Polynomial Conjunctive Query Rewriting under Unary Inclusion Dependencies

S. Kikot, R. Kontchakov, and M. Zakharyaschev

Department of Computer Science and Information Systems,
Birkbeck College, London, U.K.
{kikot,roman,michael}@dcs.bbk.ac.uk

Abstract. Ontology-based data access (OBDA) is widely accepted as an important ingredient of the new generation of information systems. In the OBDA paradigm, potentially incomplete relational data is enriched by means of ontologies, representing intensional knowledge of the application domain. We consider the problem of conjunctive query answering in OBDA. Certain ontology languages have been identified as FO-rewritable (e.g., *DL-Lite* and sticky-join sets of TGDs), which means that the ontology can be incorporated into the user's query, thus reducing OBDA to standard relational query evaluation. However, all known query rewriting techniques produce queries that are exponentially large in the size of the user's query, which can be a serious issue for standard relational database engines. In this paper, we present a polynomial query rewriting for conjunctive queries under unary inclusion dependencies. On the other hand, we show that binary inclusion dependencies do not admit polynomial query rewriting algorithms.

1 Introduction

Ontology-based data access (OBDA) [9, 12, 18] has recently emerged as an important ingredient of the new generation of information systems. OBDA is required in those cases where data, stored in relational databases, is regarded as potentially incomplete, and so is supplemented by ontologies describing the background knowledge of the application domain. In logical terms, OBDA involves the following reasoning problem:

QA(D, Σ, q)**:** given a database instance D, an ontology Σ and a query $q(\boldsymbol{x})$, find all certain answers $\boldsymbol{a}$ to $q(\boldsymbol{x})$ in D under Σ.

In other words, the task is to check, given a tuple $\boldsymbol{a}$, whether $\mathcal{I} \models q(\boldsymbol{a})$ for every model $\mathcal{I}$ of (D, Σ). The OBDA paradigm can be viable in practice only if its efficiency is comparable with the efficiency of standard database query evaluation, where *data complexity* was identified as a proper efficiency measure [22]. This idea lies behind the *DL-Lite* family of description logics, designed with the aim of OBDA [7, 8], and the *DL-Lite*-based *OWL 2 QL* profile of the *OWL 2* Web Ontology Language. Its distinctive feature is that 'in *OWL 2 QL*, conjunctive query answering can be implemented using conventional relational database

systems. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in LogSpace with respect to the size of the data' (`www.w3.org/TR/owl2-profiles`). In fact, in *OWL 2 QL*, conjunctive query answering is in AC^0 for data complexity, and this problem is *first-order rewritable* in the sense that $\mathrm{QA}(D, \Sigma, q)$ can be reduced to the query evaluation problem $\mathrm{QE}(D, q')$, where the query q' does not depend on D.

Recently, other families of languages generalising *DL-Lite* and featuring first-order rewritability have emerged: linear Datalog± [5], sticky-join sets of TGDs (tuple-generating dependencies) [6], atomic-hypothesis and domain-restricted $\forall\exists$-rules [4]. From the complexity-theoretic point of view, all these languages are perfectly suitable for OBDA. On the more practical side, there is a number of different query rewriting algorithms, which have been implemented in such systems as QuOnto [1, 19], REQUIEM [17], Presto [21] and Nyaya [10]. However, in all of these algorithms, the size of the rewritten query q', posed to the database system, can be $\mathcal{O}((|\Sigma| \cdot |q|)^{|q|})$ in the worst case.

In this paper, we try (1) to clarify whether the exponential blow-up in the size of the rewritten query is inevitable, and (2) to identify languages for which polynomial rewritings are possible. We concentrate on the language of inclusion dependencies, covering *DL-Lite* and *OWL 2 QL*. In Section 3, we give a polynomial rewriting of conjunctive queries under *unary* inclusion dependencies. This result improves on the polynomial rewriting from [13], which reduces $\mathrm{QA}(D, \Sigma, q)$ to $\mathrm{QE}(D + aux, q')$, where aux is a set of fresh constants encoding the canonical model of (D, Σ). Note also the recent polynomial reduction [11] of $\mathrm{QA}(D, \Sigma, q)$ to $\mathrm{QE}(D + \{0, 1\}, q'')$, which uses two fresh constants 0, 1 and works for the extension Datalog± of *OWL 2 QL* (see Remark 1). In Section 4, we show that unary inclusion dependencies are near the border separating the languages with polynomial rewritings by proving that no polynomial-time algorithm can produce a query rewriting under binary inclusion dependencies.

2 Preliminaries

A *relational schema* $\mathcal{R}$ is a set of relational symbols (predicates), each associated with its arity. The ontology language we consider in this paper consists of *inclusion dependencies*, which can be thought of as *sentences* (in the first-order language of similarity type $\mathcal{R}$) of the form

$$\forall \boldsymbol{x} \big(\exists \boldsymbol{y}\, P_1(u_1, \dots, u_n) \to \exists \boldsymbol{z}\, P_2(v_1, \dots, v_m) \big),$$

where P_1 is an n-ary predicate in $\mathcal{R}$, P_2 an m-ary predicate in $\mathcal{R}$, all the variables u_i (and v_i) are pairwise distinct and $\boldsymbol{x} = \{u_1, \dots, u_n\} \setminus \boldsymbol{y} = \{v_1, \dots, v_m\} \setminus \boldsymbol{z}$. An inclusion dependency is called *unary* if $\boldsymbol{x}$ is a singleton. A *database instance* D is a set of ground atoms. We use the standard model-theoretic notions and notation. Thus, if $\mathcal{I}$ is a first-order structure, Σ a set of inclusion dependencies and D a database instance then $\mathcal{I} \models (D, \Sigma)$ means that all formulas in D and Σ are true in $\mathcal{I}$; in this case $\mathcal{I}$ is called a *model* of (D, Σ).

A *conjunctive query* (CQ) $q(\boldsymbol{x})$ is a first-order formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of variables and φ is constructed, using only $\wedge$, from atoms of the form $P(t_1, \dots, t_n)$, with P being an n-ary predicate and t_i a *term* (an individual name or variable from $\boldsymbol{x}$ or $\boldsymbol{y}$). Given a database instance D, we use $\mathsf{Ind}(D)$ to denote the set of constants in D. A tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(D)$ is a *certain answer* to $q(\boldsymbol{x})$ over (D, Σ) if $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of (D, Σ); in this case we write $(D, \Sigma) \models q[\boldsymbol{a}]$. To simplify notation, we will often identify q with the set of its atoms; $\mathsf{term}(q)$ is the set of terms in q.

