Tractability Guarantees for DL-Lite Query Answering

Meghyn Bienvenu[1], Magdalena Ortiz[2], Mantas Šimkus[2], and Guohui Xiao[2]

[1] Laboratoire de Recherche en Informatique, CNRS & Université Paris Sud, Orsay, France
[2] Institute of Information Systems, Vienna University of Technology, Vienna, Austria

Abstract. It is a classic result in database theory that conjunctive query (CQ) answering, which is NP-complete in general, is feasible in polynomial time when restricted to acyclic queries. Subsequent results identified more general structural properties of CQs (like bounded treewidth) which ensure tractable query evaluation. In this paper, we lift these tractability results to knowledge bases formulated in the core dialect of DL-Lite. The proof exploits known properties of query matches in this logic and involves a query-dependent modification of the data. To obtain a more practical approach, we next propose a concrete polynomial-time algorithm for answering acyclic CQs based upon a rewriting into datalog. We also show how the algorithm can be extended to a larger class of (nearly acyclic) CQs. A preliminary evaluation of our proof-of-concept implementation suggests the interest of our approach for handling large acyclic CQs.

1 Introduction

Description logics of the DL-Lite family [3] have become the languages of choice for ontology-based data access, in no small part due to the very low data complexity of conjunctive query (CQ) answering over knowledge bases formulated in these logics. While low data complexity is rightfully considered important for scalability, it does not by itself guarantee efficient query answering in practice. Indeed, most existing query answering algorithms for DL-Lite proceed by rewriting the input CQ into a UCQ which is then posed to the database. The query rewriting step typically results in a considerably larger query, which can make this approach infeasible for queries of moderate size. Even when the query rewriting phase is not too costly, the evaluation of the CQs over the database can be challenging when the database is very large, since the problem is known to be NP-complete in combined complexity, and all known algorithms require exponential time in the size of the query.

The NP-hardness of CQ answering in relational databases motivated the search for classes of CQs which admit efficient evaluation. A well-known result in database theory shows that CQ answering becomes feasible in polynomial time when restricted to the class of *acyclic CQs* [21]. Later investigations lead to the identification of more general structural properties, such as bounded treewidth, query width, or hypertree width [4, 7], which guarantee tractable CQ answering. Since the NP-hardness of CQ answering in DLs is inherited from relational databases, it is natural to ask whether these tractability results also transfer to the DL setting. This would be very desirable since it is likely that most of the queries that will actually occur in applications are acyclic. While there are no collections of real-world CQs that can be used to support this claim in the DL

setting, one can find some compelling evidence by looking at the closely related setting of SPARQL queries over RDF data, where it has been reported that acyclic queries (in fact, acyclic conjunctive graph patterns) comprise more than 99% of the queries in a log of around three million queries posed to the DBpedia endpoint [16]. Unfortunately, lifting positive results from databases to the DL setting is often not possible, even for DL-Lite. For instance, for the logic DL-Lite$_\mathcal{R}$, which underlies the QL profile of the OWL 2 standard [14], CQ answering was recently shown to be NP-hard already for acyclic queries [9].

In this paper, we show that for plain DL-Lite (without role hierarchies) the picture is brighter: all polynomial-time upper bounds for classes of CQs known from relational databases carry over to DL-Lite. Although this general tractability result relies on known properties of the logic, to our knowledge, it has not been pointed out before. The proof reduces the problem of answering a CQ over a knowledge base $\mathcal{K}$ to answering the same CQ over a database that results from a polynomial expansion of the dataset in $\mathcal{K}$. The algorithm arising from this reduction has a disadvantage: it involves a query-dependent expansion of the data. For this reason, we propose an alternative algorithm based on a rewriting into non-recursive datalog, which runs in polynomial time for acyclic CQs. We also show how this approach can be extended to a larger family of CQs that are almost acyclic. We have implemented a simple prototype of the rewriting algorithm, and it shows promising results for answering large acyclic CQs.

The general tractability result and rewriting algorithm for acyclic CQs are presented in [2]; the extension to nearly acyclic queries is new. Due to space restrictions, we consider only DL-Lite in this paper and invite the interested reader to consult [2] to see how our results can be adapted to $\mathcal{ELH}$ ontologies [1].

2 Preliminaries

Description Logics We briefly recall the syntax and semantics of DL-Lite and its extension DL-Lite$_\mathcal{R}$ [3]. Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite sets of concept names, role names, and individuals, respectively, and let $\overline{\mathsf{N_R}} = \mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$ be the set of *(complex) roles*. For $R \in \overline{\mathsf{N_R}}$, R^- denotes r^- if $R = r \in \mathsf{N_R}$, and r if $R = r^-$.

An *ABox* is a finite set of *assertions* of the forms $A(b)$ and $r(b, c)$ with $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $b, c \in \mathsf{N_I}$. A *TBox* is a finite set of *axioms*, which in DL-Lite are *concept inclusions* of the form $B_1 \sqsubseteq (\neg)B_2$, with B_1, B_2 of the form $A \in \mathsf{N_C}$ or $\exists R$ with $R \in \overline{\mathsf{N_R}}$. In DL-Lite$_\mathcal{R}$, TBoxes may additionally contain *role inclusions* of the form $R_1 \sqsubseteq (\neg)R_2$ with $R_1, R_2 \in \overline{\mathsf{N_R}}$. A *knowledge base (KB)* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and ABox $\mathcal{A}$.

