# The Combined Approach to OBDA: Taming Role Hierarchies using Filters

Carsten Lutz[1], İnanç Seylan[1], David Toman[2], and Frank Wolter[3]

[1]Universität Bremen, Germany
{clu,seylan}@informatik.uni-bremen.de
[2]Cheriton School of CS, University of Waterloo, Canada
david@cs.uwaterloo.ca
[3]University of Liverpool, United Kingdom
wolter@liverpool.ac.uk

**Abstract.** There are several approaches to implementing query answering over instance data in the presence of an ontology that target conventional relational database systems (SQL databases) as their back-end. They all share the limitation that, for ontologies formulated in OWL2 QL or versions of the description logic DL-Lite that admit both role hierarchies and inverse roles, it seems impossible to avoid an exponential blowup of the query (and sometimes this is even provable). We consider the *combined approach* and propose to replace its query rewriting part with a *filtering technique*. This is natural from an implementation perspective and allows us to handle both inverse roles and role hierarchies without an exponential blowup. We also carry out an experimental evaluation that demonstrates the scalability of this approach.

## 1 Introduction

In recent years, ontology-based data access (OBDA) has emerged as a promising and challenging application of ontologies. The idea is to enrich instance data with a 'semantic layer' in the form of an ontology, used as an interface for querying and to derive additional answers. A central research problem in this area is to design query answering engines that can deal with sufficiently expressive ontology languages yet scale to large data sets. The most popular ontology languages that have been considered for OBDA include the three profiles OWL2 RL, OWL2 QL, and OWL2 EL, as well as various description logics and datalog variants related to these profiles [2, 3, 5, 11, 15].

Currently, there are two major methodologies for answering queries in an OBDA setting: rewriting-based approaches (also called backward chaining) and materialization-based approaches (also called forward chaining). In the former, one compiles the ontology $\mathcal{T}$ and the query $q$ into a new query $q_{\mathcal{T}}$ that contains the relevant knowledge from the ontology, i.e., the answers to $q$ over $\mathcal{A}$ and $\mathcal{T}$ coincide with the answers to $q_{\mathcal{T}}$ over $\mathcal{A}$. One can thus store $\mathcal{A}$ in a relational database management system RDBMS and execute $q_{\mathcal{T}}$ over $\mathcal{A}$. In materialization

approaches, the instance data $\mathcal{A}$ is completed with the relevant knowledge from the ontology $\mathcal{T}$, i.e. for any query $q$, the answers given to $q$ over $\mathcal{A}$ and $\mathcal{T}$ coincide with the answers given to $q$ over the completed instance data $\mathcal{A}_\mathcal{T} \supseteq \mathcal{A}$ without any ontology. Thus, one can store $\mathcal{A}_\mathcal{T}$ in a relational database management system (RDBMS) and execute $q$ over $\mathcal{A}_\mathcal{T}$.

A technical problem that arises in materialization approaches is that the completed data $\mathcal{A}_\mathcal{T}$ easily becomes infinite; in particular, this may happen when the ontology expresses cyclic dependencies and has existential quantifiers in the heads of its concept inclusions, which is allowed in most ontology languages including the ones mentioned above. To overcome this problem, an economic way of reusing individuals introduced for existential quantifiers has been proposed in [8, 9] for the case where ontologies are formulated in description logics from the $\mathcal{EL}$ and DL-Lite families, which are the logical cores of the OWL2 EL and OWL2 QL ontology languages. While the resulting completed data sets are finite and give correct answers to instance queries, they can give spurious answers to conjunctive queries (CQs). To recover soundness for CQs, it is thus necessary to include an additional step, resulting in the *combined approach* to query answering: the original query is rewritten in a way that eliminates spurious answers. In sharp contrast to pure rewriting, the *auxiliary query rewriting* required in the combined approach turns out to be rather simple—an additional *selection condition* applied to the results of the original CQ over the completed data—and often of polynomial size. Indeed, experiments indicate that the combined approach admits very efficient query execution for expressive variants of $\mathcal{EL}$ and DL-Lite [8, 9].

Unfortunately, there are certain combinations of logical operators that are important from an application perspective, but for which an exponential blowup of the query seems to be unavoidable both in the query rewriting approach and in the combined approach. In particular, this is the case for the combination of inverse roles and role hierarchies as found in DL-Lite$_\mathcal{R}$ [3], the extension of basic DL-Lite with role hierarchies that underpins OWL2 QL. It has been shown that, in the query rewriting approach, an exponential blowup of the query size is unavoidable when the ontology is formulated in DL-Lite$_\mathcal{R}$ [7]. For the combined approach, an auxiliary query rewriting strategy for DL-Lite$_\mathcal{R}$ ontologies and CQs is presented in [8], but it incurs an exponential blowup and it seems unlikely that the rewriting can be improved to a poly-sized one (although this question is yet to be resolved).

In this paper, we present a new variation on the combined approach that can handle CQs and DL-Lite$_\mathcal{R}$ ontologies and eliminates the need for auxiliary query rewriting altogether, thus also eliminating the need to deal with exponentially sized queries. Specifically, we replace auxiliary query rewriting with a *filtering component*: spurious answers are eliminated by a polynomial-time filtering procedure (called a *filter* in the rest of the paper) that is installed as a user-defined function in the underlying RDBMS. Our main contributions are as follows.

(1) We develop a polynomial time procedure for filtering out spurious answers to CQs for ontologies formulated in DL-Lite$_\mathcal{R}$. Interestingly, the existence of such

a filtering procedure appears to be quite sensitive to how exactly the instance data is completed. Compared to the data completion for the original combined approach [8], the filtering technique requires subtle modifications to the data completion in order to obtain a polytime filter.

(2) To analyze the performance of our approach and to compare it with the query rewriting approach, we modify the Lehigh University Benchmark (LUBM) [6] by introducing additional existential restrictions and subconcepts into the LUBM ontology and incompleteness into the LUBM Data Generator. Additional queries that are designed to stress-test the performance of the filtering procedure are introduced as well.

(3) Finally, the resulting ontologies, data, and queries are used to demonstrate the feasibility of our approach, and to show that it scales to large amounts of instance data.