We are interested in data-independent reductions of the query answering problem $\mathrm{QA}(D, \Sigma, q)$, where q is a CQ and Σ a finite set of unary inclusion dependencies, to the query evaluation problem $\mathrm{QE}(D, q')$, where q' is a (not necessarily conjunctive) query over D. More precisely, $q'(\boldsymbol{x})$ is an *FO-rewriting of* $q(\boldsymbol{x})$ *under* Σ if, for any database instance D and any tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(D)$, we have $(D, \Sigma) \models q[\boldsymbol{a}]$ iff $D \models q'[\boldsymbol{a}]$.

3 Polynomial Rewriting

Let Σ be a set of unary inclusion dependencies. Each dependency in Σ can be regarded as a formula of the form

$$\forall x \, \big(\pi_j P_1(u_1, \dots, u_n) \to \pi_k P_2(v_1, \dots, v_m)\big),$$

where $x = u_j = v_k$, and π_j and π_k denote projections onto the j-th and k-th argument, respectively:

$$\pi_j P_1(u_1, \dots, u_n) = \exists u_1, \dots, u_{j-1}, u_{j+1}, \dots, u_n \, P_1(u_1, \dots, u_n),$$
$$\pi_k P_2(v_1, \dots, u_m) = \exists v_1, \dots, v_{k-1}, v_{k+1}, \dots, v_m \, P_2(v_1, \dots, v_m).$$

First we observe that answering conjunctive queries under unary inclusion dependencies can be polynomially reduced to the case where the language does not contain predicates of arity greater than 2. Indeed, for each n-ary predicate R with $n > 2$, we can introduce n binary predicates $R_1, \dots, R_n$, replace each $\pi_k R(x_1, \dots, x_n)$ in Σ with $\exists y \, R_k(x_k, y)$ and add the following unary inclusion dependencies:

$$\forall y \, \big(\exists x_i \, R_i(x_i, y) \to \exists x_j \, R_j(x_j, y)\big), \qquad 1 \leq i \neq j \leq n.$$

Let Σ' be the resulting set of unary inclusion dependencies. We also need to modify D accordingly: for each atom $R(a_1, \dots, a_n) \in D$, take a fresh constant $a_{R(a_1,\dots,a_n)}$, replace the atom with $R_i(a_i, a_{R(a_1,\dots,a_n)})$, for $1 \leq i \leq n$, and denote the result by D'. Finally, given a conjunctive query q, let q' be the result of replacing each $R(t_1, \dots, t_n)$ in q with the conjunction $\exists y \, (\bigwedge_{i=1}^{n} R_i(t_i, y))$. It is not hard to see that $(D, \Sigma) \models q[\boldsymbol{a}]$ iff $(D', \Sigma') \models q'[\boldsymbol{a}]$, for any $\boldsymbol{a} \subseteq \mathsf{Ind}(D)$.

To simplify presentation, from now on we only deal with unary inclusion dependencies over unary and binary relations, i.e., inclusion dependencies

$$\forall x \, (\psi_1(x) \to \psi_2(x)),$$

where ψ_1 and ψ_2 are formulas of the form $A(x)$, $\exists y\, P(x,y)$ or $\exists y\, P(y,x)$. To make notation more convenient, we often denote $P(y,x)$ (in both queries and dependencies) by $P^-(x,y)$. We will call P and P^- the two *directions* of binary predicate P.

3.1 Canonical Model (Chase)

As inclusion dependencies are in essence Horn clauses, for any (D,Σ), there is a structure $\mathcal{U}_{D,\Sigma}$ such that, for all conjunctive queries $q(\boldsymbol{x})$ and $\boldsymbol{a} \subseteq \mathsf{Ind}(D)$, we have $(D,\Sigma) \models q[\boldsymbol{a}]$ iff $\mathcal{U}_{D,\Sigma} \models q[\boldsymbol{a}]$. The structure $\mathcal{U}_{D,\Sigma}$, called the *canonical model* (*chase*) of (D,Σ), is constructed as follows. For each direction of a binary predicate P in $\mathcal{R}$, we introduce fresh symbols c_P and c_{P^-}, and call them the *witnesses* for $\exists y\, P(x,y)$ and $\exists y\, P^-(x,y)$, respectively. (In the case of k-ary R, one would have to consider k distinct witnesses $c_{R,i}$, for $1 \leq i \leq k$.) A *path generated by* $a \in \mathsf{Ind}(D)$ *under* Σ is a finite sequence $ac_{R_1} \cdots c_{R_n}$, $n \geq 0$, such that

$$a \rightsquigarrow_{D,\Sigma} c_{R_1} \rightsquigarrow_{D,\Sigma} \cdots \rightsquigarrow_{D,\Sigma} c_{R_n},$$

where $\rightsquigarrow_{D,\Sigma}$ is the *generating relation* defined as follows, for $1 \leq i < n$:

($\mathbf{C}_0$) $a \rightsquigarrow_{D,\Sigma} c_{R_1}$ if $(D,\Sigma) \models \exists y\, R_1(a,y)$ and $D \not\models \exists y\, R_1(a,y)$;
($\mathbf{C}_1$) $c_{R_i} \rightsquigarrow_{D,\Sigma} c_{R_{i+1}}$ if $\Sigma \models \forall x\, (\exists y\, R_i^-(x,y) \to \exists y\, R_{i+1}(x,y))$ and $R_i^- \neq R_{i+1}$.

(A path $ac_{R_1} \cdots c_{R_n}$ generated by a can be thought of as a labelled null constructed from a by the sequence $c_{R_1}, \ldots, c_{R_n}$.) Denote by $\mathsf{path}_\Sigma(a)$ the set of all paths generated by $a \in \mathsf{Ind}(D)$ under Σ, and by $\mathsf{tail}(\sigma)$ the last element in such a path σ. Now, the canonical model $\mathcal{U}_{D,\Sigma}$ with domain $\Delta^{\mathcal{U}_{D,\Sigma}}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{U}_{D,\Sigma}} &= \bigcup\nolimits_{a \in \mathsf{Ind}(D)} \mathsf{path}_\Sigma(a),\\
a^{\mathcal{U}_{D,\Sigma}} &= a, \text{ for all } a \in \mathsf{Ind}(D),\\
A^{\mathcal{U}_{D,\Sigma}} &= \{a \in \mathsf{Ind}(D) \mid (D,\Sigma) \models A(a)\} \cup\\
&\quad \{\sigma \mid \mathsf{tail}(\sigma) = c_R \text{ and } \Sigma \models \forall x\, (\exists y\, R^-(x,y) \to A(x))\},\\
P^{\mathcal{U}_{D,\Sigma}} &= \{(a,b) \in \mathsf{Ind}(D) \times \mathsf{Ind}(D) \mid (D,\Sigma) \models P(a,b)\} \cup\\
&\quad \{(\sigma, \sigma \cdot c_P) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{D,\Sigma} c_P, \} \cup \{(\sigma \cdot c_{P^-}, \sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{D,\Sigma} c_{P^-}\}.
\end{aligned}$$

3.2 Tree Witnesses

Let $\mathsf{C}_\mathcal{R}$ be the set of all witnesses c_P and c_{P^-} for binary predicates P in $\mathcal{R}$, and $\mathsf{C}_\mathcal{R}^*$ the set of all finite words over $\mathsf{C}_\mathcal{R}$ (including the empty word ε). We use $\mathsf{tail}(\sigma)$ to denote the last element of $\sigma \in \mathsf{C}_\mathcal{R}^* \setminus \{\varepsilon\}$; by definition, $\mathsf{tail}(\varepsilon) = \varepsilon$.