The semantics of KBs is defined in terms of *(DL) interpretations* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ in the usual way, see [3] for details. $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies every axiom in $\mathcal{T}$ and assertion in $\mathcal{A}$, and we call $\mathcal{K}$ *consistent* if it admits some model. We use $\mathsf{Ind}(\mathcal{A})$ for the individuals occurring in $\mathcal{A}$, and let $\mathcal{I}_\mathcal{A}$ be the interpretation with $\Delta^\mathcal{I} = \mathsf{Ind}(\mathcal{A})$ such that (i) $c \in A^\mathcal{I}$ iff $A(c) \in \mathcal{A}$, and (ii) $(c, d) \in r^\mathcal{I}$ iff $r(c, d) \in \mathcal{A}$.

Queries We first recall *non-recursive datalog queries* [12]. Let $\mathsf{N_V}$ and $\mathsf{N_D}$ be countably infinite sets of *variables* and *datalog relations*, respectively. Each $\sigma \in \mathsf{N_D}$ has an associated non-negative integer *arity*. *Atoms* are expressions of the form $p(\boldsymbol{x})$, where $\boldsymbol{x} \in (\mathsf{N_V})^n$, and (i) $p \in \mathsf{N_C}$ and $n = 1$, (ii) $p \in \overline{\mathsf{N_R}}$ and $n = 2$, or (iii) $p \in \mathsf{N_D}$ and n is

the arity of p; atoms of forms (i) and (ii) are called *DL-atoms*. A *rule* ρ is an expression of the form $h(\boldsymbol{x}) \leftarrow \alpha_1, \ldots, \alpha_m$, where $h(\boldsymbol{x}), \alpha_1, \ldots, \alpha_m$ are atoms, h is a datalog relation, and every variable of $\boldsymbol{x}$ occurs in $body(\rho) = \{\alpha_1, \ldots, \alpha_m\}$. Abusing notation, we write $\alpha \in \rho$ instead of $\alpha \in body(\rho)$. The variables in $head(\rho)$ are called the *answer variables* of ρ. Given a set of rules P, we let $Dep(P) = (V, E)$ be the directed graph such that: (a) V is the set of all datalog relations occurring in P, and (b) $(p_1, p_2) \in E$ whenever there is a rule $\rho \in P$ where p_1 is the relation in $head(\rho)$, and p_2 occurs in $body(\rho)$. A *non-recursive datalog query* is a pair $Q = (P, q)$ where P is a set of rules such that $Dep(P)$ has no cycle, and q is a datalog relation; its *arity* is the arity of q.

Given a rule ρ and a DL interpretation $\mathcal{I}$, an *assignment* is a function π that maps every variable of ρ to an object in $\Delta^{\mathcal{I}}$. For a concept atom $A(x) \in \rho$, we write $\mathcal{I} \models_\pi A(x)$ if $\pi(x) \in A^{\mathcal{I}}$, and for a role atom $r(x_1, x_2) \in \rho$, we write $\mathcal{I} \models_\pi r(x_1, x_2)$ if $(\pi(x_1), \pi(x_2)) \in r^{\mathcal{I}}$. We call π a *match* for ρ in $\mathcal{I}$ if $\mathcal{I} \models_\pi \alpha$ for all DL-atoms $\alpha \in \rho$. A tuple $\boldsymbol{t}$ is an *answer* to a query $Q = (P, h)$ in an interpretation $\mathcal{I}$ if there exists a rule $\rho = h(\boldsymbol{x}) \leftarrow \beta$ in P and a match π for ρ in $\mathcal{I}$ such that (i) $\boldsymbol{t} = \pi(\boldsymbol{x})$ and (ii) for each non-DL-atom $p(\boldsymbol{y}) \in \rho$, $\pi(\boldsymbol{y})$ is an answer to (P, p) in $\mathcal{I}$. We use $\mathsf{ans}(Q, \mathcal{I})$ to denote the set of answers to Q in $\mathcal{I}$. The set $\mathsf{cert}(Q, \mathcal{K})$ of *certain answers* to an n-ary query Q over a KB $\mathcal{K}$ is defined as $\{\boldsymbol{a} \in (\mathsf{N_I})^n \mid \boldsymbol{a}^{\mathcal{I}} \in \mathsf{ans}(Q, \mathcal{I})$ for any model $\mathcal{I}$ of $\mathcal{K}\}$.

A *union of conjunctive queries (UCQ)* is a non-recursive datalog query $Q = (P, q)$ such that every rule in P has head relation q and has only DL-atoms in its body. A *conjunctive query (CQ)* is a UCQ of the form $(\{\rho\}, q)$. We typically represent a UCQ as a set of CQs, and often use single rules (or rule bodies) to denote CQs.

The *query output tuple (QOT)* problem takes as input a query Q, a KB $(\mathcal{T}, \mathcal{A})$, and a tuple of individuals $\boldsymbol{a}$, and consists in deciding whether $\boldsymbol{a} \in \mathsf{cert}(Q, (\mathcal{T}, \mathcal{A}))$. Whenever we talk about the *complexity of query answering*, we mean the complexity of the QOT problem. We focus on *combined complexity*, which is measured in terms of the size of the whole input ($\boldsymbol{a}$, Q, $\mathcal{T}$, $\mathcal{A}$).