Some technical proofs and details of our experimental evaluation are presented in the appendix of the full version of this paper, available at `http://informatik.uni-bremen.de/~clu/combined/`

## 2 Preliminaries

We introduce DL-Lite$_\mathcal{R}$-TBoxes, ABoxes, and conjunctive queries. Let $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ be countably infinite sets of *individual names*, *concept names* and *role names*. *Roles* $R$, *simple concepts* $C$, and *concepts* $D$ are built according to the following syntax rules, where $P$ ranges over $\mathsf{N_R}$ and $A$ over $\mathsf{N_C}$:

$$R ::= P \mid P^-, \qquad C ::= A \mid \exists R, \qquad D ::= C \mid \neg C \mid \exists R.A.$$

As usual, we use $\mathsf{N_R^-}$ to denote the set of all roles and identify $(P^-)^-$ with $P$. In DL-Lite$_\mathcal{R}$, a *TBox* is a finite set $\mathcal{T}$ of *concept inclusions (CIs)* $C \sqsubseteq D$ with $C$ a simple concept and $D$ a concept, and *role inclusions (RIs)* $R_1 \sqsubseteq R_2$ with $R_1, R_2$ roles.[1]

An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $P(a, b)$, where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. We denote by $\mathsf{Ind}(\mathcal{A})$ the set of individual names that occur in $\mathcal{A}$, and write $P^-(a, b) \in \mathcal{A}$ instead of $P(b, a) \in \mathcal{A}$ whenever convenient. A *knowledge base* (*KB*) is a pair $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox.

The semantics of TBoxes and ABoxes is defined in the standard way based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty *domain* and $\cdot^\mathcal{I}$ an *interpretation function* that maps each $A \in \mathsf{N_C}$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, each $P \in \mathsf{N_R}$ to a relation $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and each $a \in \mathsf{N_I}$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$; for details consult [1, 3]. An interpretation is a *model* of a TBox $\mathcal{T}$ if it satisfies all inclusions in $\mathcal{T}$; models of ABoxes and knowledge bases are defined analogously. A knowledge base is *consistent* if it has a model. For a CI or RI $\alpha$, we write $\mathcal{T} \models \alpha$ when $\alpha$ is a consequence of $\mathcal{T}$ (satisfied in all models of $\mathcal{T}$). Instead of

[1] A set of role inclusions is also called a *role hierarchy*.

$\mathcal{T} \models R \sqsubseteq S$, we will usually write $R \sqsubseteq^*_{\mathcal{T}} S$ to clearly distinguish consequences of this form (which are RIs) from consequences of the form $\mathcal{T} \models \exists R \sqsubseteq \exists S$ (which are CIs). Note that, in DL-Lite$_{\mathcal{R}}$, deciding consistency and logical consequence amounts to computing a form of transitive closure [3].

Let $\mathsf{N_V}$ be a countably infinite set of *variables*. Taken together, the sets $\mathsf{N_V}$ and $\mathsf{N_I}$ form the set $\mathsf{N_T}$ of *terms*. A *first-order* (*FO*) *query* is a first-order formula $q = \varphi(\boldsymbol{x})$ in the signature $\mathsf{N_C} \cup \mathsf{N_R}$ and with terms from $\mathsf{N_T}$, where the concept and role names are treated as unary and binary predicates, respectively, and $\boldsymbol{x}$ are the free variables of $\varphi$ called the *answer variables*; we say that $q$ is *$k$-ary* if $\boldsymbol{x}$ comprises $k$ variables. If $k = 0$, then $q$ is a *Boolean* query. A *conjunctive query (CQ)* is an FO query of the form $q = \exists \boldsymbol{y}\, \psi(\boldsymbol{y}, \boldsymbol{x})$, where $\psi$ is a conjunction of *concept atoms* $A(t)$ and *role atoms* $P(t, t')$ where $t, t' \in \mathsf{N_T}$. As in the case of ABox assertions, we do not distinguish between $P^-(t, t')$ and $P(t', t)$. We denote by $\mathsf{term}(q)$ the set of terms in $q$.

Let $q = \varphi(\boldsymbol{x})$ be a $k$-ary FO query with $\boldsymbol{x} = x_1, \dots, x_k$, and $\mathcal{I}$ an interpretation. A mapping $\pi \colon \mathsf{term}(q) \to \Delta^{\mathcal{I}}$ with $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{term}(q) \cap \mathsf{N_I}$ is a *match* for $q$ in $\mathcal{I}$ if $\mathcal{I}$ satisfies $q$ under the variable assignment that maps each $t \in \mathsf{term}(q)$ to $\pi(t)$; in this case, we write $\mathcal{I} \models^{\pi} q$. For a $k$-tuple of individual names $\boldsymbol{a} = a_1, \dots, a_k$, a match $\pi$ for $q$ in $\mathcal{I}$ is an *$\boldsymbol{a}$-match* if $\pi(x_i) = a_i^{\mathcal{I}}$ for $i \leq k$. We say that $\boldsymbol{a}$ is an *answer* to $q$ in an interpretation $\mathcal{I}$ if there is an $\boldsymbol{a}$-match for $q$ in $\mathcal{I}$ and use $\mathsf{ans}(q, \mathcal{I})$ to denote the set of all answers to $q$ in $\mathcal{I}$. Finally, $\boldsymbol{a}$ is a *certain answer* to $q$ over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$ and $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$. The set of all certain answers to $q$ over $\mathcal{K}$ is denoted by $\mathsf{cert}(q, \mathcal{K})$. The query answering problem considered in this paper is: given a DL-Lite$_{\mathcal{R}}$ knowledge base $\mathcal{K}$ and a CQ $q$, compute $\mathsf{cert}(q, \mathcal{K})$.

To simplify notation, throughout the paper we adopt the *unique name assumption* (UNA), i.e., require that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for distinct $a, b \in \mathsf{N_I}$. This assumption has no impact on the query answering problem.

## 3 ABox Completion

As explained in the introduction, the central idea of the combined approach is to materialize consequences of the TBox in the ABox as a preprocessing step, and then to execute queries over the completed data stored in an RDBMS as a plain table. We illustrate this using two examples from the university domain, similar in spirit to the LUBM ontology from [6] used in the experimental evaluation.

