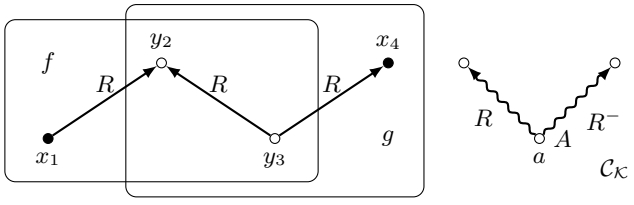
Returning back to case (B), we can say now that there must exist a tree witness f for (z, z') such that $\mathfrak{a}(z)$ satisfies all $A(s) \in \boldsymbol{q}$ with $s \in f^{-1}(\mathsf{r}_f)$. Case ($\mathrm{B}^-$) is symmetric, and in case (O) there must exist $R(t, t') \in \boldsymbol{q}$ for which (B) holds and $P(z, z')$ is 'covered' by a tree witness for (t, t') (as we assumed that $\mathfrak{a}(t) \in \mathsf{ind}(\mathcal{A})$, for some $t \in \mathsf{term}(\boldsymbol{q})$).

This analysis suggests the following idea. Given a CQ $\boldsymbol{q}$ and KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we guess pairs of adjacent terms (t, t') in $\boldsymbol{q}$ that will be mapped to edges of the tree part of $\mathcal{C}_\mathcal{K}$ starting from $\mathsf{ind}(\mathcal{A})$ (as in case (B) above) and, for each such pair (t, t'), we guess a tree witness for (t, t'). The part of the query that is not covered by the chosen tree witnesses will be mapped to $\mathsf{ind}(\mathcal{A})$ (as in case (A)). The query representing these guesses is then evaluated over $\mathcal{I}_\mathcal{A}$. If $\boldsymbol{q}$ has no answer variables, then we also have to take account of the case where the whole $\boldsymbol{q}$ is mapped into the tree part of $\mathcal{C}_\mathcal{K}$. Unfortunately, this idea cannot be implemented in a straightforward way, as shown by the following example.

Example 4 Let $\mathcal{K} = (\mathcal{T}, \{A(a)\})$, where $\mathcal{T} = \{A \sqsubseteq \exists R, A \sqsubseteq \exists R^-\}$. Consider the query

$$\boldsymbol{q}(x_1, x_4) = \{R(x_1, y_2), R(y_3, y_2), R(y_3, x_4)\}$$

shown in the picture below alongside $\mathcal{C}_\mathcal{K}$.



We obviously have a tree witness f for (x_1, y_2) such that $\mathsf{dom}\, f = \{x_1, y_2, y_3\}$ and $f^{-1}(\mathsf{r}_f) = \{x_1, y_3\}$, and also a tree witness g for (x_4, y_3) with $\mathsf{dom}\, g = \{x_4, y_3, y_2\}$ and $g^{-1}(\mathsf{r}_f) = \{x_4, y_2\}$. Although these tree witnesses cover the whole query $\boldsymbol{q}$, they are only 'realised' in $\mathcal{C}_\mathcal{K}$ under *conflicting* maps: f sends x_1, y_3 to a and y_2 to $a \cdot w_{[R\top]}$, while g sends x_4, y_2 to a and y_3 to $a \cdot w_{[R^-\top]}$; in fact, $\mathcal{K} \not\models \boldsymbol{q}(a, a)$.

This example motivates the following definitions. Let Θ be the set of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$. We say that $f, g \in \Theta$ are *compatible* if $\mathsf{dom}\, f \cap \mathsf{dom}\, g \subseteq f^{-1}(\mathsf{r}_f) \cap g^{-1}(\mathsf{r}_g)$. If f and g are incompatible and neither $\mathsf{dom}\, f \subseteq \mathsf{dom}\, g$ nor $\mathsf{dom}\, g \subseteq \mathsf{dom}\, f$, then we call f and g *conflicting*. In Example 4, $\mathsf{dom}\, f \cap \mathsf{dom}\, g = \{y_2, y_3\}$, and so f and g are conflicting.

We are now in a position to formulate a rewriting based on the idea discussed above. Given a tree witness $f \in \Theta$ for (t, t') with $\mathsf{r}_f = a_{RB}$, we first define a *tree-witness formula* tw_f for f by taking

$$\mathsf{tw}_f = \mathsf{ext}_{\exists R.B}(t) \wedge \bigwedge_{s \in f^{-1}(\mathsf{r}_f)} \big((t = s) \wedge \bigwedge_{A(s) \in \boldsymbol{q}} \mathsf{ext}_A(s)\big).$$

A (possibly empty) subset $\Xi \subseteq \Theta$ is called *consistent* if all pairs of tree witnesses in Ξ are compatible. Now, assuming that $\vec{y}$ is a list of all bound variables in $\boldsymbol{q}(\vec{x})$, we set

$$\boldsymbol{q}_e(\vec{x}) \;=\; \mathsf{detached}_\boldsymbol{q} \;\vee \bigvee_{\substack{\Xi \subseteq \Theta \\ \Xi \text{ consistent}}} \exists \vec{y}\, \Big(\bigwedge_{f \in \Xi} \mathsf{tw}_f \wedge \bigwedge_{\substack{A(s) \in \boldsymbol{q} \\ s \notin \mathsf{dom}\, f \\ \text{for all } f \in \Xi}} \mathsf{ext}_A(s) \wedge \bigwedge_{\substack{P(s,s') \in \boldsymbol{q} \\ \{s,s'\} \not\subseteq \mathsf{dom}\, f \\ \text{for all } f \in \Xi}} \mathsf{ext}_P(s,s') \Big),$$

where

– $\mathsf{detached}_\boldsymbol{q} = \bot$ if $\boldsymbol{q}$ has at least one answer variable; otherwise $\mathsf{detached}_\boldsymbol{q}$ is a disjunction of the sentences $\exists x\, \mathsf{ext}_{\exists R.B}(x)$ such that there is a homomorphism from $\boldsymbol{q}$ to the RB-subtree of $\mathcal{C}_\mathcal{T}$.

Clearly, $\boldsymbol{q}_e$ is a positive existential formula built from the atoms occurring in $\boldsymbol{q}$ and $\mathcal{T}$ together with equality unifying certain terms; it contains the same variables and constants as the original CQ $\boldsymbol{q}$. Moreover, if we consider the predicates ext_E for concepts and roles as primitive, then $\boldsymbol{q}_e$ is a union of conjunctive queries (UCQ), where each subquery can be thought of as the result of folding the respective pieces of $\boldsymbol{q}$ into the tree witness formulas.

Theorem 5 *For every ABox $\mathcal{A}$ and every $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}_e[\vec{a}]$.*

We illustrate the rewriting $\boldsymbol{q}_e$ by a simple example.