Canonical Models Every consistent DL-Lite KB $(\mathcal{T}, \mathcal{A})$ possesses a *canonical model* $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Its domain $\Delta^{\mathcal{T},\mathcal{A}}$ consists of all words $aR_1 \ldots R_n$ ($n \geq 0$) such that: (i) $a \in \mathsf{Ind}(\mathcal{A})$ and $R_i \in \overline{\mathsf{N_R}}$, (ii) if $n \geq 1$, then $\mathcal{T}, \mathcal{A} \models \exists R_1(a)$, and (iii) for $1 \leq i < n$, $R_i^- \neq R_{i+1}$ and $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$. We call $w' \in \Delta^{\mathcal{T},\mathcal{A}}$ a *child* of $w \in \Delta^{\mathcal{T},\mathcal{A}}$ if $w' = wR$ for some R. The interpretation function is defined as follows:

$$
\begin{aligned}
a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \ \text{ for all } a \in \mathsf{Ind}(\mathcal{A}) \\
A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \{aR_1 \ldots R_n \mid n \geq 1 \text{ and } \mathcal{T} \models \exists R_n^- \sqsubseteq A\} \\
r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} \cup \{(w_1, w_2) \mid w_2 = w_1 r\} \cup \{(w_2, w_1) \mid w_2 = w_1 r^-\}
\end{aligned}
$$

Note that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is composed of a *core*, which is obtained by restricting $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ to the objects in $\mathsf{Ind}(\mathcal{A})$, and an *anonymous part* consisting of (possibly infinite) trees rooted at objects in the core. It is well-known that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can be homomorphically mapped into any model of $\mathcal{T}$ and $\mathcal{A}$, which yields:

Fact 1 *Let $\mathcal{K}$ be a consistent DL-Lite KB, and let $\mathcal{I}_{\mathcal{K}}$ be its canonical model. Then $\mathsf{cert}(Q, \mathcal{K}) = \mathsf{ans}(Q, \mathcal{I}_{\mathcal{K}})$ for every non-recursive datalog query Q.*

3 General Tractability Result

In this section, we observe that the answers to a CQ ρ over a consistent DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ coincide with the answers to ρ in an interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ that can be constructed in polynomial time from $\mathcal{K}$ and ρ. As a consequence, we obtain that any class of CQs that is tractable for plain databases is also tractable for DL-Lite KBs.

To establish this result, we rely on the notion of *tree witness* introduced in [11]:

Definition 1. *Let ρ be a CQ and let $R(x, y)$ be such that either $R(x, y) \in \rho$ or $R^-(y, x) \in \rho$. A* tree witness *for $R(x, y)$ in ρ is a partial map f from the variables in ρ to words over the alphabet $\overline{\mathsf{N_R}}$ satisfying:*
(a) $f(y) = R$,
(b) if $f(z) = wS$, $S'(z, z') \in \rho$ or $S'^-(z', z) \in \rho$, and $S' \neq S^-$, then $f(z') = wSS'$,
(c) if $f(z) = wS$ and $S(z', z) \in \rho$ or $S^-(z, z') \in \rho$, then $f(z') = w$.
and such that its domain is minimal (w.r.t. $\subseteq$) among partial maps satisfying (a)-(c).

It follows from the definition that there can be at most one tree witness for each $R(x, y)$, which we denote by $f_{R(x,y)}$. Notice that each tree witness f for a CQ ρ naturally induces a tree-shaped CQ, denoted ρ_f, obtained by restricting ρ to the variables in the domain of f and unifying variables z, z' with $f(z) = f(z')$.

Definition 2. *A tree witness $f_{R(x,y)}$ is* valid w.r.t. $\mathcal{T}$ *if for every word $R_1 \dots R_n$ in the range of $f_{R(x,y)}$, and every $1 \leq i < n$, we have $R_{i-1} \neq R_i$ and $\mathcal{T} \models \exists R_{i-1}^- \sqsubseteq \exists R_i$.*

The notion of validity essentially mimics condition (iii) in the definition of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, thereby ensuring that whenever a tree witness $f_{R(x,y)}$ is valid and there is an element of the form wR in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, then the element w is the root of a tree that contains ww' for every w' in the range of $f_{R(x,y)}$. Moreover, the image of any match for ρ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ restricted to the elements in the anonymous part always corresponds to a union of such trees. Hence, if a match π for ρ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ maps x and y to objects w and wR respectively, then there is a tree witness $f_{R(x,y)}$ for $R(x, y)$ in ρ which is valid w.r.t. $\mathcal{T}$ and such that $\pi(z) = wf_{R(x,y)}(z)$ for each variable z in the domain of $f_{R(x,y)}$. This means that by taking the range of all the valid tree witnesses, we can obtain all the tree-shaped structures that can potentially participate in matches for subqueries of ρ. By appropriately augmenting the core with these structures, we obtain an interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ that is sufficient for retrieving all query answers. Formally, we have:

Definition 3. *Let $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ be the structure obtained by adding to the core:*
(a) the objects aw such that w occurs in the range of a valid tree witness $f_{R(x,y)}$, $\mathcal{T}, \mathcal{A} \models \exists R(a)$, and
(b) the objects xSw where x is a variable in ρ and S, w satisfy:
  (i) there is an individual $a \in \mathsf{Ind}(\mathcal{A})$ and a chain of (possibly inverse) roles $R_1, \dots, R_n$ of length at most $|\mathcal{T}|$ such that $R_n = S$, $\mathcal{T}, \mathcal{A} \models \exists R_1(a)$, and for each $1 < i \leq n$, $R_{i-1}^- \neq R_i$, and $\mathcal{T} \models \exists R_{i-1}^- \sqsubseteq \exists R_i$,
  (ii) there exists a variable y and valid tree witness $f_{R(x,y)}$ with $S^- \neq R$ and $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$ whose range contains w.

  Note that x is not *an object of the domain of $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$.*
To extend the interpretations of concept and role names to these new objects, we let $wR \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}}$ whenever $\mathcal{T} \models \exists R^- \sqsubseteq A$, and $(w, wR) \in R^{\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}}$ for new w, wR.

Item (a) of Definition 3 attaches to each individual a a copy of the tree structure associated with tree witness f whenever the query q_f has a match rooted at a. Item (b) handles the situation in which there is match π for a connected component ρ' of the query ρ which maps all variables to objects in the anonymous part, in which case the set $\{\pi(y) \mid y \text{ is variable in } \rho'\}$ form a tree. By letting x be the variable in ρ' mapped closest to the ABox, and S be its incoming arc, the new objects xSw introduced in (b) induce the same tree structure, allowing the match for ρ' to be reproduced in $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$.