Consider a conjunctive query $q(\boldsymbol{x})$. Without loss of generality, we will assume that (the primal graph of) q is connected. Let R be a direction P or P^- for a binary predicate P in $\mathcal{R}$ and t a term in q. A *partial* function f from $\mathsf{term}(q)$ to $\mathsf{C}_\mathcal{R}^*$ is called a *tree witness for* (R,t) *in* q (cf. [13]) if the following conditions hold:

– $f(t) = \varepsilon$,
– for all atoms $S(s, s') \in q$ with $f(s)$ defined, we have

$$f(s') = \begin{cases} c_R, & \text{if } f(s) = \varepsilon \text{ and } S = R, \\ \sigma, & \text{if } f(s) = \sigma \cdot c_{S^-}, \\ f(s) \cdot c_S, & \text{if } f(s) \neq \varepsilon \text{ and } \mathsf{tail}(f(s)) \neq c_{S^-}, \end{cases}$$

– the domain of f is minimal with respect to set-theoretic inclusion.

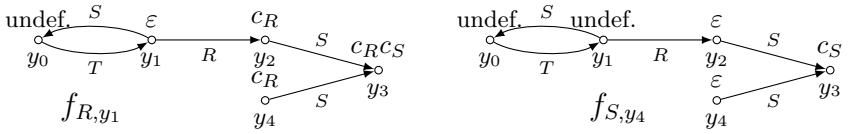
By definition, if a tree witness for (R, t) exists then it is unique; in this case we denote it by $f_{R,t}$ and use $\mathsf{dom}\, f_{R,t}$ for the domain of $f_{R,t}$. Note that even if q contains no atom of the form $R(t, t')$, the tree witness for (R, t) exists and $f_{R,t}(t) = \varepsilon$. Denote by $q|_{R,t}$ the set of atoms of q whose terms are in $\mathsf{dom}\, f_{R,t}$. If $q|_{R,t}$ is regarded as a *query*, we assume that all of its variables are *free*.

Informally, a tree witness $f_{R,t}$ has *root* t and *direction* R, and describes the situation when t is mapped to a database instance constant a such that the type of a in the canonical model requires an R-successor but the database instance does not provide it; cf. $(\mathbf{C}_0)$. In this case, the only choice for mapping every t' in $R(t, t') \in q$ is $ac_R = a \cdot f_{R,t}(t')$. Further, every t'' in $S(t', t'') \in q$ has to be mapped to $ac_R c_S = a \cdot f_{R,t}(t'')$ provided that $S \neq R^-$; however, if $S = R^-$ then t'' can only be mapped to a, which reflects the fact that ac_R has a single R^--successor a in the canonical model.

Example 1. To illustrate, consider the CQ

$$q \;=\; \{T(y_0, y_1),\ S(y_1, y_0),\ R(y_1, y_2),\ S(y_2, y_3),\ S(y_4, y_3)\}.$$

The tree witnesses for (R, y_1) and (S, y_4) in q exist and are as depicted below:

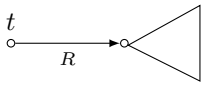


On the other hand, for (S, y_1) and (T^-, y_1), tree witnesses do not exist (because y_1 is part of the cycle).

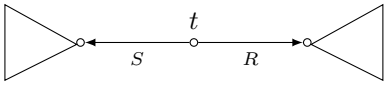
If a term s is such that $f_{R,t}(s)$ is defined and $f_{R,t}(s) \neq \varepsilon$, then a tree witness exists for s and any direction S for which $\mathsf{tail}(f_{R,t}(s)) \neq c_{S^-}$; moreover, the tree witness for (S, s) is 'included' in the tree witness for (R, t). In the example above, the existence of a tree witness for (R, y_1) implies that a tree witness for (S, y_2) exists since $f_{R,y_1}(y_2) = c_R \neq c_{S^-}$; however, one cannot guarantee that a tree witness exists for (R^-, y_2); and in fact, it does not exist since the cycle is reachable in the direction R^-. On the other hand, if $f_{R,t}(s) = \varepsilon$ then s can also play role of root for the same direction R, in which case the tree witnesses for (R, t) and (R, s) coincide, e.g., f_{S,y_2} and f_{S,y_4} in the example above. Thus, we obtain:

Proposition 1. *Suppose a tree witness for (R, t) exists and $s \in \mathsf{dom}\, f_{R,t}$. If $f_{R,t}(s) \neq \varepsilon$ then a tree witness exists for every (S, s) with $\mathsf{tail}(f_{R,t}(s)) \neq c_{S^-}$. If $f_{R,t}(s) = \varepsilon$ then a tree witness exists for (S, s) with $S = R$. In either case, $\mathsf{dom}\, f_{S,s} \subseteq \mathsf{dom}\, f_{R,t}$ and $f_{R,t}(s') = f_{R,t}(s) \cdot f_{S,s}(s')$, for all $s' \in \mathsf{dom}\, f_{S,s}$.*

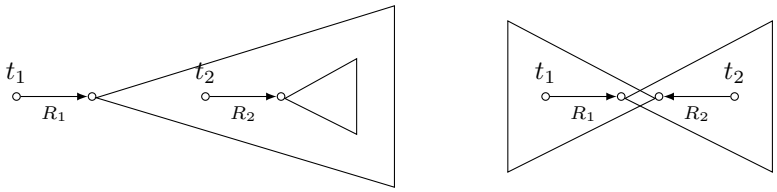
Even if a tree witness for (R, t) exists, $q|_{R,t}$ is not necessarily a tree-shaped query. Define a relation $\equiv_R$ as the set of all pairs (t, s) such that a tree witness for (R, t) exists and $f_{R,t}(s) = \varepsilon$. By Proposition 1, $\equiv_R$ is an equivalence relation (on its domain). By taking the quotient of $q|_{R,t}$ under $\equiv_R$, we obtain a tree reduct of $q|_{R,t}$ (cf. [15]), which can be depicted as follows:



We call q a *quasi-tree with root* $t \in \mathsf{term}(q)$ if a tree witness for (R, t) exists for *all* directions R and $\bigcup_R \mathsf{dom}\, f_{R,t} = \mathsf{term}(q)$:



If there are two tree witnesses with a common term that is not a root for both of them then either one is a sub-tree of the other or they are part of the same quasi-tree:



Proposition 2. *Suppose q is not a quasi-tree and tree witnesses exist for (R_1, t_1) and (R_2, t_2). If $f_{R_1,t_1}(t_2)$ is defined, $f_{R_1,t_1}(t_2) \neq \varepsilon$ then $\mathsf{dom}\, f_{R_2,t_2} \subsetneq \mathsf{dom}\, f_{R_1,t_1}$ and $f_{R_1,t_1}(s) = f_{R_1,t_1}(t_2) \cdot f_{R_2,t_2}(s)$, for all $s \in \mathsf{dom}\, f_{R_2,t_2}$.*

Proof. Suppose first that $\mathsf{tail}(f_{R_1,t_1}(t_2)) = c_{R_2^-}$. Then, by Proposition 1, tree witnesses exists for (S, t_2) in all directions S, and so q is a quasi-tree with root t_2, contrary to our assumption. So, $\mathsf{tail}(f_{R_1,t_1}(t_2)) \neq c_{R_2^-}$ and, by Proposition 1, $f_{R_1,t_1}(s) = f_{R_1,t_1}(t_2) \cdot f_{R_2,t_2}(s)$, for $s \in \mathsf{dom}\, f_{R_2,t_2}$. Since $f_{R_1,t_1}(t_2) \neq \varepsilon$, $\mathsf{dom}\, f_{R_2,t_2} \subsetneq \mathsf{dom}\, f_{R_1,t_1}$.

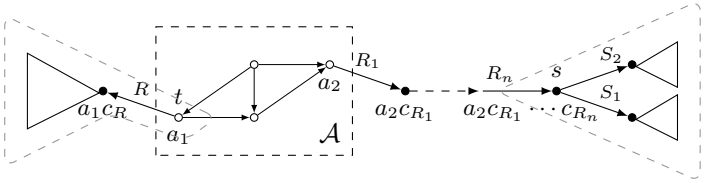
3.3 Query Rewriting

We are now in a position to introduce the ingredients of our polynomial rewriting. Fix a relational schema $\mathcal{R}$. Consider (D, Σ) and $q(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$. We can easily compute answers to queries of the form $A(x)$ or $\exists y\, R(x, y)$:

Proposition 3. *For all formulas $\psi(x)$ of the form $A(x)$ or $\exists y\, R(x,y)$ and all $a \in \mathsf{Ind}(D)$, we have $\mathcal{U}_{D,\Sigma} \models \psi(a)$ iff $D \models \mathsf{ext}_\psi(a)$, where*

$$\mathsf{ext}_\psi(x) \quad = \bigvee_{\substack{\forall x\,(\psi'(x) \to \psi(x)) \text{ is a unary incl. dep. in } \mathcal{R} \\ \Sigma \models \forall x\,(\psi'(x) \to \psi(x))}} \psi'(x).$$

Note that, for all other elements σ in the domain $\Delta^{\mathcal{U}_{D,\Sigma}}$ of the canonical model $\mathcal{U}_{D,\Sigma}$, we have $\mathcal{U}_{D,\Sigma} \models \psi[\sigma]$ iff $\Sigma \models \forall x\,(\exists y\, T^-(x,y) \to \psi(x))$, where $\mathsf{tail}(\sigma) = c_T$.



Consider now a binary atom $R(t,t') \in q$ and the ways its terms can be mapped in the canonical model $\mathcal{U}_{D,\Sigma}$.

1. If both t and t' are mapped to database constants $a, a' \in \mathsf{Ind}(D)$ then we have $\mathcal{U}_{D,\Sigma} \models R(a,a')$ iff $R(a,a') \in D$ because the canonical model $\mathcal{U}_{D,\Sigma}$ inherits the binary relations between constants from D.

2. If t is mapped to a constant $a \in \mathsf{Ind}(D)$ and t' to a labelled null in $\Delta^{\mathcal{U}_{D,\Sigma}} \setminus \mathsf{Ind}(D)$, then $R(t,t')$ can only be true if (i) $a \rightsquigarrow_{D,\Sigma} c_R$, (ii) a tree witness for (R,t) exists, and (iii) $q|_{R,t}$ can be embedded into the sub-tree of $\mathsf{path}_\Sigma(a)$ beginning with the edge (a, ac_R); see the left-hand side of the picture above. For condition (i) we have the following:

Proposition 4. *For all directions R and all $a \in \mathsf{Ind}(D)$, we have $a \rightsquigarrow_{D,\Sigma} c_R$ iff $D \models \mathsf{wt}_R(a)$, where*

$$\mathsf{wt}_R(x) \;=\; \mathsf{ext}_{\exists y\, R(x,y)}(x) \wedge \neg \exists w\, R(x,w).$$

For condition (iii), consider the conjunction $\mathsf{treeA}^q_{R,t}(x)$ of the formulas:

- $(\mathbf{t}_0)$ $\mathsf{ext}_A(x)$, for all $A(s) \in q|_{R,t}$ with $f_{R,t}(s) = \varepsilon$;
- $(\mathbf{t}_1)$ $\top$ if $\Sigma \models \forall x\,(\exists y\, T^-(x,y) \to A(x))$ and $\bot$ otherwise, for all $A(s) \in q|_{R,t}$ with $\mathsf{tail}(f_{R,t}(s)) = c_T$;
- $(\mathbf{t}_2)$ $\top$ if $\Sigma \models \forall x\,(\exists y\, T^-(x,y) \to \exists y\, S(x,y))$ and $\bot$ otherwise, for all $S(s,s') \in q|_{R,t}$ with $\mathsf{tail}(f_{R,t}(s)) = c_T$.

The following lemma states that the formula $\mathsf{treeA}^q_{R,t}(x)$ precisely describes this situation for the database instance constants:

Lemma 1. *(i) If a tree witness for (R,t) exists and $D \models \mathsf{wt}_R(a) \wedge \mathsf{treeA}^q_{R,t}(a)$, for $a \in \mathsf{Ind}(D)$, then $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{a}} q|_{R,t}$ holds for an assignment $\mathfrak{a}$ such that $\mathfrak{a}(s) = a \cdot f_{R,t}(s)$, for all $s \in \mathsf{dom}\, f_{R,t}$.*

(ii) If $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{a}} q$ and there is $R(t,t') \in q$ with $\mathfrak{a}(t) = a \in \mathsf{Ind}(D)$ and $\mathfrak{a}(t') = a \cdot c_R$, then a tree witness for (R,t) exists and $D \models \mathsf{wt}_R(a) \wedge \mathsf{treeA}^q_{R,t}(a)$.