Example 6 Suppose that $\boldsymbol{q}(x) = \{R_i(x, y_i) \mid i \leq n\}$ and $\mathcal{T} = \{A_i \sqsubseteq \exists R_i \mid i \leq n\}$. Each pair (x, y_i) gives rise to one tree witness f_i with $\mathsf{tw}_{f_i} = A_i(x) \vee \exists y\, R_i(x, y)$, and $\boldsymbol{q}_e = \bigvee_{N \subseteq [0,n]} \exists \vec{y} (\bigwedge_{i \in N} \mathsf{tw}_{f_i} \wedge \bigwedge_{j \notin N} R_j(x, y_j))$.

As all other known pure rewritings for *OWL 2 QL*, $\boldsymbol{q}_e$ is of exponential size: $\mathcal{O}((n_{\mathcal{T},\boldsymbol{q}} + 1)^{|\boldsymbol{q}|} \cdot |\mathcal{T}| \cdot |\boldsymbol{q}|^2)$, where $n_{\mathcal{T},\boldsymbol{q}}$ is the maximum number of distinct tree witness formulas tw_f for tree witnesses containing a pair (t, t') of adjacent terms in $\boldsymbol{q}$. Two recent results may help to shed some light on whether this exponential blowup is unavoidable in *OWL 2 QL*.

One of them shows that no polynomial-time algorithm can construct *pure* rewritings for CQs over *OWL 2 QL* ontologies, unless P = NP (Kikot, Kontchakov, and Zakharyaschev 2011, Theorem 2). The idea of the proof is as follows. First, we encode any CNF $\chi = \bigwedge_{j=1}^m D_j$ over propositional variables $p_1, \ldots, p_n$ as an *OWL 2 QL* TBox,

$\mathcal{T}_\chi$, containing the axioms, for $1 \leq i \leq n$, $1 \leq j \leq m$ and $k = 0, 1$,

$$A_{i-1} \sqsubseteq \exists P^-.X_i^k, \qquad X_i^k \sqsubseteq A_i, \qquad C_j \sqsubseteq \exists P.C_j,$$
$$X_i^0 \sqsubseteq \exists P.C_j \quad \text{if } \neg p_i \in D_j, \qquad X_i^1 \sqsubseteq \exists P.C_j \quad \text{if } p_i \in D_j,$$

and consider the CQ

$$\boldsymbol{q}(y_0) = A_0(y_0) \wedge \textstyle\bigwedge_{i=1}^n P(y_i, y_{i-1}) \wedge A_n(y_n) \wedge \\ \textstyle\bigwedge_{j=1}^m \big(P(y_n, z_0^j) \wedge \bigwedge_{i=1}^n P(z_{i-1}^j, z_i^j) \wedge C_j(z_n^j)\big).$$

Now, suppose $\boldsymbol{q}'(y_0)$ is a rewriting of $\boldsymbol{q}(y_0)$ and $\mathcal{T}_\chi$ that *does not use any constants*. Consider the ABox $\mathcal{A} = \{A_0(a)\}$. It is not hard to see that $(\mathcal{T}_\chi, \mathcal{A}) \models \boldsymbol{q}[a]$ iff χ is satisfiable. On the other hand, checking whether $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}'[a]$ can be done in polynomial time in $|\boldsymbol{q}'|$ because the domain of $\mathcal{I}_\mathcal{A}$ is a singleton. Thus, constructing the rewriting $\boldsymbol{q}'$ must be at least as hard as deciding satisfiability of χ. (See the Conclusions for a further discussion and results.)

The argument above does not go through if databases are assumed to have two special constants, say 0 and 1, which can be employed in rewritings $\boldsymbol{q}'$. Indeed, as shown by Gottlob and Schwentick (2011), using 0 and 1, $\neq$ and fresh predicates of arity $\mathcal{O}(\log(|\boldsymbol{q}| \cdot |\mathcal{T}|))$, one can construct a nonrecursive Datalog rewriting $\boldsymbol{q}'$ for any given CQ $\boldsymbol{q}$ and *OWL 2 QL* ontology $\mathcal{T}$ in polynomial time. Intuitively, $\boldsymbol{q}'$ uses the extra resources to encode a part of the canonical model, which is enough to provide all certain answers to $\boldsymbol{q}$, to guess a map from $\boldsymbol{q}$ into this part and then check whether it is a homomorphism. As argued in the introduction, RDBMSs do not appear to be best suitable for tasks of that sort, although thorough experiments are required to confirm or refute this claim.

Very often, however, there are other ways to construct polynomial rewritings. Let us assume for a moment that the following condition holds:

(conf) *there are no conflicting tree witnesses in* Θ.

In this case, the query $\boldsymbol{q}_e$ can be transformed into the query

$$\boldsymbol{q}_c(\vec{x}) \;=\; \mathsf{detached}_{\boldsymbol{q}} \vee \\ \exists \vec{y} \bigwedge_{\substack{\{t,t'\} \\ t \text{ and } t' \\ \text{adjacent}}} \Big[\Big(\bigwedge_{\substack{A(s) \in \boldsymbol{q} \\ s \in \{t,t'\}}} \mathsf{ext}_A(s) \wedge \bigwedge_{R(t,t') \in \boldsymbol{q}} \mathsf{ext}_R(t,t') \Big) \vee \bigvee_{\substack{f \in \Theta \\ t,t' \in \mathsf{dom}\, f}} \mathsf{tw}_f \Big].$$

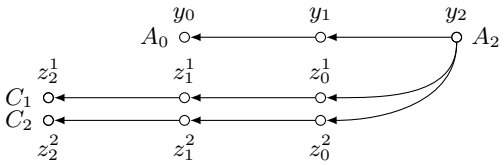
If $\boldsymbol{q}$ does not contain binary predicates then there cannot be any tree witnesses and we set $\boldsymbol{q}_c = \boldsymbol{q}_e$.

Theorem 7 *If $\mathcal{T}$ and $\boldsymbol{q}$ satisfy* **(conf)** *then, for any ABox $\mathcal{A}$ and $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}_c[\vec{a}]$.*

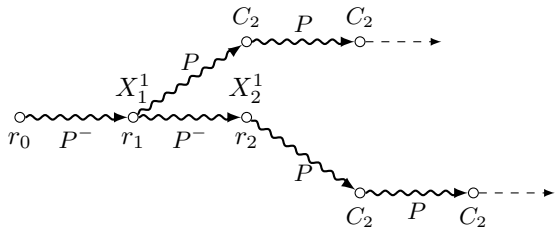
The size of $\boldsymbol{q}_c$ is $\mathcal{O}(n_{\mathcal{T},\boldsymbol{q}} \cdot |\mathcal{T}| \cdot |\boldsymbol{q}|^2)$. Thus, if every pair (t, t') of adjacent terms in $\boldsymbol{q}$ gives rise to polynomially many distinct tree-witness formulas tw_f and condition **(conf)** holds then $\boldsymbol{q}_c$ is a *polynomial* rewriting of $\boldsymbol{q}$ and $\mathcal{T}$. (If **(conf)** does not hold then $\boldsymbol{q}_c$ may return wrong answers.)