Theorem 1. *Let ρ be a CQ, let $(\mathcal{T},\mathcal{A})$ be a consistent DL-Lite knowledge base, and let $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ be the interpretation defined above. The following statements hold:*

1. *For every tuple $\boldsymbol{a}$ of individuals, $\boldsymbol{a} \in \mathsf{ans}(\rho, \mathcal{I}_{\mathcal{T},\mathcal{A}})$ iff $\boldsymbol{a} \in \mathsf{ans}(\rho, \mathcal{I}_{\mathcal{T},\mathcal{A},\rho})$.*
2. *$\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ can be built in polynomial time in ρ, $\mathcal{T}$ and $\mathcal{A}$.*

Proof (sketch). The "if" direction of statement 1 is immediate since $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ corresponds to a subinterpretation of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. For the "only if" direction, we must show how every match π for ρ in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can be reproduced in $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$. This is straightforward for connected components of ρ which contain an answer variable since we can directly use π. The remaining connected components are mapped into one of the tree-shaped structures from item (b) of Definition 3. For statement 2, existence and validity of tree witnesses can be tested in polynomial time. The existence of a role chain and individual having the properties stated in (b) can be decided by initializing a set Reach with all roles S such that $\mathcal{T},\mathcal{A} \models \exists S(a)$ for some $a \in \mathsf{Ind}(\mathcal{A})$, and adding U to Reach whenever there is $V \in \mathsf{Reach}$ such that $\mathcal{T} \models \exists V^- \sqsubseteq \exists U$. Since there are only polynomially many objects of the forms aw and bw as above, and instance checking and TBox reasoning are tractable for DL-Lite KBs [3], $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ can be built in polynomial time.

In light of Theorem 1 and Fact 1, to determine whether $\boldsymbol{a} \in \mathsf{cert}(\rho,(\mathcal{T},\mathcal{A}))$, it is sufficient to test the consistency of $(\mathcal{T},\mathcal{A})$ and then, if $(\mathcal{T},\mathcal{A})$ is consistent, to decide whether $\boldsymbol{a} \in \mathsf{ans}(\rho, \mathcal{I}_{\mathcal{T},\mathcal{A},\rho})$. If we view $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ as a plain relational database, the latter check is just a special case of the QOT problem. Hence, we obtain:

Corollary 1. *Let $\mathcal{Q}$ be a class of CQs for which the query output tuple problem over plain relational structures is decidable in polynomial time. Then the query output tuple problem for $\mathcal{Q}$ is also tractable for KBs formulated in DL-Lite.*

The construction of $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ above is easily extended to DL-Lite$_\mathcal{R}$ using the definitions of canonical models and tree witnesses from [10]. However, the construction is no longer polynomial since there can be exponentially many tree witnesses for a single atom $R(x,y)$ in ρ [9], and so we do not obtain an analogue of Theorem 1. Indeed, it follows from results by Kikot et al. [9] that the tuple output problem for tree-shaped CQs is NP-complete for DL-Lite$_\mathcal{R}$ KBs. Therefore, to obtain tractability results to DL-Lite$_\mathcal{R}$, one must impose some syntactic restriction on $\mathcal{T}$ and ρ that ensures a polynomial bound on the number of tree witnesses, e.g. the absence of *twisty roles* proposed in [10].

4 Answering Acyclic Queries

The expansion technique presented in Section 3 allows us to convert any polynomial-time algorithm for evaluating a class of CQs over plain databases into a polynomial-time

algorithm for evaluating the same class of queries over DL-Lite KBs. However, the resulting algorithm would involve building the structure $\mathcal{I}_{\mathcal{T},\mathcal{A},\rho}$ for each input query ρ, which is clearly undesirable. We now present our main contribution: a polynomial-time procedure for evaluating acyclic CQs which is based upon rewriting CQs into non-recursive datalog programs.

We begin with some preliminaries. To prepare for the extension to nearly acyclic queries in the next section, we will allow CQs to use a special concept name core, whose purpose is to force some variables to be mapped to the core. Semantically, this is achieved by defining $\mathsf{cert}(\rho, \mathcal{K})$ as the set of tuples $\boldsymbol{a}$ such that in every model $\mathcal{I}$ of $\mathcal{K}$, there is a match π for the CQ obtained by removing all core atoms from ρ, such that π maps the answer variables to $\boldsymbol{a}$ and maps every variable x with $\mathsf{core}(x) \in \rho$ to an element $a^{\mathcal{I}}$ (with a an individual in $\mathcal{K}$). Note that for CQs with no core atoms, this definition coincides with the earlier one. We say that a variable x is a *core variable* in ρ if it is an answer variable, or ρ contains the atom $\mathsf{core}(x)$ or an atom of the form $r(x, x)$.

As usual, the query graph $G(\rho)$ of a CQ ρ is defined as the undirected graph whose nodes are the variables of ρ, and that has an edge between x and x' if ρ contains a (body) atom $r(x, x')$ or $r(x', x)$. A CQ ρ is *acyclic* if $G(\rho)$ is acyclic, and *rooted* if every connected component of $G(\rho)$ has at least one core variable. We consider a slight generalization of acyclicity: we say that a CQ ρ is *acyclic modulo core variables* (*c-acyclic* for short) if the graph $G^{-}(\rho)$ is acyclic, where $G^{-}(\rho)$ is obtained by deleting from $G(\rho)$ each edge (x, x') where x and x' are core variables.