3. If both t, t' are mapped to anonymous elements in $\Delta^{\mathcal{U}_{D,\Sigma}} \setminus \mathsf{Ind}(D)$, then two more cases need consideration.

3.1. Suppose first that there is a tree witness for some (S, s) such that s is mapped to a database instance constant $a \in \mathsf{Ind}(D)$ with $a \rightsquigarrow_{D,\Sigma} c_S$, (iv) both t and t' are in $\mathsf{dom}\, f_{S,s}$, and (v) all the terms $s' \in \mathsf{dom}\, f_{S,s}$ with $f_{S,s}(s') \neq \varepsilon$ are existentially quantified variables in q (only existential variables can be mapped to labelled nulls). In this case, by Lemma 1, $R(t, t')$ is true in $\mathcal{U}_{D,\Sigma}$ if the formula

$$\mathsf{wt}_S(s) \land \mathsf{treeA}^q_{S,s}(s) \land \bigwedge\nolimits_{s \equiv_S s'} (s = s')$$

is true in D under an assignment $\mathfrak{a}$ such that $\mathfrak{a}(s') = a \cdot f_{S,s}(s')$, for $s' \in \mathsf{dom}\, f_{S,s}$. The disjunction of all such formulas for (S, s) satisfying (iv)–(v) depends only on the choice of terms t, t' and will be denoted by $\mathsf{attached\text{-}tree}_{t,t'}(\boldsymbol{x}, \boldsymbol{y})$. (This case is a generalisation of Case 2.)

3.2. Thus, it remains to consider the case (shown in the right-hand side of the picture) when the whole query is mapped to the tree-shaped part of $\mathcal{U}_{D,\Sigma}$ consisting of labelled nulls. Then q a quasi-tree and all terms in q are existentially quantified variables that are mapped to the sub-tree $\mathsf{path}_\Sigma(a)$ of $\mathcal{U}_{D,\Sigma}$ generated by some $a \in \mathsf{Ind}(D)$. More precisely, a generates a sequence of the form $a \rightsquigarrow_{D,\Sigma} c_{R_1} \rightsquigarrow_{D,\Sigma} \cdots \rightsquigarrow_{D,\Sigma} c_{R_n}$, q has a root s (i.e., $\mathsf{term}(q) = \bigcup_S \mathsf{dom}\, f_{S,s}$), s is mapped to $\sigma = a c_{R_1} \cdots c_{R_n}$, while all other terms s' are mapped to $\sigma \cdot f_{S,s}(s')$. The latter condition can be captured by a formula similar to the one in Case 3.1. The difference is that now we begin with a labelled null σ with $\mathsf{tail}(\sigma) = c_{R_n}$ (rather than a database constant a). To cope with this, consider the union q' of q and $\{R_n(v, s)\}$, for a fresh variable v, and let $\mathsf{treeT}^q_{c_{R_n},s}$ be $\mathsf{treeA}^{q'}_{R_n,v}$, where the tree witnesses are computed in the query q'. Note that $\mathsf{treeT}^q_{c_{R_n},s}$ is a sentence because q' has no atoms for item $(\mathbf{t}_0)$. We denote by $\mathsf{detached\text{-}tree}$ the disjunction of sentences

$$\exists w\, \mathsf{wt}_{R_1}(w) \land \mathsf{treeT}^q_{c_{R_n},s}$$

for all roots s of q and all pairs of directions R_1, R_n such that there are directions $R_2, \dots, R_{n-1}$ with $\Sigma \models \forall x\, (\exists y\, R_i^-(x, y) \to \exists y\, R_{i+1}(x, y))$ and $R_i^- \neq R_{i+1}$, for $1 \leq i < n$; cf. $(\mathbf{C}_1)$; if q is not a quasi-tree containing only existentially quantified variables, we set $\mathsf{detached\text{-}tree} = \bot$. The following lemma is an analogue of Lemma 1:

Lemma 2. *Let $q = \exists \boldsymbol{y}\, \varphi(\boldsymbol{y})$.*

(i) If $D \models \mathsf{detached\text{-}tree}$ then there is $\sigma \in \Delta^{\mathcal{U}_{D,\Sigma}}$ such that $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{a}} \varphi$ for an assignment $\mathfrak{a}$ with $\mathfrak{a}(s') = \sigma \cdot f_{S,s}(s')$, for all directions S and $s' \in \mathsf{dom}\, f_{S,s}$.

(ii) If $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{a}} \varphi$ for an assignment $\mathfrak{a}$ such that $\mathfrak{a}(t) \notin \mathsf{Ind}(D)$, for all terms $t \in \mathsf{term}(q)$, then q is a tree-shaped query and $D \models \mathsf{detached\text{-}tree}$.

Denote by q^* the result of replacing each atom $A(t)$ and $P(t, t')$ in q with

$$A^*(t) = \mathsf{ext}_{A(x)}(t) \lor \mathsf{attached\text{-}tree}_{t,t}(\boldsymbol{x}, \boldsymbol{y}) \lor \mathsf{detached\text{-}tree},$$
$$P^*(t, t') = P(t, t') \lor \mathsf{attached\text{-}tree}_{t,t'}(\boldsymbol{x}, \boldsymbol{y}) \lor \mathsf{detached\text{-}tree},$$

respectively. Note that these formulas depend not only on the predicate name but also on the terms in the atom. The length of q^* is $\mathcal{O}(|q|^2 \cdot |\mathcal{T}|^3)$ and can be made $\mathcal{O}(|q|^2 \cdot |\mathcal{T}|)$ if the sentence $\mathsf{detached\text{-}tree}$ is computed separately (in fact, for the majority of queries, e.g., queries with answer variables, it is simply $\bot$).

Theorem 1. *Let $q(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$ be a connected query. Then $\mathcal{U}_{D,\Sigma} \models q[\boldsymbol{a}]$ iff $D \models q^*[\boldsymbol{a}]$, for all $\boldsymbol{a} \subseteq \mathsf{Ind}(D)$.*