Example 8 The exponential query $\boldsymbol{q}_e$ in Example 6 reduces to polynomial $\boldsymbol{q}_c = \exists \vec{y} \bigwedge_{i \leq n} (R_i(x, y_i) \vee \mathsf{tw}_{f_i})$.

A good illuminative example of a CQ with exponentially many tree witness formulas is $\boldsymbol{q}(y_0)$ over the TBox $\mathcal{T}_\chi$ constructed above for a CNF χ. One can show that the pairs (z_i^j, z_{i-1}^j) give rise to exponentially many different tree witness formulas tw_f for a suitable χ.



To illustrate, consider the CQ $\boldsymbol{q}(y_0)$ for $n = m = 2$ and suppose that χ is such that the $P^- X_1^1$-subtree of $\mathcal{C}_{\mathcal{T}_\chi}$ contains the fragment as shown in the picture below.



We can construct a tree witness f for (z_1^1, z_0^1) by taking $f(z_1^1) = r_0$, $f(z_0^1) = r_1$, $f(y_2) = r_2$, after which we have three different options for defining $f(z_1^2)$ and $f(z_0^2)$: go back to r_0, go to r_1 and then take the C_2-branch, or take the C_2-branch starting from r_2. The last two tree witnesses give the same tree witness formula, which is different from that given by the first tree witness. Imagine now some large n and m. It is this fact that makes it 'hard' to construct a rewriting for $\boldsymbol{q}(y_0)$ and $\mathcal{T}_\chi$.

The TBox $\mathcal{T}_\chi$ and CQ $\boldsymbol{q}(y_0)$ involve a very complex interplay between role inclusions (or concepts of the form $\exists R.C$) and inverse roles, which appears to be rather artificial compared to how roles are used in real-world ontologies. The experiments to be reported later on in the paper demonstrate that real-world ontologies and queries generate very few tree witnesses, which are never in conflict, and so the rewriting $\boldsymbol{q}_c$ is both short and correct.

In the next section we analyse conflicting tree witnesses in more detail.

Conflicting Tree Witnesses

One simple way to tackle the problem of conflicting tree witnesses, identified in Example 4, would be to introduce a *fresh constant symbol*, say ν, representing all the non-ABox elements of the canonical models. We then use the following variant of the tree witness formula tw_f for (t, t'):

$$\mathsf{tw}_f^\nu \;=\; \mathsf{tw}_f \wedge \bigwedge_{s \,\in\, \mathsf{dom}\, f \setminus f^{-1}(\mathsf{r}_f)} (s = \nu).$$

We denote the resulting 'impure' rewriting by $\boldsymbol{q}_c^\nu$. Given an ABox $\mathcal{A}$, denote by $\mathcal{I}_\mathcal{A}^\nu$ the interpretation $\mathcal{I}_\mathcal{A}$ extended with a new domain element (interpreting) ν. Thus, ν is not involved in the interpretation of any predicate in $\mathcal{I}_\mathcal{A}$, and so, of any predicate ext_E.

Example 9 In the context of Example 4, tw_g^ν would contain the conjuncts $\mathsf{ext}_{\exists R^-.\top}(x_4)$ and $(y_2 = x_4)$, which cannot be satisfied in $\mathcal{I}_\mathcal{A}^\nu$ at the same time as the conjunct $(y_2 = \nu)$ of tw_f^ν. Thus, $\mathcal{I}_\mathcal{A}^\nu \not\models \boldsymbol{q}_c^\nu(a, a)$.

The following result shows that $\boldsymbol{q}_c^\nu$ is a correct rewriting over this interpretation.

Theorem 10 *For any ABox $\mathcal{A}$ and any $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_\mathcal{A}^\nu \models \boldsymbol{q}_c^\nu[\vec{a}]$.*

The rewriting $\boldsymbol{q}_c^\nu$ can be viewed as a step in the direction of the combined approach (Lutz, Toman, and Wolter 2009; Kontchakov et al. 2010), though without expanding the ABox by applying the TBox axioms. Roughly, in both cases the RDBMS has to guess whether a bound variable is mapped to $\mathsf{ind}(\mathcal{A})$ or to the tree part of the canonical model.

There is another way to 'suppress conflicts' in $\boldsymbol{q}_c$ for the majority of practical cases, while keeping the resulting rewriting 'pure.' For a CQ $\boldsymbol{q}$, let $\boldsymbol{q}_c^+$ be the rewriting obtained from $\boldsymbol{q}_c$ by replacing every tw_f in it with the following formula tw_f^+:

$$\mathsf{tw}_f^+ = \mathsf{tw}_f \wedge \bigwedge_{f,g \text{ conflicting}} (\boldsymbol{q}_{g \setminus f})_c^+,$$

where $\boldsymbol{q}_{g\setminus f}$ denotes the restriction of the query $\boldsymbol{q}$ to the set $\mathsf{dom}\, g \setminus (\mathsf{dom}\, f \setminus f^{-1}(\mathsf{r}_f))$ (i.e., all terms in g that are not non-root terms in f) and $(\boldsymbol{q}_{g\setminus f})_c^+$, in turn, is the rewriting (as defined above) of the query $\boldsymbol{q}_{g\setminus f}$ in which all the variables in $f^{-1}(\mathsf{r}_f) \cup g^{-1}(\mathsf{r}_g)$ are regarded to be answer variables.

Example 11 In the context of Example 4, the rewriting $\boldsymbol{q}_c^+$ is constructed in two steps as follows: first, we obtain a representation of $\boldsymbol{q}_c^+$ as the following formula:

$$\exists y_2, y_3 \left(\left(R(x_1, y_2) \vee \mathsf{tw}_f^+ \right) \wedge \left(R(y_3, y_2) \vee \mathsf{tw}_f^+ \vee \mathsf{tw}_g^+ \right) \wedge \left(R(y_3, x_4) \vee \mathsf{tw}_g^+ \right) \right),$$

where

$$\mathsf{tw}_f^+ = \mathsf{ext}_{\exists R}(x_1) \wedge (x_1 = y_3) \wedge (R(y_3, x_4))_c^+,$$
$$\mathsf{tw}_g^+ = \mathsf{ext}_{\exists R^-}(x_4) \wedge (x_4 = y_2) \wedge (R(x_1, y_2))_c^+,$$

with $R(y_3, x_4)$ and $R(x_1, y_2)$ being two new queries, which have only answer variables. Then, the rewritings of these two queries coincide with the queries themselves: $R(y_3, x_4)$ and $R(x_1, y_2)$, respectively. So, for example, the modified tree witness formula tw_f^+ for f contains the conjunct $R(y_3, x_4)$, which cannot be satisfied in the interpretation $\mathcal{I}_\mathcal{A}$ with $\mathcal{A} = \{A(a)\}$.