Our rewriting procedure works on queries which are both rooted and c-acyclic (we will discuss later how to handle the non-rooted case). To every rooted c-acyclic CQ ρ, we associate the set of connected components $\{T_1, \ldots, T_n\}$ of $G^{-}(\rho)$. Because ρ is rooted, every T_i is a connected acyclic graph containing at least one vertex which is a core variable. We select an arbitrary core variable x_i for each T_i and designate it as the root of T_i, allowing us to view T_i as a tree. Then, given a pair of variables x, y of ρ, we say y is a *child* of x if y is a child of x in the (unique) tree T_i that contains x, and define *descendant* as the transitive closure of child.

Rewriting procedure for rooted queries Consider a rooted c-acyclic CQ $\rho = q(\boldsymbol{v}) \leftarrow \alpha$ and a DL-Lite TBox $\mathcal{T}$. Let X and X_r denote respectively the set of core variables and the set of root variables (recall that $X_r \subseteq X$). For a variable x of ρ, we denote by $\boldsymbol{x}_\rho$ the tuple $\langle y_1, \ldots, y_k \rangle$ consisting of the answer variables of ρ that are descendants of x. We rewrite the query ρ into the non-recursive datalog program $\mathsf{rew}_{\mathcal{T}}(\rho) = (P, q)$ defined as follows. In addition to concepts and roles, and the relation q, the program uses the following datalog relations: (i) $(|\boldsymbol{x}_\rho| + 1)$-ary relations q_x, q'_x for every variable x of ρ, and (ii) unary relations q_A and $q_{\exists R}$ for every $A \in \mathsf{N_C}$ and $R \in \overline{\mathsf{N_R}}$ occurring in ρ. We now describe the rules in P. There is a single top-level rule defining q:

$$q(\boldsymbol{v}) \leftarrow \bigwedge_{x \in X_r} q_x(x, \boldsymbol{x}_\rho) \wedge \bigwedge_{x_i, x_j \in X \wedge r(x_i, x_j) \in \rho} r(x_i, x_j) \tag{1}$$

For every variable x in ρ, with $Y = \{y_1, \ldots, y_n\}$ the set of children of x in ρ, we have

$$q_x(x, \boldsymbol{x}_\rho) \leftarrow \bigwedge_{A(x) \in \rho, A \neq \mathsf{core}} q_A(x) \wedge \bigwedge_{y \in Y} q'_y(x, \boldsymbol{y}_\rho) \tag{2}$$

and for every $y \in Y$, we also have

$$q'_y(x, \boldsymbol{y}_\rho) \leftarrow \bigwedge_{r(x,y)\in\rho} r(x,y) \wedge \bigwedge_{s(y,x)\in\rho} s(y,x) \wedge q_y(y, \boldsymbol{y}_\rho) \tag{3}$$

and for all $y \in Y$ satisfying the following conditions:

(i) there is an atom $R(x,y) \in \rho$ or $R^-(y,x) \in \rho$ and the tree witness $f_{R(x,y)}$ exists and is valid
(ii) for every u in the domain of $f_{R(x,y)}$ with $f_{R(x,y)}(u) = wS$ and $A(u) \in \rho$, we have $\mathcal{T} \models \exists S^- \sqsubseteq A$
(iii) the set $Z = \{z \mid f_{R(x,y)}(z) = \varepsilon \wedge z \neq x\}$ contains all core variables in the domain of $f_{R(x,y)}$

we have the further rule

$$q'_y(x, \boldsymbol{y}) \leftarrow q_{\exists R}(x) \wedge \bigwedge_{z\in Z} q_z(x, \boldsymbol{z}_\rho) \tag{4}$$

Finally, for every $B \in \mathsf{N_C} \cup \{\exists R \mid R \in \overline{\mathsf{N_R}}\}$ with q_B a datalog relation in P, we have

$$\begin{array}{ll} q_B(x) \leftarrow A(x) & \text{for all } A \in \mathsf{N_C} \text{ such that } \mathcal{T} \models A \sqsubseteq B \\ q_B(x) \leftarrow s(x,y) & \text{for all } s \in \mathsf{N_R} \text{ such that } \mathcal{T} \models \exists s \sqsubseteq B \\ q_B(x) \leftarrow s(y,x) & \text{for all } s \in \mathsf{N_R} \text{ such that } \mathcal{T} \models \exists s^- \sqsubseteq B \end{array} \tag{5}$$

Notice that the special concept core does not appear in the program $\mathsf{rew}_\mathcal{T}(\rho)$.

Intuitively, the relation q_x corresponds to the query $\rho|x$ whose answer variables are $\{x\} \cup \boldsymbol{x}_\rho$ and whose body is obtained by restricting the body of ρ to the atoms whose arguments among x and its descendants; the relation q'_y corresponds to the query $\rho|xy$ (with y a child of x) obtained by adding to $\rho|y$ the role atoms linking x and y. Rule (1) stipulates that a tuple is in the answer to ρ if it makes true all of the role atoms linking two core variables and each of the queries $\rho|x$ associated with a root variable x of ρ. Then rule (2) states that to make $\rho|x$ hold at an individual, we must satisfy the concept atoms for x and the query $\rho|xy$ for each child y of x. Rules (3) and (4) provide two ways of satisfying $\rho|xy$. The first way, captured by rule (3), is to map y to an ABox individual, in which case the role atoms between x and y must occur in the ABox, and the query $\rho|y$ must hold at this individual. The second possibility, treated by rule (4), is that y is mapped to an element of the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ which is a child of x. For this to occur, several conditions must be verified. First, if y is an R-successor of x, then the tree witness $f_{R(x,y)}$ must exist and be valid w.r.t. $\mathcal{T}$. Second, we must ensure that all concept atoms concerning variables that are mapped inside the anonymous part by $f_{R(x,y)}$ are satisfied (this is checked in item (ii)). Finally, since core variables cannot be mapped inside the anonymous part, we need condition (iii), which checks that every core variable z in the domain of $f_{R(x,y)}$ is such that $f_{R(x,y)}(z) = \varepsilon$. If all of these conditions are met, then rule (4) states that the query $\rho|xy$ can be satisfied by making $\exists R$ hold at x (thereby guaranteeing the existence of the required paths in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$) *and* by satisfying the remainder of the query $\rho|xy$ (i.e. the query obtained

by removing the atoms mapped inside the anonymous part). The latter corresponds to the union of the queries $\rho|z$ where z is a descendant of x with $f_{R(x,y)}(z) = \varepsilon$. Finally, the rules in (5) provide the standard rewriting of concepts A and $\exists R$ w.r.t. $\mathcal{T}$.