Proof. ($\Rightarrow$) If $\mathcal{U}_{D,\Sigma} \models q[\boldsymbol{a}]$ then there is an assignment $\mathfrak{b}$ such that $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{b}} \varphi$ and $\mathfrak{b}(x_i) = a_i \in \mathsf{Ind}(D)$, for all answer variables x_i. If $\mathfrak{b}(y) \in \mathsf{Ind}(D)$ for all variables y, then clearly $D \models q^*[\boldsymbol{a}]$. Otherwise, for each y with $\mathfrak{b}(y) \notin \mathsf{Ind}(D)$, one can trace (via binary atoms) the sequence $\sigma = \mathfrak{b}(y)$ back either to $a = \mathfrak{b}(y_0) \in \mathsf{Ind}(D)$ or to the shortest sequence $\sigma' = \mathfrak{b}(y_0) \in \Delta^{\mathcal{U}_{D,\Sigma}} \setminus \mathsf{Ind}(D)$. In the latter case, by Lemma 2 (*ii*), q is a tree-shaped query and $D \models \mathsf{detached\text{-}tree}$, whence $D \models^{\mathfrak{b}} \varphi$. In the former case, by Lemma 1 (*ii*), there is a tree witness for (S, y_0), where S is such that $\sigma = a c_S \dots$, and $D \models^{\mathfrak{b}} \mathsf{attached\text{-}tree}_{t,t'}$ for all $t, t' \in \mathsf{dom}\, f_{S,y_0}$. Thus, $D \models^{\mathfrak{b}} P^*(t, t')$ and $D \models^{\mathfrak{b}} A^*(t)$, for all $P(t, t')$ and $A(t)$ in $q|_{S,y_0}$. By repeating this procedure for each variable y with $\mathfrak{b}(y) \notin \mathsf{Ind}(D)$, we can clearly cover all atoms in q, and therefore, $D \models q^*[\boldsymbol{a}]$.

($\Leftarrow$) Let $D \models q^*[\boldsymbol{a}]$. Then there is an assignment $\mathfrak{b}$ in D such that $D \models^{\mathfrak{b}} \varphi^*$ and $\mathfrak{b}(x_i) = a_i \in \mathsf{Ind}(D)$, for all answer variables x_i, where φ^* is the quantifier-free part of q^*.

If $D \models \mathsf{detached\text{-}tree}$ then q has no answer variables or constants and, by Lemma 2, $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{b}} \varphi$, for some assignment $\mathfrak{b}$. Thus, $\mathcal{U}_{D,\Sigma} \models q$. So, for the rest of the proof we assume $D \not\models \mathsf{detached\text{-}tree}$.

We claim that we can choose tree witnesses $f_{S_1,s_1}, \dots f_{S_n,s_n}$ such that their domains without roots are pairwise disjoint, i.e.,

$$\big(\mathsf{dom}\, f_{S_i,s_i} \setminus \{s \mid s \equiv_{S_i} s_i\}\big) \cap \big(\mathsf{dom}\, f_{S_j,s_j} \setminus \{s \mid s \equiv_{S_j} s_j\}\big) = \emptyset, \qquad \text{for all } i \neq j$$

and if $D \models^{\mathfrak{b}} \mathsf{attached\text{-}tree}_{t,t'}$ then $t, t' \in \mathsf{dom}\, f_{S_i,s_i}$, for some $1 \leq i \leq n$, in which case

$$D \models^{\mathfrak{b}} \mathsf{treeA}^q_{S_i,s_i}(s_i) \wedge \bigwedge\nolimits_{s_i \equiv_S s'} (s_i = s');$$

Indeed, by Proposition 2, the domains of any two tree witnesses either do not intersect (but on the roots) or coincide or the domain of one is subsumed by the domain of other. Consider the assignment $\mathfrak{b}'$ in $\mathcal{U}_{D,\Sigma}$ given by

$$\mathfrak{b}'(v) = \begin{cases} \mathfrak{b}(v) \cdot f_{S_i,s_i}(v), & \text{if } v \in \mathsf{dom}\, f_{S_i,s_i} \text{ for } 1 \leq i \leq n, \\ \mathfrak{b}(v), & \text{otherwise.} \end{cases}$$

Consider now terms t, t'. If $D \models \mathsf{attached\text{-}tree}_{t,t'}$ then, by Lemma 1 (*i*), we have $\mathcal{U}_{D,\Sigma} \models^{\mathfrak{b}'} q|_{S_i,s_i}$, for $1 \leq i \leq n$ with $t, t' \in \mathsf{dom}\, f_{S_i,s_i}$. Otherwise, both t and t' are interpreted by database instance constants and so the atom containing them is true in $\mathcal{U}_{D,\Sigma}$ by Proposition 3.

Example 2. Consider the CQ from Example 1:

$$q \;=\; \exists y_0 y_1 y_2 y_3 y_4 \big(T(y_0, y_1) \land S(y_1, y_0) \land R(y_1, y_2) \land S(y_2, y_3) \land S(y_4, y_3)\big).$$

As q contains a cycle, it is not a quasi-tree, and so $\mathsf{detached\text{-}tree} = \bot$. As both y_0 and y_1 are part of the cycle, $\mathsf{attached\text{-}tree}_{y_0,y_1} = \bot$, whence

$$T^*(y_0, y_1) = T(y_0, y_1) \quad \text{and} \quad S^*(y_1, y_0) = S(y_1, y_0).$$

Consider $R(y_1, y_2)$: the only existing tree witness that has both t_1 and t_2 is f_{R,y_1}, all others either do not include one of the terms or would reach the cycle. So, if $\Sigma \models \forall x\, (\exists y\, R^-(x, y) \to \exists y\, S(x, y))$ then $\mathsf{attached\text{-}tree}_{y_1,y_2} = \mathsf{wt}_R(y_1)$, otherwise it is $\bot$.

Both $S(y_2, y_3)$ and $S(y_4, y_3)$ are considered similarly. In either case there are three tree witnesses containing the terms: f_{R,y_1}, f_{R,y_2} and f_{R,y_4}. The last two in fact coincide and give the disjunct $\mathsf{wt}_S(y_2) \land (y_2 = y_4)$ to $\mathsf{attached\text{-}tree}_{y_2,y_3}$. And similarly to the previous case, if $\Sigma \models \forall x\, (\exists y\, R^-(x, y) \to \exists y\, S(x, y))$ then $\mathsf{attached\text{-}tree}_{y_2,y_3}$ contains another disjunct $\mathsf{wt}_R(y_1)$.

To sum up: if $\Sigma \models \forall x\, (\exists y\, R^-(x, y) \to \exists y\, S(x, y))$ then

$$\begin{aligned} &R^*(y_1, y_2) = R(y_1, y_2) \lor \mathsf{wt}_R(y_1), \\ &S^*(y_k, y_3) = S(y_k, y_3) \lor \mathsf{wt}_R(y_1) \lor (\mathsf{wt}_S(y_2) \land (y_2 = y_4)), \text{ for } k = 2, 4; \end{aligned}$$

otherwise,

$$\begin{aligned} &R^*(y_1, y_2) = R(y_1, y_2), \\ &S^*(y_k, y_3) = S(y_k, y_3) \lor (\mathsf{wt}_S(y_2) \land (y_2 = y_4)), \text{ for } k = 2, 4. \end{aligned}$$

Example 3. Consider now the CQ

$$q \;=\; \exists y_1 y_2 y_3 y_4 \big(R(y_1, y_2) \land S(y_2, y_3) \land S(y_4, y_3)\big).$$

This query is a quasi-tree with three roots: y_1, y_2 and y_4. Suppose that

$$\begin{aligned} &\Sigma \models \forall x\, (\exists y\, T^-(x, y) \to \exists y\, R^-(x, y)), \\ &\Sigma \models \forall x\, (\exists y\, T^-(x, y) \to \exists y\, S(x, y)). \end{aligned}$$

Then $\mathsf{detached\text{-}tree}$ will contain a disjunct $\exists w\, \mathsf{wt}_T(w)$ for the root y_2 (if there is no T as above then there will be no such disjunct). Sentence $\mathsf{detached\text{-}tree}$ will contain similar disjuncts for the other two roots, y_1 and y_4 (provided that Σ entails inclusion dependencies required in each case).