*Example 1. For any ABox $\mathcal{A}$, the concept inclusions*

$$\mathsf{Student} \sqsubseteq \mathsf{Person} \tag{1}$$

$$\mathsf{Student} \sqsubseteq \exists \mathsf{takesCourse} \tag{2}$$

*lead to the following additions: (1) for every assertion $\mathsf{Student}(a) \in \mathcal{A}$, add (1) $\mathsf{Person}(a)$ and (2) $\mathsf{takesCourse}(a, b)$ for some fresh individual $b$ (unless such assertions are already present). After this completion, a CQ such as*

$$q_1(x) = \exists y\, \mathsf{Person}(x) \wedge \mathsf{takesCourse}(x, y)$$

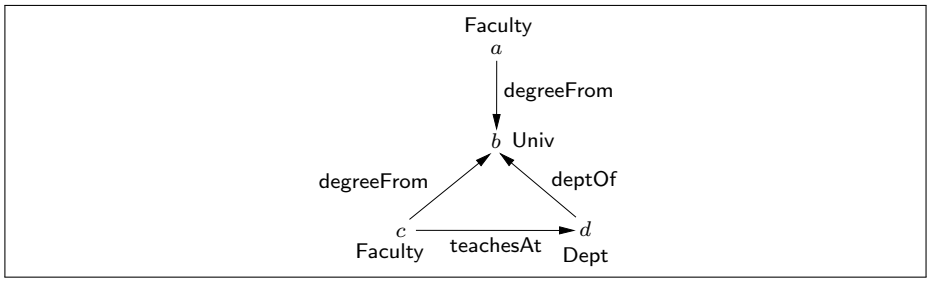


**Fig. 1.** Completed ABox for Example 3.

*correctly returns each $a$ with* $\mathsf{Student}(a) \in \mathcal{A}$ *as a certain answer.*

The following example shows that naive completion can result in infinite ABoxes.

*Example 2. Completed naively, the ABox* $\{\mathsf{Faculty}(a)\}$ *and LUBM inclusions*

$$\mathsf{Faculty} \sqsubseteq \exists \mathsf{degreeFrom} \qquad \exists \mathsf{degreeFrom}^- \sqsubseteq \mathsf{Univ} \tag{3}$$

$$\mathsf{Univ} \sqsubseteq \exists \mathsf{deptOf}^- \qquad \exists \mathsf{deptOf} \sqsubseteq \mathsf{Dept} \tag{4}$$

$$\mathsf{Dept} \sqsubseteq \exists \mathsf{teachesAt}^- \qquad \exists \mathsf{teachesAt} \sqsubseteq \mathsf{Faculty} \tag{5}$$

*result in an infinite role chain that indefinitely repeats the roles* $\mathsf{degreeFrom}$, $\mathsf{deptOf}^-$, *and* $\mathsf{teachesAt}^-$.

The problem can be overcome by reusing fresh individuals in an economic way.

*Example 3. Consider again the TBox (3)-(5). By reusing individuals, the ABox* $\{\mathsf{Faculty}(a)\}$ *can be completed as shown in Figure 1, replacing the infinite role chain with a cycle. Individual reuse compromises soundness of query answering as some queries now have spurious answers; for example, the CQ*

$$q_2(x) = \exists y, z\ \mathsf{Faculty}(x) \land \mathsf{degreeFrom}(x, y) \land \mathsf{Univ}(y) \land \mathsf{deptOf}(z, y) \land \mathsf{Dept}(z) \land \mathsf{teachesAt}(x, z)$$

*returns $c$ as an answer when executed over the completed ABox shown in Figure 1. This answer is spurious for two reasons: first, the cycle in Figure 1 is present only due to individual reuse and thus should be disregarded for query matches; and second, the freshly introduced individuals $b, c, d$ are 'labeled nulls' and thus can never be returned as answers.*

To recover soundness, it is necessary to eliminate the spurious answers. In the original combined approach, this was achieved by query rewriting [8, 9]. In this paper, the spurious answers are eliminated by a filtering procedure that is installed as a user-defined function in the RDBMS. In the remainder of this section, we introduce ABox completion in full detail. In the subsequent section, we describe the filtering procedure.

From a conceptual perspective, the ABox completion step can be viewed as replacing the original ABox with the *canonical model* $\mathcal{I}_\mathcal{K}$ of the knowledge

base $\mathcal{K}$ [8]. To define $\mathcal{I}_\mathcal{K}$, we need a few preliminaries. From now on, we will generally disallow concepts of the form $\exists R.C$. This can be done without loss of generality since each CI $D \sqsubseteq \exists R.C$ can be replaced with $D \sqsubseteq \exists R_C$, $R_C \sqsubseteq R$, and $\exists R_C^- \sqsubseteq C$, where $R_C$ is a fresh role name.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_\mathcal{R}$ KB. We use $\mathsf{rol}(\mathcal{T})$ to denote the set of all role names in $\mathcal{T}$ plus their inverses. The canonical model comprises at most two fresh individuals for every role in $\mathsf{rol}(\mathcal{T})$. However, we only want to introduce the fresh individuals for a given role when necessary. Formally, we call a role $R \in \mathsf{rol}(\mathcal{T})$ *generating in* $\mathcal{T}$ if there exist an $a \in \mathsf{Ind}(\mathcal{A})$ and $R_0, \dots, R_n \in \mathsf{rol}(\mathcal{T})$ such that $R_n = R$ and the following conditions hold:

**(agen)** $\mathcal{K} \models \exists R_0(a)$ and $R_0(a, b) \notin \mathcal{A}$ for all $b \in \mathsf{Ind}(\mathcal{A})$ (written $a \rightsquigarrow \exists R_0$),

**(rgen)** for $i < n$, $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$ (written $\exists R_i^- \rightsquigarrow \exists R_{i+1}$).