Theorem 12 *Suppose that $\boldsymbol{q}$ and $\mathcal{T}$ satisfy the condition*

(conf$_1$) *for every $f \in \Theta$, if there are $g, h \in \Theta$ such that the pairs f, g and f, h are both conflicting, then either $\mathsf{dom}\, g \subseteq \mathsf{dom}\, h$ or $\mathsf{dom}\, h \subseteq \mathsf{dom}\, g$.*

Then, for every ABox $\mathcal{A}$ and every $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_\mathcal{A} \models \boldsymbol{q}_c^+[\vec{a}]$.

We do not know any cases where $\boldsymbol{q}_c^+$ is not correct. It is to be noted, however, that the rewriting $\boldsymbol{q}_c^+$ can be of exponential size even if every pair (t, t') in $\boldsymbol{q}$ gives rise to at most one tree witness.

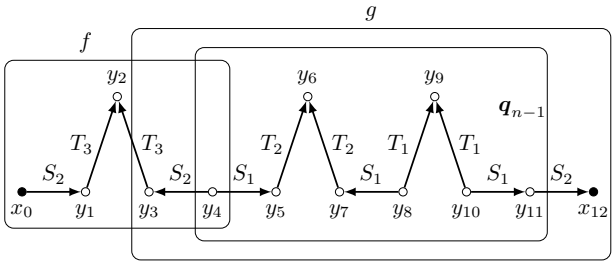
Example 13 Given a word $R_1 \dots R_m$ over roles, define the CQ

$$\boldsymbol{q}^{R_1 \dots R_m}(x_0, x_m) = \{R_i(x_{i-1}, x_i) \mid 1 \le i \le m\}.$$

Let σ_n be the following sequence of words of roles:

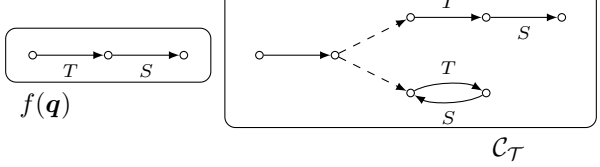
$$\sigma_1 = T_1 T_1^-, \quad \sigma_{n+1} = S_n T_{n+1} T_{n+1}^- S_n^- \sigma_n S_n, \text{ for } n \ge 1.$$

Now, consider the CQs $\boldsymbol{q}_n = \boldsymbol{q}^{\sigma_n}(x_0, x_n)$ and TBoxes $\mathcal{T}_n = \{A_n \sqsubseteq \exists R_n,\ A_{n+1} \sqsubseteq \exists S_n.A_n\}$. It is not hard to see that $\boldsymbol{q}_n$ contains conflicting tree witnesses f and g as shown in the picture below, the query $\boldsymbol{q}_{n-1}$ is a subquery of $\boldsymbol{q}_{g\setminus f}$, and so $(\boldsymbol{q}_{n-1})_c^+$ occurs in $(\boldsymbol{q}_n)_c^+$ four times. On the other hand, there is a constant c such that $|\boldsymbol{q}_n| = |\boldsymbol{q}_{n-1}| + c$.



Sufficient Conditions for Polynomial Rewriting

The rewriting $\boldsymbol{q}_c$ is exponentially long if there are exponentially many tree witness formulas tw_f for some pair (t, t') of adjacent terms in $\boldsymbol{q}$. Each tw_f is determined by the RB-subtree of $\mathcal{C}_\mathcal{T}$ (containing the range of f), the domain $\mathsf{dom}\, f$ of f and the set $f^{-1}(\mathsf{r}_f)$; the terms in this set will be called the *roots* of f. Now we show that, for a large class of CQs $\boldsymbol{q}$ and *OWL 2 QL* TBoxes $\mathcal{T}$, all tree witnesses for the same pair (t, t') in $\boldsymbol{q}$ have the same domain and roots. As the number of distinct RB-subtrees of $\mathcal{C}_\mathcal{T}$ is polynomial in the size of $\mathcal{T}$, the rewriting $\boldsymbol{q}_c$ in this case is also polynomial; moreover, we show that it can be constructed in polynomial time in $|\boldsymbol{q}|$ and $|\mathcal{T}|$. Roughly, Theorem 19 to be proved below demonstrates that this can be done if condition **(conf)** holds and there are no roles T, S and a tree witness f for which $f(\boldsymbol{q})$ and $\mathcal{C}_\mathcal{T}$ simultaneously contain fragments of the form



Here, by $f(\boldsymbol{q})$ we mean the quotient $\boldsymbol{q}_f/_\sim$ of $\boldsymbol{q}_f$ modulo the equivalence relation $\sim$ defined by taking $s \sim s'$ iff $f(s) = f(s')$. Roles T and S are called *adjacent in* $f(\boldsymbol{q})$ if $T(s_1, s_2), S(s_2, s_3) \in f(\boldsymbol{q})$, for some $s_i \in \mathsf{term}\, f(\boldsymbol{q})$ such that $s_2 \notin f^{-1}(\mathsf{r}_f)$.

We say that a role S is *forward in* $\mathcal{T}$ if $u \rightsquigarrow v$ for all $(u, v) \in S^{\mathcal{C}_\mathcal{T}}$. If neither S nor its inverse S^- is forward then S is said to be a *twisty role in* $\mathcal{T}$.