We establish the correctness of our rewriting procedure.

Theorem 2. *Let ρ be a rooted c-acyclic CQ and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a DL-Lite KB. Then* $\mathsf{cert}(\rho, \mathcal{K}) = \mathsf{ans}(\mathsf{rew}_{\mathcal{T}}(\rho), \mathcal{I}_{\mathcal{A}})$.

Proof (idea). The key step in the proof is showing that for every variable x in ρ, $\mathsf{cert}(\rho|x, \mathcal{K}) = \mathsf{ans}((\mathsf{rew}_{\mathcal{T}}(\rho), q_x), \mathcal{I}_{\mathcal{A}})$. This can be proved by induction on the number of variables in $\rho|x$, utilizing Proposition 1 and properties of tree witnesses.

The next theorem shows that our rewriting procedure provides a polynomial-time algorithm for evaluating rooted c-acyclic conjunctive queries.

Theorem 3. *Given a rooted c-acyclic CQ ρ and a DL-Lite KB $(\mathcal{T}, \mathcal{A})$, the program* $\mathsf{rew}_{\mathcal{T}}(\rho)$ *can be computed in polynomial time in the size of ρ and $\mathcal{T}$, and deciding if* $\boldsymbol{a} \in \mathsf{ans}(\mathsf{rew}_{\mathcal{T}}(\rho), \mathcal{I}_{\mathcal{A}})$ *can be done in polynomial time in the size of* $\mathsf{rew}_{\mathcal{T}}(\rho)$ *and $\mathcal{A}$.*

Proof. For the first point, we observe that the number of relations in $\mathsf{rew}_{\mathcal{T}}(\rho)$ is linear in the number of atoms in ρ and that each relation is defined using linearly many rules in the size of ρ and $\mathcal{T}$. We also note that testing the conditions for rules of type (4) can be done in polynomial time in the size of ρ and $\mathcal{T}$ (cf. Section 3). The second statement is true because once the answer variables in $\mathsf{rew}_{\mathcal{T}}(\rho)$ have been instantiated with the individuals in $\boldsymbol{a}$, we have a non-recursive datalog program (with constants) where every rule contains at most two variables. It is known that datalog programs of this form can be evaluated in polynomial time (cf. [5]).

Handling non-rooted queries. We now show that non-rooted c-acyclic CQs can be answered via a reduction to the rooted query case. Given an c-acyclic CQ ρ over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the set $\mathsf{cert}(\rho, \mathcal{K})$ can be computed using the following steps:

1. If ρ is rooted, then return $\mathsf{cert}(\rho, \mathcal{K})$.
2. Choose a maximally connected component β of ρ that contains no core variable.
3. If β has a match fully in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, drop it and go to step 1.
4. If there is a variable x in β such that the rooted query $g(x) \leftarrow \beta$ has a non-empty answer over $\mathcal{K}$, then drop β from ρ and go to step 1. Otherwise, return $\emptyset$.

It is not hard to show that the condition in step 3 holds just in the case that there exists a variable x and a role $S \in \overline{\mathsf{N_R}}$ which is reachable in the canonical model (cf. Definition 3 b.i) such that there is a valid tree witness f for $S(z, x)$ in $\beta_{S,x} = \beta \cup \{S(z, x)\}$ (z fresh) such that: (i) if $f(y) = \epsilon$, then $y = z$, and (ii) $\mathcal{T} \models \exists R^- \sqsubseteq A$ whenever $f(y) = wR$ and $A(y) \in \beta$. It follows that step 3 can be carried out in polynomial time, and thus we obtain a polynomial-time procedure for arbitrary c-acyclic CQs.

Answering UCQs Given a union of c-acyclic CQs $Q = (P, q)$, we can decide whether $\boldsymbol{a} \in \mathsf{cert}(Q, \mathcal{K})$ by testing whether $\boldsymbol{a} \in \mathsf{cert}(\rho, \mathcal{K})$ for some c-acyclic CQ $\rho \in P$. Since there are only linearly many $\rho \in P$, and each check $\boldsymbol{a} \in \mathsf{cert}(\rho, \mathcal{K})$ takes only polynomial time, we obtain a polynomial procedure for unions of c-acyclic CQs.

Answering acyclic queries in DL-Lite$_\mathcal{R}$ To support role inclusions, we can replace atoms $r(x, y)$ by atoms $q_r(x, y)$, and add the corresponding rules $q_r(x, y) \leftarrow S(x, y)$ with $\mathcal{T} \models S \sqsubseteq r$. Using the notion of tree witness in [10], the modified rewriting can be directly employed for DL-Lite$_\mathcal{R}$, although polynomiality can be guaranteed only in the special cases in which the number of tree witnesses is polynomially bounded.

5 Answering Nearly Acyclic Queries

We show how the rewriting algorithm from the previous section can be generalized to queries which are "nearly" acyclic. Formally, we call a conjunctive query ρ *k-acyclic* if deleting k atoms from ρ leads to a c-acyclic query[3]. Our goal is to show that, when k is bounded by a constant, any k-acyclic query ρ can be rewritten in polynomial time into a union of c-acyclic CQs which is equivalent for the purposes of query answering.