To facilitate the implementation of efficient filters, we refine the definition of canonical models as given in [8]: in some cases, we introduce two fresh individuals for a given role instead of only a single one. The need for the additional individual is related to particular role configurations in the TBox called a loop: a set $\{R, S\} \subseteq \mathsf{rol}(\mathcal{T})$ (where potentially $R = S$) is a *loop in* $\mathcal{T}$ if $R \neq S^-$, $\mathcal{T} \models \exists R^- \sqsubseteq \exists S$, $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$, and there is some $T \in \mathsf{rol}(\mathcal{T})$ such that $S^- \sqsubseteq^*_\mathcal{T} T$ and $R \sqsubseteq^*_\mathcal{T} T$. Let $\mathfrak{L}_\mathcal{T}$ denote the set of all roles that occur in a loop in $\mathcal{T}$. The canonical model $\mathcal{I}_\mathcal{K}$ is then based on the domain

$$\begin{aligned}\Delta^{\mathcal{I}_\mathcal{K}} = \mathsf{Ind}(\mathcal{A}) \cup \{c_{R,0} \mid R \in \mathsf{rol}(\mathcal{T}) \setminus \mathfrak{L}_\mathcal{T} \text{ is generating in } \mathcal{K}\} \\ \cup \{c_{R,0}, c_{R,1} \mid R \in \mathfrak{L}_\mathcal{T} \text{ is generating in } \mathcal{K}\}.\end{aligned}$$

To define the extension of roles in $\mathcal{I}_\mathcal{K}$, we need some additional preparation. Let "$\prec$" be an arbitrary, but fixed total ordering on $\mathsf{rol}(\mathcal{T})$. For all $d, d' \in \Delta^{\mathcal{I}_\mathcal{K}}$ and each role $R$, we write $d \rightsquigarrow_R d'$ whenever there is an $S$ such that $S \sqsubseteq^*_\mathcal{T} R$ and one of the following cases applies:

- $d = a \in \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow \exists S$, and $d' = c_{S,0}$;
- $d = c_{T,i}$, $\exists T^- \rightsquigarrow \exists S$, $d' = c_{S,j}$, and one of the following holds
  - $i = j$ and $\{S, T\}$ is not a loop in $\mathcal{T}$;
  - $i = j$, $\{S, T\}$ is a loop in $\mathcal{T}$, and $S \prec T$;
  - $i = \overline{j}$, $\{S, T\}$ is a loop in $\mathcal{T}$, and $T = S$ or $T \prec S$ (for $\overline{0} = 1$ and $\overline{1} = 0$).

The *canonical model* $\mathcal{I}_\mathcal{K}$ *for* $\mathcal{K}$ is now defined as follows, based on the domain $\Delta^{\mathcal{I}_\mathcal{K}}$ introduced above:

$$\begin{aligned}A^{\mathcal{I}_\mathcal{K}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{c_{R,i} \in \Delta^{\mathcal{I}_\mathcal{K}} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\}, \\ R^{\mathcal{I}_\mathcal{K}} &= \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \exists S : S(a, b) \in \mathcal{A} \text{ and } S \sqsubseteq^*_\mathcal{T} R\} \cup \\ &\quad \{(d, d') \in \Delta^{\mathcal{I}_\mathcal{K}} \mid d \rightsquigarrow_R d' \text{ or } d' \rightsquigarrow_{R^-} d\}, \\ a^{\mathcal{I}_\mathcal{K}} &= a.\end{aligned}$$

The ABox completion consists of replacing the ABox $\mathcal{A}$ originally stored in the RDBMS with its canonical model $\mathcal{I}_\mathcal{K}$. This can be achieved by executing a set of FO/SQL-queries whose size is polynomial in the size of $\mathcal{T}$ [8].

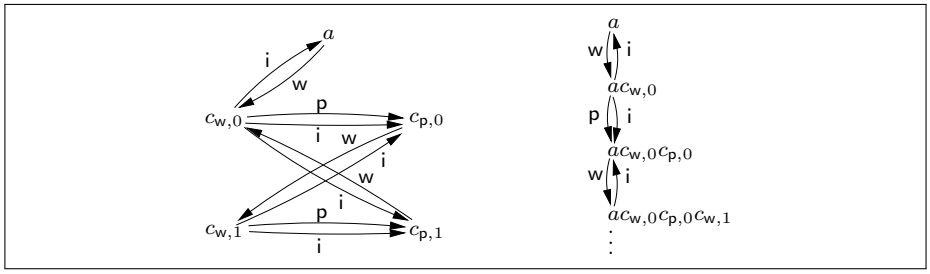


**Fig. 2.** Canonical model $\mathcal{I}_\mathcal{K}$ and unraveled canonical model $\mathcal{U}_\mathcal{K}$ for Example 5.

It can be shown that $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$ whenever $\mathcal{K}$ is consistent. Note that one can find a Boolean CQ $q_\mathcal{T}$ of size polynomial in the size of $\mathcal{T}$ such that for any ABox $\mathcal{A}$ stored in the RDBMS, $q_\mathcal{T}$ gives a positive answer iff $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent [8]. We can thus safely assume that the knowledge base has been tested for consistency before query answering.

*Example 4. Reconsider Examples 2 and 3. The canonical model $\mathcal{I}_\mathcal{K}$ for the ABox $\{\mathsf{Faculty}(a)\}$ and TBox (3)-(5) is the structure displayed in Figure 1. Following our construction above, the fresh individuals $b, c, d$ are named $c_{\mathsf{degreeFrom},0}$, $c_{\mathsf{teachesAt}^-,0}$, and $c_{\mathsf{deptOf}^-,0}$. Note that the TBox (3)-(5) does not give rise to any loops, and thus all $c_{R,i}$ have index $i = 0$.*

*Example 5. The following TBox gives rise to the loop $\{\mathsf{worksFor}, \mathsf{paysSalaryOf}\}$:*

$$\mathsf{Employee} \sqsubseteq \exists \mathsf{worksFor} \qquad \exists \mathsf{worksFor}^- \sqsubseteq \mathsf{Employer} \tag{6}$$

$$\mathsf{Employer} \sqsubseteq \exists \mathsf{paysSalaryOf} \qquad \exists \mathsf{paysSalaryOf}^- \sqsubseteq \mathsf{Employee} \tag{7}$$

$$\mathsf{worksFor}^- \sqsubseteq \mathsf{isAffiliatedWith} \qquad \mathsf{paysSalaryOf} \sqsubseteq \mathsf{isAffiliatedWith}. \tag{8}$$

*The canonical model $\mathcal{I}_\mathcal{K}$ for the ABox $\{\mathsf{Employee}(a)\}$ and the TBox (6)-(8) with $\mathsf{paysSalaryOf} \prec \mathsf{worksFor}$ is shown on the left-hand side of Figure 2, where concept names are omitted and role names are abbreviated by their first letter.*

Note that a more straightforward version of the canonical model could be obtained by identifying all elements $c_{R,0}$ and $c_{R,1}$. Section 4 explains why this leads to problems for efficient filtering. Also note that real world ontologies seem to contain only very few loops, if any, and thus having two domain elements per role that participates in a loop should not significantly increase the size of canonical models in practical cases.