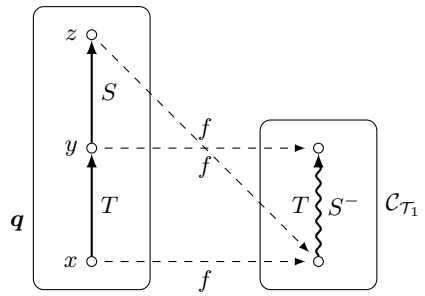
A tree witness f is called *perfect* if, for every pair T, S of adjacent roles in $f(\boldsymbol{q})$ such that S is twisty in $\mathcal{T}$, we have

(perf) $\mathcal{C}_\mathcal{T} \not\models inv(T,S) \wedge suc(T,S)$,

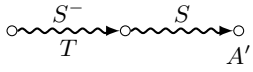
where

$$inv(T,S) = \exists x,y\,(T(x,y) \wedge S(y,x)),$$
$$suc(T,S) = \exists x,y,z\,(T(x,y) \wedge S(y,z) \wedge (x \neq z)).$$

Example 14 Consider $\mathcal{T}_1 = \{A \sqsubseteq \exists T,\ T \sqsubseteq S^-\}$ and $\boldsymbol{q}(x) = \{T(x,y), S(y,z)\}$. We have $\mathcal{C}_{\mathcal{T}_1} \models inv(T,S)$ and $\mathcal{C}_{\mathcal{T}_1} \models inv(R,R^-)$, for every role R. Both T and S^- are forward roles in $\mathcal{T}_1$. The homomorphism f shown below is a perfect tree witness for (x,y):



Consider now $\mathcal{T}_2 = \{A \sqsubseteq \exists T,\ T \sqsubseteq S^-,\ \exists T^- \sqsubseteq \exists S.A'\}$. As $\mathcal{C}_{\mathcal{T}_2}$ contains the following fragment



S is a twisty role in $\mathcal{T}_2$, and $\mathcal{C}_{\mathcal{T}_2} \models inv(T,S) \wedge suc(T,S)$. Thus, there is no *perfect* tree witness for (x,y) in $\boldsymbol{q}$ and $\mathcal{T}_2$, though there are two 'imperfect' tree witnesses.

It turns out that if all tree witnesses f for a pair of terms in $\boldsymbol{q}$ are perfect then all of them have the same domain and roots, and so (*i*) the tree-witness formulas tw_f occur in the disjunctions for the same edges (t,t') and (*ii*) the tw_f may differ only in their $\mathsf{ext}_{\exists R.B}(t)$ conjuncts. In the following example, exponentially many different tree witnesses give rise to the same tree witness formula.

Example 15 Let $\boldsymbol{q}(x) = \{S(x,y), R(y,z_i) \mid 1 \leq i \leq n\}$ and $\mathcal{T} = \{A \sqsubseteq \exists S,\ \exists S^- \sqsubseteq \exists R.B_1,\ \exists S^- \sqsubseteq \exists R.B_2\}$. There are 2^n (perfect) tree witnesses for (x,y), as each z_i can be mapped either to a B_1- or a B_2-point in $\mathcal{C}_\mathcal{T}$. All these tree witnesses have the same domain (all terms in $\boldsymbol{q}$) and only one root (x). They define the same tree-witness formula $\mathsf{ext}_{\exists S}(x)$.

The notion of universal tree witness we are about to introduce will serve as a compact representation for all such similar tree witnesses.

For a pair (t,t') of adjacent terms in $\boldsymbol{q}$, a tree-shaped CQ $\boldsymbol{f}$ is called a *universal tree witness for* (t,t') *in* $\boldsymbol{q}$ *and* $\mathcal{T}$ if there is a partial homomorphism f from $\boldsymbol{q}$ *onto* $\boldsymbol{f}$ such that, for every tree witness g for (t,t') in $\boldsymbol{q}$ and $\mathcal{T}$,

– $\mathsf{dom}\, g = \mathsf{term}(\boldsymbol{f})$, and

– there exists a forward homomorphism f' from $\boldsymbol{f}$ to $\mathcal{C}_\mathcal{T}$ such that $g(s) = f'(f(s))$, for every $s \in \mathsf{term}(\boldsymbol{f})$,

where by a *forward* homomorphism with understand a homomorphism that preserves the distance from the root (recall that both $\boldsymbol{f}$ and $\mathcal{C}_\mathcal{T}$ are tree-shaped).

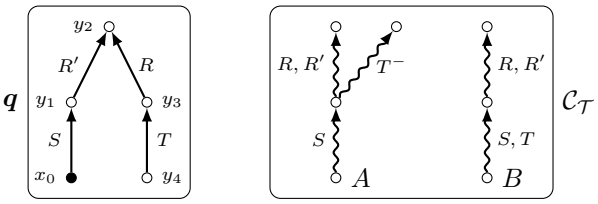
Lemma 16 (*i*) *If all tree witnesses for a pair* (t,t') *of terms in* $\boldsymbol{q}$ *and* $\mathcal{T}$ *are perfect, then there is a unique* (*up to isomorphism*) *universal tree witness for* (t,t') *in* $\boldsymbol{q}$ *and* $\mathcal{T}$.

(*ii*) *There is a polynomial-time algorithm which, given* $\boldsymbol{q}$, $\mathcal{T}$ *and a pair* (t,t') *of adjacent terms in* $\boldsymbol{q}$, *checks whether all the tree witnesses for* (t,t') *in* $\boldsymbol{q}$ *and* $\mathcal{T}$ *are perfect, and if this is the case, returns a universal tree witness for* (t,t') *in* $\boldsymbol{q}$ *and* $\mathcal{T}$.

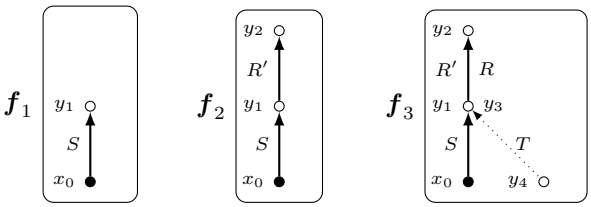
As a consequence of this lemma we obtain that, if all tree witnesses for (t,t') are perfect, then all of them share the same domain and roots.

Example 17 In Example 15, a universal tree witness for (x,y) coincides with the whole query $\boldsymbol{q}(x)$.

In the proof of Lemma 16, we construct a finite sequence $\boldsymbol{f}_i$ of tree-shaped CQs $\boldsymbol{f}_i$ such that the final $\boldsymbol{f}_n$ is a universal tree witness for (t,t'). We begin by taking all the atoms with terms t,t' and then try to satisfy the tree witness condition **(t1)** by adding atoms with new terms to the current $\boldsymbol{f}_i$. Condition **(perf)**, which is being checked during the construction, allows us to decide whether we have to add a new term to $\boldsymbol{f}_i$ or reuse an existing one. To illustrate, we give one more example.