Let us consider all tree witnesses and their contribution to the formulas $\mathsf{attached\text{-}tree}_{t,t'}$:

- f_{R,y_1} provides $\mathsf{wt}_R(y_1)$ only if $\Sigma \models \forall x\, (\exists y\, R^-(x, y) \to \exists y\, S(x, y))$;
- f_{R^-,y_2} provides $\mathsf{wt}_{R^-}(y_2)$;
- f_{S^-,y_3} provides $\mathsf{wt}_{S^-}(y_3)$ only if $\Sigma \models \forall x\, (\exists y\, S(x, y) \to \exists y\, R^-(x, y))$;
- f_{S,y_2} and f_{S,y_4} provide $\mathsf{wt}_S(y_2) \land (y_2 = y_4)$.

For $R(y_1, y_2)$ we have to take three tree witnesses f_{R,y_1}, f_{R^-,y_2} and f_{S^-,y_3}; and for both $S(y_2, y_3)$ and $S(y_4, y_3)$ we have to take four tree witnesses f_{R,y_1}, f_{S^-,y_3}, f_{S,y_2} and f_{S,y_4}.

So, if the additional unary dependencies mentioned above do not follow from Σ then we have

$$q^* = \exists \boldsymbol{y} \big[\big(R(y_1, y_2) \lor \mathsf{wt}_{R^-}(y_2) \big) \land \big(S(y_2, y_3) \lor (\mathsf{wt}_S(y_2) \land (y_2 = y_4)) \big) \land \big(S(y_4, y_3) \lor (\mathsf{wt}_S(y_4) \land (y_2 = y_4)) \big) \big].$$

If $\Sigma \models \forall x \, (\exists y \, R^-(x, y) \to \exists y \, S(x, y))$ then $R^*(y_1, y_2)$, $S^*(y_2, y_3)$ and $S^*(y_4, y_3)$ will contain the extra disjunct $\mathsf{wt}_R(y_1)$; if $\Sigma \models \forall x \, (\exists y \, S(x, y) \to \exists y \, R^-(x, y))$ they all will contain $\mathsf{wt}_{S^-}(y_3)$.

The examples above show that the rewritings often contain duplicating subformulas: for instance, in Example 3, if $\Sigma \models \forall x \, (\exists y \, R^-(x, y) \to \exists y \, S(x, y))$ then all conjuncts of the rewriting will contain a disjunct $\mathsf{wt}_R(y_1)$, and so, the rewriting is equivalent to the union of a shorter query and $\exists y_1 \, \mathsf{wt}_R(y_1)$. These considerations suggest that some simple optimisations can significantly reduce the size of the rewritings.

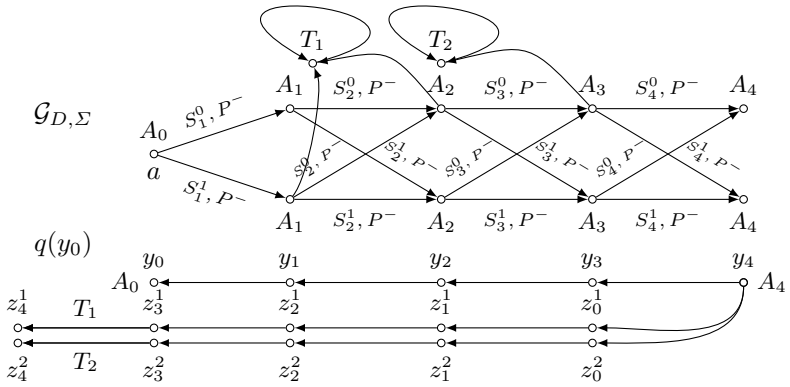
4 No Polynomial Rewriting Algorithm under Binary Inclusion Dependencies

Theorem 2. *Unless* P = NP*, no polynomial-time algorithm can reduce conjunctive query answering under binary inclusion dependencies to the problem of evaluating queries* (*independently of the database instance*).

Proof. The result follows from NP-hardness of query answering under binary inclusion dependencies with a singleton database instance, which is shown by reduction of Boolean satisfiability. Given a CNF $\varphi = \bigwedge_{j=1}^{m} D_j$ over variables $p_1, \dots, p_n$, where D_j is a clause, we consider the following set Σ of inclusion dependencies, for $1 \le i \le n$, $1 \le j \le m$, $k = 0, 1$:

$$\forall x \, (A_{i-1}(x) \to \exists y \, S_i^k(x, y)), \qquad \forall x \, (\exists y \, S_i^k(y, x) \to A_i(x)),$$
$$\forall xy \, (S_i^k(y, x) \to P(x, y)),$$
$$\forall x \, (\exists y \, S_i^0(x, y) \to \exists y \, T_j(x, y)) \quad \text{if } \neg p_i \in D_j,$$
$$\forall x \, (\exists y \, S_i^1(x, y) \to \exists y \, T_j(x, y)) \quad \text{if } p_i \in D_j,$$
$$\forall x \, (\exists y \, T_j^-(x, y) \to \exists y \, T_j(x, y)), \qquad \forall xy \, (T_j(x, y) \to P(x, y)).$$

The canonical model $\mathcal{U}_{D,\Sigma}$ of $D = \{A_0(a)\}$ is obtained by unravelling the 'generating' interpretation $\mathcal{G}_{D,\Sigma}$ shown below:



Consider the CQ $q(y_0)$, also depicted above:

$$q(y_0) \;=\; \exists y_1,\dots,y_n\ \exists z_0^1,\dots,z_n^1,\dots,z_0^m,\dots,z_n^m \\ A_0(y_0) \wedge \bigwedge_{i=1}^{n} P(y_i,y_{i-1}) \wedge A_n(y_n) \wedge \\ \bigwedge_{j=1}^{m}\Big(P(y_n,z_0^j) \wedge \bigwedge_{i=1}^{n} P(z_{i-1}^j,z_i^j) \wedge T_j(z_{n-1}^j,z_n^j)\Big).$$

(Note n atoms P connecting y_n to y_0, but $n+1$ atoms P connecting y_n to z_n^j, which means that any match of q in $\mathcal{U}_{D,\Sigma}$ must map z_n^j onto a point in the infinite chain of T_j-edges.) One can show now that φ is satisfiable iff $(D,\Sigma) \models q(a)$.