To characterize the spurious answers that have to be filtered out, it is useful to introduce an unraveled (infinite) version of canonical models. Let $\mathcal{K}$ be a knowledge base. A *path* is a finite sequence $ad_1 \cdots d_n$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$, $d_1, \ldots, d_n \in \Delta^{\mathcal{I}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow_R d_1$ for some $R \in \mathsf{N}_\mathsf{R}^-$, and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N}_\mathsf{R}^-$, $1 \leq i < n$. We denote by $\mathsf{tail}(\sigma)$ the last element of the path $\sigma$.

The unraveled canonical model $\mathcal{U}_\mathcal{K}$ is then defined by taking:

$$\begin{aligned} \Delta^{\mathcal{U}_\mathcal{K}} &\text{ is the set of all paths in } \mathcal{I}_\mathcal{K}, \\ a^{\mathcal{U}_\mathcal{K}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}), \\ A^{\mathcal{U}_\mathcal{K}} &= \{\sigma \in \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathsf{tail}(\sigma) \in A^{\mathcal{I}_\mathcal{K}}\}, \end{aligned}$$

$$\begin{aligned} R^{\mathcal{U}_\mathcal{K}} = {} & \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \exists S : S(a,b) \in \mathcal{A} \text{ and } S \sqsubseteq^*_\mathcal{T} R\} \cup \\ & \{(\sigma, \sigma d) \mid \sigma d \in \Delta^{\mathcal{U}_\mathcal{K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_R d\} \cup \\ & \{(\sigma d, \sigma) \mid \sigma d \in \Delta^{\mathcal{U}_\mathcal{K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_{R^-} d\}. \end{aligned}$$

As an example, the canonical model $\mathcal{U}_\mathcal{K}$ for the KB from Example 4 is shown on the right-hand side of Figure 2. The following result shows that, as one would expect, $\mathcal{U}_\mathcal{K}$ does not suffer from spurious answers.

**Theorem 1.** *For every consistent DL-Lite$_\mathcal{R}$-KB $\mathcal{K}$ and every CQ $q$, we have* $\mathsf{cert}(q,\mathcal{K}) = \mathsf{ans}(q,\mathcal{U}_\mathcal{K})$.

The proof of Theorem 1 is standard and omitted, see [8] for a similar proof.

## 4 Filtering

To remove spurious answers, we install a filtering procedure as a user-defined function in the RDBMS. The procedure takes as input a match of the query in the canonical model $\mathcal{I}_\mathcal{K}$ stored in the RDBMS and returns "false" if this match is spurious and "true" otherwise. We assume that the filtering procedure has access to the query and the TBox, but not to the data. To define its behavior more precisely, we formally define spurious matches based on unraveled canonical models $\mathcal{U}_\mathcal{K}$ and Theorem 1.

Let $\mathcal{K}$ be a KB and $q(\boldsymbol{x})$ a CQ. A match $\pi$ of $q$ in $\mathcal{I}_\mathcal{K}$ *is reproduced by* a match $\tau$ of $q$ in $\mathcal{U}_\mathcal{K}$ if for all $t \in \mathsf{term}(q)$, we have $\pi(t) = \mathsf{tail}(\tau(t))$. We say that $\pi$ is *spurious* if it is not reproduced by any match $\tau$ of $q$ in $\mathcal{U}_\mathcal{K}$. The following lemma, which is an immediate consequence of Theorem 1, shows that $\mathcal{I}_\mathcal{K}$ can be used for query answering when spurious matches are filtered out.

**Lemma 1.** $\boldsymbol{a} \in \mathsf{cert}(q,\mathcal{K})$ *iff there is an $\boldsymbol{a}$-match $\pi$ of $q$ in $\mathcal{I}_\mathcal{K}$ that is not spurious.*

We want to show that it can be decided in time polynomial in the size of $q$ and $\mathcal{T}$ whether a given match in $\mathcal{I}_\mathcal{K}$ is spurious. Clearly, it is enough to test for each maximally connected component of $q$ whether the given match is spurious on that component. We can thus w.l.o.g. assume that $q$ is connected.

We need a few preliminaries. An *anonymous path* is a path without the leading individual name, i.e., it is a finite sequence $d_1 \cdots d_n$, $n \geq 1$, such that $d_1, \ldots, d_n \in \Delta^{\mathcal{I}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N}_\mathsf{R}^-$, $1 \leq i < n$. We use $\mathsf{Paths}$ to denote the set of all paths, both anonymous and non-anonymous. A *root configuration for $q$ given $\pi$* is a set $\rho \subseteq \mathsf{term}(q)$ such that one of the following conditions is true:

- $\rho$ is the set of those $t \in \mathsf{term}(q)$ such that $\pi(t) \in \mathsf{N}_\mathsf{I}$ and this set is non-empty;

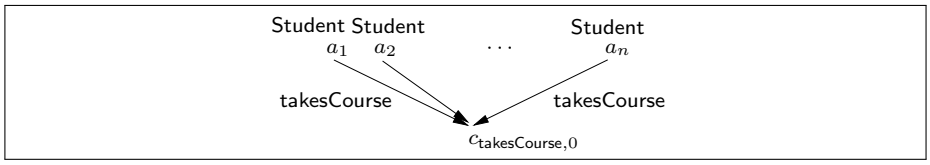


**Fig. 3.** Canonical model $\mathcal{I}_\mathcal{K}$ for Example 6.

– the above set is empty and $\rho$ contains exactly one term (actually a variable).