Example 18 Consider $\mathcal{T} = \{A \sqsubseteq \exists S.C,\ C \sqsubseteq \exists T^-,\ \exists S^- \sqsubseteq \exists R,\ R \sqsubseteq R',\ B \sqsubseteq \exists T',\ T' \sqsubseteq T,\ T' \sqsubseteq S\}$ and $\boldsymbol{q}(x_0) = \{S(x_0,y_1), R'(y_1,y_2), R(y_3,y_2), T(y_4,y_3)\}$ in the picture above. A universal tree witness for (x_0,y_1) must contain $S(x_0,y_1)$. As the role in $R'(y_1,y_2)$ is forward, the universal tree witness must contain this atom too. The role R in $R^-(y_2,y_3)$ is also forward, so we add this atom and identify y_1 with y_3. This gives three approximations of the universal tree witness we are constructing:



In $T(y_4,y_3)$, the role T is twisty in $\mathcal{T}$. The canonical model $\mathcal{C}_\mathcal{T}$ suggests two alternative ways of treating $T(y_4,y_3)$: either to add it as it is, or to identify y_4 with root x_0. Thus, no universal tree witness for (x_0,y_1) exists. This is also signalled by the fact that condition **(perf)** fails: by identifying y_1 and y_3, we make S and T adjacent, while both $suc(S,T)$ and $inv(S,T)$ hold in $\mathcal{C}_\mathcal{T}$. Note that S and T were not adjacent in the original query $\boldsymbol{q}$; that is why **(perf)** is checked for the image $f(\boldsymbol{q})$ of every tree witness f. As the number of tree witnesses can be exponential, this complication might suggest that **(perf)** cannot be checked efficiently. The proof

of Lemma 16 shows that this check can be done in polynomial time without constructing all tree witnesses.

Strictly speaking, a universal tree witness is not a tree witness in the sense of our original definition, but rather a convenient structure representing all tree witnesses for (t, t'), even if there are exponentially many of them. Universal tree witnesses can be used to further simplify the rewriting $\boldsymbol{q}_c$. Namely, all formulas tw_f for (t, t') are merged into a single tree-witness formula $\mathsf{tw}_{\boldsymbol{f}}$ for the universal tree witness $\boldsymbol{f}$ for (t, t'), which is defined by taking:

$$\mathsf{tw}_{\boldsymbol{f}} = \Big[\bigvee_{\substack{\exists R.B \text{ such that there is} \\ \text{a homomorphism } h\colon \boldsymbol{f} \to \mathcal{C}_{\mathcal{T}} \\ h(t)=a_{RB} \\ h(t')=b_{RB}}} \mathsf{ext}_{\exists R.B}(t) \Big] \wedge \bigwedge_{s \text{ is root of } \boldsymbol{f}} \big((t = s) \wedge \bigwedge_{A(s) \in \boldsymbol{q}} A(s) \big).$$

As a universal tree witness is unique for each pair (t, t') of adjacent terms, let us denote it by $\boldsymbol{q}_{(t,t')}$. We are now in a position to define our *polynomial* rewriting $\boldsymbol{q}_p$ for $\boldsymbol{q}$ and $\mathcal{T}$:

$$\boldsymbol{q}_p(\vec{x}) \;=\; \mathsf{detached}_{\boldsymbol{q}} \vee$$
$$\exists \vec{y} \bigwedge_{\substack{\{t,t'\} \\ t \text{ and } t' \\ \text{adjacent}}} \Big[\big(\bigwedge_{\substack{A(s) \in \boldsymbol{q} \\ s \in \{t,t'\}}} \mathsf{ext}_A(s) \wedge \bigwedge_{R(t,t') \in \boldsymbol{q}} \mathsf{ext}_R(t,t') \big) \vee \bigvee_{\substack{s \text{ and } s' \text{ adjacent} \\ t,t' \in \mathsf{term}(\boldsymbol{q}_{(s,s')})}} \mathsf{tw}_{\boldsymbol{q}_{(s,s')}} \Big].$$

Theorem 19 *If all tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ are perfect and condition* **(conf)** *is satisfied then, for any ABox $\mathcal{A}$ and any $\vec{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}_p[\vec{a}]$. Moreover, $\boldsymbol{q}_p$ is constructed in time polynomial in $|\boldsymbol{q}|$, $|\mathcal{T}|$.*

Theorem 19 can be substantially sharpened by taking account of the 'types' of terms in $\boldsymbol{q}$.

Example 20 Consider again the TBox $\mathcal{T}_2$ from Example 14 and the CQ $\boldsymbol{q}_1(x) = \{T(x, y), S(y, z), A''(z)\}$. This time we have only one tree witness for (x, y). It is not perfect; yet, it is type-perfect because the 'typed' *suc* formula $\exists x, y, z \, \big(T(x, y) \wedge S(y, z) \wedge A''(z) \wedge (x \neq z)\big)$ does not hold in $\mathcal{C}_{\mathcal{T}_2}$.

If an ontology $\mathcal{T}$ does not contain any twisty roles, then all tree witnesses in any CQ $\boldsymbol{q}$ over $\mathcal{T}$ are clearly perfect. On the other hand, all examples of conflicting tree witnesses given above involve twisty roles. The following theorem shows that this is no accident:

Theorem 21 *Suppose that $\boldsymbol{q}$ is a CQ and $\mathcal{T}$ an OWL 2 QL ontology without twisty roles. Then there are no conflicting tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$. Thus, the rewriting $\boldsymbol{q}_p$ for $\boldsymbol{q}$ and $\mathcal{T}$ is correct and can be constructed in polynomial time.*

It may be worth noting that *OWL 2 EL* (Baader, Brandt, and Lutz 2005; 2008) ontologies satisfy this sufficient condition, and so a polynomial rewriting similar to $\boldsymbol{q}_p$ can also be used for conjunctive query answering over such ontologies provided that the *ABoxes are complete* (or saturated) *with respect to the ontologies*.

Experiments

To understand how the rewriting $\boldsymbol{q}_p$ looks like in real-world practice, we have run experiments with three known

		QuOnto size of UCQ	Requiem size of UCQ	Presto size of Datalog	Nyaya size of UCQ	tw/ utw	ext rules	non ext rules
A	Q1	783	402	69	249	8/1	53	3
	Q2	1,812	103	52	94	1	30	3
	Q3	4,763	104	55	104	0	30	1
	Q4	7,251	492	93	456	4/1	42	3
	Q5	78,885	624	71	624	0	36	1
U	Q1	5	2	6	2	0	2	1
	Q2	287	148	1	1	0	0	1
	Q3	1,260	224	8	4	0	4	1
	Q4	5,364	1,628	6	2	0	2	1
	Q5	9,245	2,960	11	10	0	7	1
	Q6	588	42	19	26	1	16	5
	Q7	5,950	630	37	39	2	16	7
S	Q1	6	6	7	6	0	6	1
	Q2	204	160	3	2	0	2	1
	Q3	1,194	480	5	4	0	4	1
	Q4	1,632	960	5	4	0	4	1
	Q5	11,487	2,880	7	8	0	6	1
	Q6	588	564	12	36	1	24	3
	Q7	118	106	14	29	1	19	5

OWL 2 QL ontologies—Adolena (A), University (U) and Stockexchange (S)—using the same conjunctive queries as in (Pérez-Urbina, Motik, and Horrocks 2009; Rosati and Almatelli 2010; Gottlob, Orsi, and Pieris 2011) and two new ones (Q6 and Q7); the ontologies and queries are available at `www.dcs.bbk.ac.uk/~roman/query-rewriting`. The results of the experiments are collected in the table above.