We start by showing the following:

Proposition 1. *Assume ρ is a k-acyclic CQ. Then we can obtain a UCQ Q such that:*
1. *each $\rho' \in Q$ is $(k-1)$-acyclic,*
2. *$\mathsf{cert}(Q, \mathcal{K}) = \mathsf{cert}(\rho, \mathcal{K})$ for any DL-Lite KB $\mathcal{K}$, and*
3. *$|Q| \leq 3$.*

If k is bounded by a constant c, we can obtain Q in polynomial time in the size of ρ.

Proof. Let Γ be a $\subseteq$-minimal set of atoms from ρ such that $\rho \setminus \Gamma$ is c-acyclic. Observe that the minimality of Γ implies that Γ contains only role atoms. Pick an atom $\alpha = R(x, y)$ in Γ. The CQs in Q capture the three possible ways for a match π to map $R(x, y)$ into the canonical model of $\mathcal{K}$: (a) both $\pi(x), \pi(y)$ are in the core, (b) $\pi(y)$ is a child of $\pi(x)$, and (c) $\pi(x)$ is a child of $\pi(y)$. To capture (a), we consider the query ρ_c obtained by adding to ρ the atoms $\mathsf{core}(x)$ and $\mathsf{core}(y)$. For case (b), we compute from ρ the query $\rho_\downarrow$ by *marking* the variables that must map into the anonymous part. We start by marking the variable y in ρ, and then exhaustively apply the following rule:

> *Suppose ρ has an atom $R_1(x_1, x_2)$ or $R_1^-(x_2, x_1)$ such that x_2 is marked. Suppose ρ also has an atom $R_2(x_2, x_3)$ or $R_2^-(x_3, x_2)$ such that $x_3 \neq x$ and x_3 is not marked. If $R_2 = R_1^-$, then replace in ρ the variable x_3 by x_1. If $R_2 \neq R_1^-$, then mark x_3.*

The query $\rho_\uparrow$, which deals with case (c), is obtained from ρ by marking x, and then applying exhaustively the above rule modified by replacing "$x_3 \neq x$" with "$x_3 \neq y$". We call ρ_d ($d \in \{\downarrow, \uparrow\}$) *proper* if the set of marked variables in ρ_d induces a tree-shaped query, and no core variable is marked. We let $Q = \{\rho_c\} \cup \{\rho_d \mid d \in \{\downarrow, \uparrow\}, \rho_d \text{ is proper}\}$. It is not too difficult to show $\mathsf{cert}(Q, \mathcal{K}) = \mathsf{cert}(\rho, \mathcal{K})$ for any KB $\mathcal{K}$.

Let ρ_α denote the query obtained by deleting α from ρ; clearly ρ_α is $(k-1)$-acyclic. To see that ρ_c is $(k-1)$-acyclic, observe that $\rho_c = \rho_\alpha \cup \{\mathsf{core}(x), R(x, y), \mathsf{core}(y)\}$, so $G^-(\rho_c \setminus \Gamma') = G^-(\rho_\alpha \setminus \Gamma')$ for any Γ' (recall that G^- is obtained from the query graph by deleting edges between core variables). Thus, since ρ_α is $(k-1)$-acyclic, ρ_c must also be $(k-1)$-acyclic. Next take a proper query ρ_d, with $d \in \{\downarrow, \uparrow\}$, and

[3] We remark that k-acyclicity is inspired by the notion of *bounded feedback edge sets* often considered in parameterized complexity, cf. [6].

let M be a simple cycle in $G^-(\rho_d)$ (if no cycle exists, ρ_d is trivially $(k-1)$-acyclic). We note that M cannot use any edge involving a marked variable of ρ_d. Moreover, if $p(u,v) \in \rho_d$ and none of u, v is marked, then $p(u,v) \in \rho_\alpha$. This implies that M is also a simple cycle in $G^-(\rho_\alpha)$. In other words, $G^-(\rho_\alpha)$ contains every simple cycle in $G^-(\rho_d)$. Thus, since ρ_α is $(k-1)$-acyclic, the query ρ_d is also $(k-1)$-acyclic.

Provided α is known, Q can be obtained in polynomial time in the size of ρ. If k is bounded by a constant, then α can also be identified in polynomial time because the set Γ can be computed by considering all (polynomially many) subsets of at most k atoms.

Repeated applications of Proposition 1 yield the desired polynomial reduction to c-acyclic UCQs:

Corollary 2. *For any k-acyclic query ρ there is a UCQ Q such that:*
1. *each $\rho' \in Q$ is c-acyclic,*
2. $\mathsf{cert}(Q, \mathcal{K}) = \mathsf{cert}(\rho, \mathcal{K})$ *for any DL-Lite KB $\mathcal{K}$, and*
3. $|Q| \leq 3^k$.

If k is bounded by a constant, Q can be computed in polynomial time in the size of ρ.

Since c-acyclic UCQs can be answered in polynomial time using the rewriting technique from Section 4, we obtain a polynomial-time procedure for k-acyclic CQs.

6 Preliminary Evaluation

We developed a prototype rewriting system that takes as input a rooted acyclic CQ ρ and a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$, and outputs an SQL statement expressing the resulting non-recursive datalog program $\mathsf{rew}_\mathcal{T}(\rho)$ (using common table expressions). We evaluated the result over ABoxes stored in a relational database, using the PostgreSQL database system, and Owlgres [19] for loading the data.

To test our prototype, we used the LUBM$^\exists_{20}$ ontology described in [13], which adds concept inclusions with additional concept names, and with existential concepts on the right hand side, to the original LUBM ontology [8]. We considered three acyclic queries from the benchmark in [13] (q_2, q_4, and q_5 from the 6 provided queries), which are rather small (at most 4 atoms), and created three additional large acyclic queries, with 13 to 34 atoms, and 7 to 17 variables (q_7, q_8, and q_9). We note that the new queries are also significantly larger than the ones of the REQUIEM test suite (≤ 7 atoms). The importance of handling such larger queries in practice has been previously argued in [18].