The filtering procedure first checks whether all answer variables are mapped to elements of $\mathsf{Ind}(\mathcal{A})$, i.e., individuals of the original ABox $\mathcal{A}$. If this is not the case, it immediately returns "false". Then the procedure iterates through all root configurations $\rho$. For each $\rho$, it constructs a sequence $S^0_\rho, S^1_\rho, \dots$ of relations $S^i_\rho \subseteq \mathsf{term}(q) \times \mathsf{Paths}$ as follows:

– $S^0_\rho$ contains all pairs $(t, \pi(t))$ with $t \in \rho$;
– $S^{i+1}_\rho$ is $S^i_\rho$ extended with the following pairs:
  (a) $(t, \sigma\pi(t))$ for all $R(s,t) \in q$ with $(s,\sigma) \in S^i_\rho$ and $\pi(s) \rightsquigarrow_R \pi(t)$;
  (b) $(t, \sigma\pi(t))$ for all $R(s,t) \in q$ with $(s, \sigma\pi(t)\pi(s)) \in S^i_\rho$ and $\pi(t) \rightsquigarrow_{R^-} \pi(s)$.

The computation stops as soon as the sequence stabilizes or $S^i_\rho$ becomes non-functional which happens after at most $|\mathsf{term}(q)|$ iterations. The procedure returns "true" if the final $S^i_\rho$ is a function with domain $\mathsf{term}(q)$ for some root configuration $\rho$, and "false" otherwise.

*Example 6. Consider the TBox (1)-(2) from Example 1, the query*

$$q_3(x,y) = \exists z\, \mathsf{Student}(x) \wedge \mathsf{Student}(y) \wedge \mathsf{takesCourse}(x,z) \wedge \mathsf{takesCourse}(y,z), \quad (9)$$

*and the ABox*

$$\{\mathsf{Student}(a_1), \dots, \mathsf{Student}(a_n)\}. \quad (10)$$

*The canonical model $\mathcal{I}_\mathcal{K}$ is shown in Figure 3. Suppose the filter gets as input the match $\pi = \{x \mapsto a_1, y \mapsto a_2, z \mapsto c_{\mathsf{takesCourse},0}\}$. There is only one possible root configuration for $\pi$, which is $\rho = \{x, y\}$. The procedure computes*

$$S_\rho = \{(x, a_1), (y, a_2), (z, a_1 c_{\mathsf{takesCourse},0}), (z, a_2 c_{\mathsf{takesCourse},0})\}$$

*which is not a function; thus, the match is identified as spurious and "false" is returned.*

*Example 7. Consider the ABox $\{\mathsf{Faculty}(a)\}$, TBox (3)-(5), and query $q_2$ from Example 3. To make things a bit more interesting, assume that $x$ is a quantified variable in $q_2$ rather than an answer variable. Recall that the canonical model $\mathcal{I}_\mathcal{K}$ is shown in Figure 1, modulo the names of fresh individuals. Given the match $\pi = \{x \mapsto c, y \mapsto b, z \mapsto d\}$ and considering the root configuration $\rho = \{x\}$, the procedure computes*

$$S_\rho = \{(x, c), (y, cb), (z, cbd), (x, cbdc)\}$$

*and stops because of non-functionality. For the other root configurations $\rho = \{y\}$ and $\rho = \{z\}$, the procedure fails in a similar way and thus returns "false".*

Similar to the "tree witnesses" from [8], the filtering procedure follows a simple idea for reproducing the input match $\pi$ in $\mathcal{I}_\mathcal{K}$ as a match $\tau$ in $\mathcal{U}_\mathcal{K}$: when we have already decided that $\tau(x) = \sigma \notin \mathsf{Ind}(\mathcal{A})$ and $R(x, y) \in q$, then there is a *uniquely determined* individual $\sigma'$ to which $y$ can be matched. This follows from requiring $\pi(y) = \mathsf{tail}(\tau(y))$ and the following property of $\mathcal{U}_\mathcal{K}$:

$$\text{if } (\sigma, \sigma') \in R^{\mathcal{U}_\mathcal{K}} \text{ and } (\sigma, \sigma'') \in R^{\mathcal{U}_\mathcal{K}} \text{ with } \sigma' \neq \sigma'', \text{ then } \mathsf{tail}(\sigma') \neq \mathsf{tail}(\sigma'').$$

In fact, it is this determinism of matches that is made explicit by Conditions (a) and (b) of the filtering procedure. Note that, without introducing two individual names $c_{R,0}$ and $c_{R,1}$ whenever $R$ is involved in a loop, the above crucial property of $\mathcal{U}_\mathcal{K}$ fails. In fact, we do not know whether polytime filtering is possible based on the variation of the canonical model where all individuals $c_{R,0}$ and $c_{R,1}$ are identified. The problem is illustrated by the following example.

*Example 8. Consider the ABox* $\{\mathsf{Employee}(a)\}$ *and TBox (6)-(8) from Example 5 and the CQ*

$$q_4(x) = \exists y, z, u\, \mathsf{w}(x, y) \land \mathsf{p}(y, z) \land \mathsf{i}(u, z).$$

*Let* $\pi = \{x \mapsto a, y \mapsto c_{\mathsf{w},0}, z \mapsto c_{\mathsf{p},0}, u \mapsto c_{\mathsf{w},1}\}$*. The only root configuration is* $\rho = \{x\}$*. During the first two iterations, the filtering procedure produces*

$$S^2_\rho = \{(x, a), (y, ac_{\mathsf{w},0}), (z, ac_{\mathsf{w},0}c_{\mathsf{p},0})\}.$$

$S^2_\rho$ *says that* $z$ *has to be mapped to* $ac_{\mathsf{w},0}c_{\mathsf{p},0}$*. Due to the atom* $\mathsf{i}(z, u) \in q_4$ *and the two* $\mathsf{i}$*-edges outgoing from* $z$ *in* $\mathcal{U}_\mathcal{K}$*, the possible targets for* $u$ *are* $ac_{\mathsf{w},0}$ *and* $ac_{\mathsf{w},1}$*. However, to produce a match in* $\mathcal{U}_\mathcal{K}$ *that is compatible with* $\pi$*, we can only choose a target that ends with* $\pi(u) = c_{\mathsf{w},1}$ *and obtain*

$$S^3_\rho = \{(x, a), (y, ac_{\mathsf{w},0}), (z, ac_{\mathsf{w},0}c_{\mathsf{p},0}), (z, ac_{\mathsf{w},0}c_{\mathsf{p},0}c_{\mathsf{w},1})\}$$