The most important conclusion we can draw from these experiments is that, in practice, queries and ontologies generate very few tree witnesses, if any; they are never in conflict with each other, the sufficient condition of Theorem 19 is satisfied, and so the rewritings $\boldsymbol{q}_c$ and $\boldsymbol{q}_p$ are both short and correct.

To describe how the rewritten queries look like and explain the figures in the table, assume first that a given CQ $\boldsymbol{q}$ contains no tree witnesses with respect to a given ontology $\mathcal{T}$. In this case, $\boldsymbol{q}_p$ is obtained from $\boldsymbol{q}$ by replacing each unary atom $A(s)$ with $\mathsf{ext}_A(s)$ and each binary atom $R(s, s')$ with $\mathsf{ext}_R(s, s')$. Roughly, the ext_E contain those concepts/roles that are located under E in the classification of $\mathcal{T}$ by an ontology reasoner. It will be convenient for us to represent the ext_E as nonrecursive Datalog programs. For example, Adolena gives the rules:

$$\begin{aligned} \mathsf{ext}_{\mathtt{affects}}(x, y) &\;\text{:-}\; \mathtt{affects}(x, y). \\ \mathsf{ext}_{\mathtt{affects}}(x, y) &\;\text{:-}\; \mathtt{isAffectedBy}(y, x). \\ \mathsf{ext}_{\mathtt{Device}}(x) &\;\text{:-}\; \mathtt{Device}(x). \\ &\cdots \end{aligned}$$

The column 'ext rules' in the table refers to the number of such Datalog rules for the given CQ and ontology.

Consider now a query containing tree witnesses, for instance the following query Q4 over Adolena:

$$\boldsymbol{q}(x) = \exists y \, \big(\mathtt{Device}(x) \wedge \mathtt{assistsWith}(x, y) \wedge \mathtt{PhysicalAbility}(x, y) \big).$$

This CQ contains a tree witness f with root

$$f(x) = a_{\mathtt{assistsWith},\mathtt{MovementAbility}}.$$

In this case, for each pair (s, s') of adjacent terms in $\boldsymbol{q}$, we introduce a fresh predicate, say $\mathsf{edge}_{s,s'}$, and define it by means of a nonrecursive Datalog program such as

$$\begin{aligned}\mathsf{edge}_{x,y}(x,y) \;&:\!-\; \mathsf{ext}_{\texttt{Device}}(x), \mathsf{ext}_{\texttt{assistsWith}}(x,y),\\ &\qquad \mathsf{ext}_{\texttt{PhysicalAbility}}(x,y).\\ \mathsf{edge}_{x,y}(x,y) \;&:\!-\; \mathsf{ext}_{\texttt{Device}}(x),\\ &\qquad \mathsf{ext}_{\exists\texttt{assistsWith.MovementAbility}}(x).\end{aligned}$$

where the first rule corresponds to choosing both x and y among the ABox individuals and the second rule comes from a single universal tree witness for (x, y). Then we replace by $\mathsf{edge}_{s,s'}$ all the atoms in the CQ that are 'covered' by the terms s and s'. In our running example, we obtain

$$\boldsymbol{q}(x) \;:\!-\; \mathsf{edge}_{x,y}(x,y).$$

The column 'non-ext rules' in the table refers to the number of such rules (one rule for the whole $\boldsymbol{q}$ plus the rules defining the $\mathsf{edge}_{s,s'}$); 'tw' gives the number of tree witnesses in the CQ, and 'utw' the number of universal tree witnesses.

In theory, a correct rewriting of a CQ $\boldsymbol{q}$ and an *OWL 2 QL* ontology $\mathcal{T}$ can be exponential only if $\boldsymbol{q}$ and $\mathcal{T}$ give rise to exponentially many tree witnesses, in which case the canonical model $\mathcal{C}_{\mathcal{T}}$ must be extremely complex. Our experiments indicate that, in practice, the contribution of tree witnesses does not look essential at all, especially in comparison with the contribution of the definitions of ext_E, which reflects the depth and width of the concept and role hierarchies in $\mathcal{T}$ rather than the complexity of $\mathcal{C}_{\mathcal{T}}$. Note also that the same predicates ext_E are used in all queries, which makes these predicates an ideal target for optimisations. The ways to minimise the influence of these rules depend on how we store the data.

There are two main approaches to storing data in OBDA. Suppose first that an ABox $\mathcal{A}$ is stored in a local database and the system has a certain degree of control over the data. In this case, one can saturate $\mathcal{A}$ with the intensional data that is implied by the TBox axioms (more precisely, construct the ABox part of the canonical model). Having done so, we do not need the ext_E predicates any more and can replace them with the corresponding E. The ABox saturation (more precisely, a finite encoding of the whole canonical model) was suggested in the combined approach (Lutz, Toman, and Wolter 2009; Kontchakov et al. 2010). However, the downside of the ABox saturation is a significant increase of the storage space required for the data (and a slowdown of updates). One solution to this problem was found by Rodriguez Muro and Calvanese (2011a; 2011b). In a nutshell, the idea is to build a 'semantic index' by assigning numerical identifiers to the concept names, used in the ontology, in such a way that all subclasses of a given concept are associated with an interval (or a few intervals) of numbers. The semantic index allows one to encode the definitions of any of the ext_E predicates as a single query that selects all instances with concept identifiers falling into the respective interval(s). In this case, the classical database indexing techniques are employed to ensure efficiency of these interval queries.

In the other typical OBDA scenario, ABoxes do not come as sets of triples stored in a single database. Instead, the sets of individuals that belong to concepts and roles are defined by means of queries (mappings) to a number of (relational) data sources (Lenzerini 2002; Calvanese et al. 2007b). Consider, for instance, the rules for the role `affects` above. One can clearly expect mappings to be defined in such a way that $\texttt{affects}(a, b) \in \mathcal{A}$ iff $\texttt{isAffectedBy}(b, a) \in \mathcal{A}$ in every ABox $\mathcal{A}$. This suggests that, in fact, there is no need for two separate rules, so that the predicate $\mathsf{ext}_{\texttt{affects}}$ can be eliminated altogether (replaced by $\texttt{affects}(x, y)$, which halves the number of CQs produced). It is also quite feasible that information about all devices is stored in a single database relation and mappings for each of the 26 subclasses of the concept `Device` select appropriate devices from the same database relation. In such a case, every ABox $\mathcal{A}$ defined by these mappings will be *complete* for all subclasses A of `Device` in the sense that $A(a) \in \mathcal{A}$ iff $(\mathcal{T}, \mathcal{A}) \models A(a)$. Therefore, with this information at hand, one can replace the 26 rules defining $\mathsf{ext}_{\texttt{Device}}(x)$ with just a single rule $\mathsf{ext}_{\texttt{Device}}(x) \;:\!-\; \texttt{Device}(x)$, or even eliminate the predicate $\mathsf{ext}_{\texttt{Device}}(x)$ altogether.