Let us analyse the above proof. As query rewriting has to be independent of database instances, one can always choose a simplest possible case for D: say, a singleton $D = \{A(a)\}$, for some unary relation A. Without any dependencies, query answering over such a D can clearly be done in polynomial time. It follows from Theorem 1 that, under unary inclusion dependencies, conjunctive queries to D can also be answered in polynomial time. But then the proof of Theorem 2 shows that, under binary inclusion dependencies, this is not the case anymore. So, the combined complexity of answering conjunctive queries over a singleton database instance of the form $\{A(a)\}$ provides an indication whether polynomial query rewriting algorithms exist or not. We will call this measure—that is, the combined complexity of the problem $\mathrm{QA}(\{A(a)\},\Sigma,q)$—the *primitive combined complexity* of query answering. It follows that if conjunctive query answering is NP-hard for primitive combined complexity, then no algorithm is capable of constructing rewritings in polynomial time (unless P = NP).

Remark 1. If the database instance is extended with fresh constants 0 and 1 then $q(y_0)$ in the proof above can be rewritten as

$$A_0(y_0) \quad \wedge \quad \exists p_1 \dots \exists p_n \,\big(\bigwedge_{i=1}^{n}(p_i \neq y_0) \wedge \bigwedge_{j=1}^{m} D'_j\big),$$

where D'_j is obtained from D_j by replacing every literal p_i with $p_i = 1$ and every $\neg p_i$ with $p_i = 0$. Moreover, using $\forall p_i$, one can polynomially encode the PSPACE-complete validity problem for QBFs. A polynomial reduction of the query answering problem $\mathrm{QA}(D, \Sigma, q)$ to the query evaluation problem $\mathrm{QE}(D + \{0, 1\}, q')$ is given in [11] for Datalog±, where $|\mathcal{T}| + |q|$ steps of the chase are simulated using 0 and 1.

5 Discussion

We conclude with some general (and possibly controversial) remarks on OBDA.

First-order rewritability (or AC^0 data complexity) does not seem to provide enough information to judge whether an ontology language is suitable for OBDA. When measuring the complexity of query evaluation in database systems, it is usually assumed that queries are negligibly small compared to data. Thus, it makes sense to consider data complexity [22], which takes account of the data but ignores the query. In OBDA, the rewritten queries can no longer be assumed to be small. However, data complexity does not differentiate among, e.g., unary inclusion dependencies, *OWL 2 QL* or the language of sticky-join sets of TGDs [6], query answering in all of which is in AC^0 for data complexity, while the primitive combined complexity, reflecting the size of the rewriting (see Section 4), ranges from P to NP and further to EXPTIME. Another explanation of database efficiency is that we only use queries with a bounded number of variables, in which case query evaluation is P-complete for combined complexity [23]. However, query rewritings may substantially increase the number of variables (for example, a CQ q is rewritten in [11] into a query with $\mathcal{O}(N \cdot \log N)$ auxiliary binary variables, where $N = |T| + |q|$).

The W3C recommendation (`www.w3.org/TR/owl2-profiles`) for OBDA is to reduce it to answering queries in database systems. Two drawbacks of this recommendation are that it (*i*) disregards the complexity of possible reductions, and (*ii*) excludes some useful ontology languages from consideration. As we saw above, rewritings of CQs in *OWL 2 QL* cannot be done in polynomial time without extra constants, variables and quantifiers as in [11]. One might argue that the complex constellation used in the proof of Theorem 2 does not occur in real-world ontologies, but then more research is needed to support this argument. A number of 'lightweight' ontology languages such as $\mathcal{EL}$ [3] or $\mathit{DL\text{-}Lite}_{horn}^{(\mathcal{HF})}$ (without the UNA) [2] are deemed not suitable for OBDA because query answering is P-complete for data complexity in them (note, however, that it is P-complete for primitive combined complexity in both languages compared to NP in the case of *OWL 2 QL*). The combined approach to OBDA [16, 14] resolves this issue by expanding the data at a pre-processing step and then rewriting and answering CQs. The expansion is linear in the size of the database and can be done by the database system itself; the size of the rewritten query for $\mathcal{EL}$ and $\mathit{DL\text{-}Lite}_{horn}^{\vec{\mathcal{F}}}$ is only quadratic [16, 13] (for *OWL 2 QL*, it is still exponential).

In this paper, we do not touch on the problem of connecting ontologies to databases, which is typically done in OBDA by means of GLAV mappings. Such

mappings introduce additional problems as tuples in the same relation can come from different data sources. Also, they provide certain information on the completeness of database predicates, which can (and should) be exploited in order to minimise the rewritings [20]. Finally, with so many languages and rewritings for OBDA suggested, it looks like the time is ripe for comprehensive experiments that could clarify the future of OBDA.

References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI). pp. 1670–1671 (2005)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC) (2008)
4. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, É.: Extending decidable cases for rules with existential variables. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI). pp. 677–682 (2009)
5. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of the 28th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS). pp. 77–86. ACM (2009)
6. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/−. In: Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR). LNCS, vol. 6333, pp. 1–17. Springer (2010)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: *DL-Lite*: Tractable description logics for ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI). pp. 602–607 (2005)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)
9. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Proc. of the 7th Int. Semantic Web Conf. (ISWC 2008). LNCS, vol. 5318, pp. 403–418. Springer (2008)
10. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the IEEE Int. Conf. on Data Engineering (ICDE) (2011)
11. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive Datalog programs. In: Proc. of the 24th Int. Workshop on Description Logics (DL). (2011)
12. Heymans, S., Ma, L., Anicic, D., Ma, Z., Steinmetz, N., Pan, Y., Mei, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Feier, C., Hench, G., Wetzstein, B., Keller, U.: Ontology reasoning with large data repositories. In: Ontology Management, pp. 89–128. Springer (2008)
13. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: 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)

14. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI). (2011)
15. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR). pp. 179–193. LNAI, vol. 5195, Springer (2008)
16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI). pp. 2070–2075 (2009)
17. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: Proc. of the 22nd Int. Workshop on Description Logics (DL). CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
18. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
19. Poggi, A., Rodriguez, M., Ruzzi, M.: Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In: Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC). (2008)
20. Rodríguez-Muro M., Calvanese, D.: Dependencies to optimize ontology-based data access. In: Proc. of the 24th Int. Workshop on Description Logics (DL). (2011)
21. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR). (2010)
22. Vardi, M.: The complexity of relational query languages (extended abstract). In: Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC). pp. 137–146 (1982)
23. Vardi, M.: On the complexity of bounded-variable queries (extended abstract). In: Proc. of the 14th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS). pp. 266–276 (1995)