*which is functional, showing that the match* $\pi$ *is not spurious. In a canonical model* $\mathcal{I}_\mathcal{K}$ *where* $c_{\mathsf{w},0}$ *and* $ac_{\mathsf{w},1}$ *are identified, there are indeed two choices for mapping of* $u$*. This makes it non-obvious how to find a polytime filtering procedure in this case, if one exists at all.*

We now analyze the runtime and correctness of the filtering procedure. First note that, in Conditions (a) and (b), the filtering procedure has to check whether $\pi(s) \rightsquigarrow_R \pi(t)$ and $\pi(t) \rightsquigarrow_{R^-} \pi(s)$, respectively. As required, both conditions can be tested without access to the ABox $\mathcal{A}$. For example, in Condition (a) we have:

- if $\pi(t) \in \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ does not hold and checking whether $\pi(t) \in \mathsf{Ind}(\mathcal{A})$ requires only to check whether or not $\pi(t)$ is of the form $c_{R,i}$;
- if $\pi(s) \in \mathsf{Ind}(\mathcal{A})$ and $\pi(t) \notin \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ holds by the construction of $\mathcal{I}_\mathcal{K}$ since $\pi$ is a match of $q$ in $\mathcal{I}_\mathcal{K}$, and;
- if $\pi(s) \notin \mathsf{Ind}(\mathcal{A})$ and $\pi(t) \notin \mathsf{Ind}(\mathcal{A})$, then $\pi(s) \rightsquigarrow_R \pi(t)$ can be checked by using only $\pi$ and $\mathcal{T}$ based on the definition of "$\rightsquigarrow_R$".

It is not hard to see that the algorithm runs in polynomial time. The runtime is quadratic in the size of $q$ because we first have to iterate over all root configurations $\rho$ and then need to compute $S_\rho$, essentially a breadth-first search

of (the graph of) $q$. We conjecture that iterating over all root configurations is avoidable at the cost of a less transparent filtering procedure, improving the runtime to linear in the size of $q$. The runtime also depends on $\mathcal{T}$ as checking the applicability of Conditions (a) and (b) involves testing consequences of the forms $\mathcal{T} \models \exists R \sqsubseteq \exists S$ and $S \sqsubseteq^*_{\mathcal{T}} R$. Since it is efficient to pre-compute all these consequences in practical cases, this amounts to a simple lookup.

The following lemma asserts correctness of the filtering procedure. It is proved in Appendix A of the full version.

**Lemma 2.** *Given a match $\pi$ of $q$ in $\mathcal{I}_\mathcal{K}$, the filtering procedure returns "true" iff $\pi$ is not spurious.*

## 5 Experiments

We carry out an experimental evaluation to analyze the performance of the combined approach with filtering, based on the IBM DB2 relational database system. We use a modified version of the ontology from the Lehigh University Benchmark (LUBM) [6] and produce test ABoxes using a modified version of the LUBM data generator. We execute five queries from the LUBM suite as well as six handcrafted ones that were designed to test the proposed approach more fully. Note that LUBM was not specifically designed for evaluating OBDA with DL-Lite$_\mathcal{R}$ and in its original form is not too useful for this purpose, for reasons explained in more detail below. Our modifications of the LUBM ontology, data generator, and query set all aim at making the LUBM suite more realistic for OBDA evaluation. We believe that this material might be interesting also for future experiments and provide it online at `http://informatik.uni-bremen.de/~clu/combined/`.

### 5.1 Ontology, Data, and Queries

The LUBM ontology comprises 42 concept names and 25 role names and is formulated in the description logic $\mathcal{ELI}$ extended with transitive roles, role hierarchies, and datatypes. The TBox contains concept inclusions of the form $A \sqsubseteq C$, concept definitions $A \equiv C$ as abbreviations for $A \sqsubseteq C$, $C \sqsubseteq A$, and domain and range restrictions of the form $\exists R \sqsubseteq A$ and $\exists R^- \sqsubseteq A$. We converted this ontology to DL-Lite$_\mathcal{R}$ by dropping all datatypes, treating the only transitive role subOrganizationOf as a standard role, replacing concept equations $A \equiv C$ with $A \sqsubseteq C$, and breaking up conjunctions $A \sqsubseteq C_1 \sqcap C_2$ into $A \sqsubseteq C_1, A \sqsubseteq C_2$.

While the resulting TBox is formulated in DL-Lite$_\mathcal{R}$ as required, it is only moderately interesting for evaluating query answering techniques: first, there is a lack of existential restrictions $\exists R$ and $\exists R.C$ on the right-hand side of concept inclusions, which leads to extremely few fresh *anonymous* individuals being generated during the ABox completion, and consequently to very few role edges between those individuals (from now on, we call this part of the canonical model $\mathcal{I}_\mathcal{K}$ the *anonymous part*); second, the overall size of the TBox is too small to be representative for real-world ontologies. To attenuate these deficiencies while still

being able to use the LUBM data generator, we extended the DL-Lite$_\mathcal{R}$-version of LUBM in two directions:

(1) We added 26 concept inclusions, many of which have existential restrictions on the right-hand side, to generate a more interesting anonymous part of canonical models. A complete list of these CIs can be found in Appendix B of the full version of this paper.

(2) With reasonable effort, it does not seem possible to significantly increase the size of LUBM (to hundreds or thousands of concepts) while retaining a careful modeling. One particularly unrealistic aspect of LUBM and a striking difference to more comprehensive ontologies is its limited concept hierarchy, where each concept has only very few subconcepts. To alleviate this shortcoming, we added subconcepts to each of the LUBM concepts Course, Department, Professor, and Student by introducing subject areas, such as MathCourse, BioCourse, and CSCourse for courses, MathProfessor, BioProfessor for professors, etc.

We call the resulting TBox LUBM$_n^\exists$ with $n$ indicating the number of subconcepts introduced in Point 2 above (20 by default). For example, LUBM$_{20}^\exists$ contains 106 concept names and 27 role names.