Conclusions

In this paper, we considered pure (positive existential) rewritings of conjunctive queries over *OWL 2 QL* ontologies and analysed why such rewritings can be lengthy. We showed that the length of a rewriting is related to the number of tree witnesses in the query, which reflect how various parts of the query can be homomorphically mapped to the tree ('intensional') part of the canonical model. Thus, a rewriting can be lengthy if the original query is sufficiently long and the intensional part of the canonical model for the ontology is sufficiently complex. We proved that by restricting the interaction between inverse roles and role inclusion axioms in ontologies and queries, we can guarantee transparent polynomial rewritings. Moreover, we also demonstrated that real-world ontologies and queries contain very few tree witnesses, satisfy the above mentioned restrictions, and so enjoy polynomial rewritings.

Remark 22 When the final version of this paper was ready for submission, we obtained some new results that shed more light on the size of pure rewritings. Below is a brief summary of these results; for details consult the preliminary report (Kikot, Kontchakov, Podolskii and Zakharyaschev 2012).

(1) An exponential blow-up is unavoidable for pure positive existential rewritings (PE) and pure nonrecursive Datalog (NDL) rewritings; pure FO-rewritings can blow-up superpolynomially unless NP $\subseteq$ P/poly.

(2) Pure NDL-rewritings are in general exponentially more succinct than pure PE-rewritings.

(3) Pure FO-rewritings can be superpolynomially more succinct than pure PE-rewritings.

(4) Impure PE-rewritings can always be made polynomial, and so they are exponentially more succinct than pure PE-rewritings.

(1)–(3) are proved by first establishing connections between pure rewritings for CQs over *OWL 2 QL* ontologies and circuits for monotone Boolean functions, and then using known

lower bounds and separation results for the circuit complexity of such functions as CLIQUE(n, k) 'a graph with n nodes contains a k-clique' and MATCHING$(2n)$ 'a bipartite graph with n vertices in each part has a perfect matching' (Razborov 1985; Borodin, von zur Gathen, and Hopcroft 1982; Raz and Wigderson 1992; Raz and McKenzie 1997)

The polynomial PE-rewriting in (4) is similar to the NDL-rewriting of Gottlob and Schwentick (2011): using two extra constants, $=$ and (polynomially-many) new existentially quantified variables, one can encode a relevant part of the canonical model of $\mathcal{T}$ in the rewritten query. The difference between the resulting impure PE-rewritings and the exponential-size pure PE-rewritings is of the same kind as the difference between deterministic and nondeterministic Boolean circuits. As shown by Razborov (1985), no polynomial-size deterministic monotone circuit can compute CLIQUE(n, k); however, it can be computed by a polynomial-size nondeterministic circuit (or a QBF), where the existentially quantified variables guess k vertices and the circuit checks whether they form a k-clique in the given graph. In the polynomial impure PE-rewriting (4), the extra constants, variables and $=$ are used to nondeterministically guess a part of the canonical model into which the query can be mapped. (We conjecture that there does not exist an impure polynomial-size rewriting if the number of new existentially quantified variables is bounded.)

Acknowledgments

The work on this paper was supported by the U.K. EPSRC grant EP/H05099X/1.

We are grateful to G. Gottlob, G. Orsi, R. Rosati and D. Tsarkov who helped us to run the experiments.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI-05*, 364–369. Professional Book Center.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the EL envelope further. In Clark, K., and Patel-Schneider, P. F., eds., *Proc. of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.

Borodin, A.; von zur Gathen, J.; and Hopcroft, J. E. 1982. Fast parallel matrix and GCD computations. In *Proc. of the 23rd Annual Symp. on Foundations of Computer Science, FOCS'82*, 65–71. IEEE Computer Society.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007a. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning* 39(3):385–429.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; and Rosati, R. 2007b. Ontology-based database access. In *Proc. of the 15th Ital. Conf. on Database Systems, SEBD 2007*, 324–331.

Chortaras, A.; Trivela, D.; and Stamou, G. 2011. Goal-oriented query rewriting for OWL 2 QL. In *Proc. of the 24th Int. Workshop on Description Logics, DL 2011*, vol. 745 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Gottlob, G., and Schwentick, T. 2011. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of the 24th Int. Workshop on Description Logics, DL 2011*, vol. 745 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Gottlob, G.; Orsi, G.; and Pieris, A. 2011. Ontological queries: Rewriting and optimization. In *Proc. of the the 27th Int. Conf. on Data Engineering, ICDE 2011*, 2–13. IEEE Computer Society.

Kikot, S.; Kontchakov, R.; and Zakharyaschev, M. 2011. On (In)Tractability of OBDA with OWL 2 QL. In *Proc. of the 24th Int. Workshop on Description Logics, DL 2011*, vol. 745 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kikot, S.; Kontchakov, R.; Podolskii, V.; and Zakharyaschev, M. 2012. Exponential lower bounds and separation for query rewriting. *CoRR*, arXiv:1202.4193, 2012.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2010. The combined approach to query answering in DL-Lite. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010*. AAAI Press.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2011. The combined approach to ontology-based data access. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, IJCAI-2011*, 2656–2661. AAAI Press.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, PODS 2002*, 233–246.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic EL using a relational database system. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009*, 2070–2075. AAAI Press.

Pérez-Urbina, H.; Motik, B.; and Horrocks, I. 2009. A comparison of query rewriting techniques for DL-Lite. In *Int. Workshop on Description Logics, DL 2009*, vol. 477 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Raz, R., and McKenzie, P. 1997. Separation of the monotone nc hierarchy. In *Proc. of the 38th Annual Symp. on Foundations of Computer Science, FOCS'97*, 234–243. IEEE Computer Society.

Raz, R., and Wigderson, A. 1992. Monotone circuits for matching require linear depth. *J. ACM* 39(3):736–744.

Razborov, A. 1985. Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* 281(4):798–801.

Rodriguez Muro, M., and Calvanese, D. 2011a. Dependencies to optimize ontology based data access. In *Proc.*

of the 24th Int. Workshop on Description Logics, DL 2011, vol. 745 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rodriguez Muro, M., and Calvanese, D. 2011b. Semantic index: Scalable query answering without forward chaining or exponential rewritings. In *Proc. of the 10th Int. Semantic Web Conf., ISWC 2011*.

Rosati, R., and Almatelli, A. 2010. Improving query answering over DL-Lite ontologies. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010*. AAAI Press.