We compared our rewriting procedure with REQUIEM [15] and IQAROS [20] which, like most of the existing query rewriting systems for the DL-Lite family, generate unions of CQs rather that non-recursive datalog programs. For the three large queries, both REQUIEM and IQAROS did not terminate (within ten minutes). Even for the small q_4 and q_5, the generated rewritings were too large to be posed directly to an off-the-shelf RDBMs: REQUIEM generated tens of thousands of queries for both, and IQAROS almost 15 thousand for q_4, and almost one thousand for q_5. In contrast, for our approach, the rewriting times were negligible for all queries (under half a second). The rewritings produced by our approach have less than 30 rules for all queries, disregarding the rules (5) of the algorithm (since the latter are independent of the query, we computed them separately and stored them using a database view per concept/role name).

#Uni	q_2	q_4	q_5	q_7	q_8	q_9
20	2.5	3.0	4.2	1.5	1.0	0.0
50	9.0	7.3	4.6	2.0	3.3	0.0
100	20.5	15.0	9.4	4.2	7.2	0.0
150	25.6	21.8	14.1	6.6	11.6	15.2
200	33.5	>600	27.0	15.2	26.9	31.2

Table 1. Scalability of our system (runtime in seconds)

We also tested the feasibility of evaluating our rewritings over large ABoxes. For this, we used the modified LUBM data generator [13] (with 5% incompleteness). Each university has approximately 17k individuals, 28K concept assertions, and 47K role assertions. We carried out our experiments on ABoxes with 20 – 200 universities, resulting in very large ABoxes (up to ca. 1.5 GB on disk). The results reported in Table 1 show that the algorithm scales well.

Finally, we note that a performance comparison with PRESTO [18], which also outputs non-recursive datalog programs, was not possible because this system is not publicly available. However, we can observe that its underlying algorithm may produce exponential-size rewritings for acyclic CQs, as witnessed by the family of queries $q(x) \leftarrow r(x,y) \wedge \bigwedge_{0\leq i\leq n} p(y,z_i) \wedge p(u_i,z_i) \wedge B_i(u_i)$ coupled with e.g. the empty TBox.[4] Intuitively, the PRESTO algorithm generates a separate rule for every possible way to select a collection of variables from $\{u_1,\ldots,u_n\}$ and identify them with y. This exponential blow-up suggests that our positive results are not merely an artifact of the datalog representation, but derive also from acyclicity.

7 Future Work

We plan to generalize our rewriting technique to larger tractable classes of CQs, like bounded treewidth CQs, and also to try to identify suitable restrictions for more expressive DLs that allow for tractable query answering. Our proof-of-concept implementation raises hopes that efficient evaluation of large (nearly) acyclic queries is feasible, but many challenges remain. For example, we observe that breaking down the queries into small rules as is done by our rewriting may lead to a loss of structure that could be used by database management systems for optimized evaluation. There are many other aspects, not directly related to the rewriting technique, that must also be taken into account to achieve practicable query answering, such as exploring more efficient forms of representing data in ABoxes, using different kinds of indexes, and considering different translations of our programs into SQL. Using semantic indexes [17] for handling the rules of type (5) appears particularly promising.

Acknowledgements This work was supported by a Université Paris-Sud Attractivité grant and ANR project PAGODA ANR-12-JS02-007-01 (Bienvenu), FWF project T515-N23 (Ortiz), FWF project P25518-N23 and WWTF project ICT12-015 (Šimkus), Vienna PhD School of informatics and EU project Optique FP7-318338 (Xiao).

[4] In fact, the exponential blowup occurs even without the atoms $B_i(u_i)$, but some quite obvious modifications to the algorithm would resolve the issue. With these atoms present, it appears non-trivial changes to the algorithm would be required.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. of IJCAI. pp. 364–369 (2005)
2. Bienvenu, M., Ortiz, M., Simkus, M., Xiao, G.: Tractable queries for lightweight description logics. In: Proc. of IJCAI (2013)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
4. Chekuri, C., Rajaraman, A.: Conjunctive query containment revisited. In: Proc. of ICDT. pp. 56–70. Springer (1997)
5. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3), 374–425 (Sep 2001)
6. Festa, P., Pardalos, P.M., Resende, M.G.C.: Feedback set problems. In: Handbook of Combinatorial Optimization. pp. 209–258. Kluwer Academic Publishers (1999)
7. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. In: Proc. of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. pp. 21–32. ACM Press (1999)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
9. Kikot, S., Kontchakov, R., Zakharyaschev, M.: On (in)tractability of OBDA with OWL 2 QL. In: Description Logics (2011)
10. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of KR. AAAI Press (2012)
11. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR. AAAI Press (2010)
12. Levy, A.Y., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artif. Intell. 104(1-2), 165–209 (1998)
13. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters (with appendix). In: Proc. of SSWS+HPCSW (2012)
14. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009), available at `http://www.w3.org/TR/owl2-overview/`
15. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: Description Logics. CEUR-WS.org (2009)
16. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like? In: Proc. of SWIM. ACM (2011)
17. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: Proc. of KR. pp. 308–318. AAAI Press (2012)
18. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of KR. AAAI Press (2010)
19. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: Proc. of OWLED. CEUR-WS.org (2008)
20. Venetis, T., Stoilos, G., Stamou, G.B.: Incremental query rewriting for OWL 2 QL. In: Description Logics. CEUR-WS.org (2012)
21. Yannakakis, M.: Algorithms for acyclic database schemes. In: Proc. of VLDB. pp. 82–94. IEEE Computer Society (1981)