To generate ABoxes, we use the LUBM Data Generator (UBA) version 1.7, modified so as to complement our modifications to the TBox. Specifically, the original UBA generates data that is complete w.r.t. existential restrictions in the LUBM ontology: it produces ABoxes $\mathcal{A}$ such that for every assertion $A(a) \in \mathcal{A}$ and CI $A \sqsubseteq \exists R$ (and $A \sqsubseteq \exists R.B$) in LUBM$_n^\exists$, there is already an $r$-successor of $a$ in $\mathcal{A}$. Our modifications introduce a controlled amount of incompleteness: the modified data generator takes a probability $p$ as a parameter and, in selected parts of the data, drops generated role assertions with probability $p$. More information can be found in Appendix C of the full version. The second modification of the data generator is linked to the subconcepts introduced in Point 2 above. Whenever the original generator produces an instance $a$ of Student, the new generator randomly chooses a value between 1 and $n$ and generates an assertion for the $i$-th subject, Subj$i$Student$(a)$; similarly for Course, Department, and Professor.

The main aim of our experiments is to show that our approach is feasible on realistic ontologies, data, and queries. Additionally, we also provide a preliminary comparison with the query rewriting approach, using the Requiem tool for producing those rewritings [10]. We use 11 queries, six of which we have handcrafted specifically for our experiments and five originating from the evaluation of Requiem presented in [10]. The latter queries are extremely simple and do in most cases neither pose a serious challenge for the filtering approach nor for pure rewriting. The former are shown in Figure 4. Note that $\mathsf{q}_3$ is very similar to the query discussed in Examples 3, 4, and 7; and is designed in such a way that spurious cycles in the anonymous part of canonical models produce spurious matches that have to be filtered out. Query $\mathsf{q}_2$ is essentially the query discussed in Example 6 and is designed to stress-test the filtering approach: based on the data generation scheme, it is expected to produce a very large number of spurious answers.

```
q1(x,y) <- Student(x), takesCourse(x,z), Course(z), teacherOf(y,z),
           Faculty(y), worksFor(y,u), Department(u), memberOf(x,u)
q2(x,y) <- Subj3Student(x), Subj4Student(y),
           takesCourse(x,z), takesCourse(y,z)
q3(x)   <- Faculty(x), degreeFrom(x,y), University(y),
           subOrganizationOf(z,y), Department(z), memberOf(x,z)
q4(x,y) <- Subj3Department(x), Subj4Department(y),
           Professor(z), memberOf(z,x), publicationAuthor(u,z),
           Professor(v), memberOf(v,y), publicationAuthor(u,v)
q5(x)   <- Publication(x), publicationAuthor(x,y), Professor(y),
           publicationAuthor(x,z), Student(z)
q6(x,y) <- University(x), University(y), memberOf(z,x), Student(z),
           memberOf(u,y), Professor(u), advisor(z,u)
```

**Fig. 4.** Queries $\mathtt{q}_1$ to $\mathtt{q}_6$.

| Universities | CA | CA (compl) | RA | RA (compl) | Inds | Inds (compl) |
|---|---|---|---|---|---|---|
| 10 | 373K | 636K | 593K | 1.3M | 201K | 201K |
| 25 | 984K | 1.6M | 1.5M | 3.6M | 528K | 528K |
| 50 | 1.9M | 3.3M | 3.1M | 7.2M | 1M | 1M |
| 75 | 3M | 5.1M | 4.7M | 10.9M | 1.6M | 1.6M |
| 100 | 4M | 6.8M | 6.3M | 14.6M | 2.1M | 2.1M |
| 125 | 5M | 8.5M | 7.9M | 18M | 2.7M | 2.7M |
| 150 | 6M | 10.1M | 9.5M | 21.8M | 3.2M | 3.2M |
| 200 | 8M | 13.5M | 12.6M | 29M | 4.3M | 4.3M |

**Fig. 5.** Number of concepts, roles, and individuals in original and completed ABoxes.

### 5.2 Results

We report on three experiments, in each experiment varying a different parameter: in experiment one, we vary the size of the ABox via the number of universities generated by the data generator; in experiment two, we vary the degree of incompleteness of the data; and in experiment three, we vary the number of subclasses, i.e., the parameter $n$ of the ontology $\text{LUBM}_n^{\exists}$. All experiments were carried out on a Linux (3.2.0) machine with a 3.5Ghz quad-core processor and 8GB of RAM, using IBM DB2 version 9.7.5.

The size of the test data is detailed in Figure 5 and query execution times for the first experiment are reported in Figure 7. Here we use 5% incompleteness of the data and $n = 20$ subclasses in $\text{LUBM}_n^{\exists}$. The orange curves are for the combined approach with filtering while the blue ones indicate the pure rewriting approach for those cases in which Requiem succeeded generating a rewriting. Using the combined approach with filtering, all queries were answered within very reasonable time. To better understand the results, it is interesting to consider the number of spurious and valid answers for each query shown in Figure 6 for the 200 universities experiment. Query $\mathtt{q}_2$, designed specifically to stress-test the filter, as expected produces a huge number of spurious answers. Indeed, the

| | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $req_1$ | $req_2$ | $req_3$ | $req_4$ | $req_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| spurious answers | 2 | 28M | 2 | 24K | 0 | 0 | 0 | 22K | 0 | 163K | 0 |
| valid answers | 4.6M | 0 | 0 | 0 | 8.3M | 0 | 0 | 410K | 48K | 137K | 0 |

**Fig. 6.** Number of answers for 200 universities, 5% incompleteness, 20 subclasses.

comparably long execution time of this query appears to be mainly due to the fact that DB2 has to handle a large number of spurious answers before the filtering takes place, and not to a poor performance of the filter itself. The execution times of $q_1$ and $q_5$ can also be explained by a large number of answers. Note that the number of filter calls is actually the sum of the numbers of answers, both spurious and valid. Also note that, in principle, it is possible to avoid an extremely large number of spurious answers in $q_2$ (and any other query) at the cost of slightly increasing the size of the canonical model: duplicate the anonymous part of the canonical model so that no two individuals in the original ABox 'share' an anonymous part of the canonical model. Analyzing this further in experiments is left for future work.

Experiments two and three are reported about in Figures 8 and 9. Here, we only tested the filtering approach. In both cases, we use 100 universities. In experiment two, the number of subclasses is fixed to 20 while in experiment three, the degree of incompleteness is fixed to 5%. In general, the degree of incompleteness has virtually no effect on the execution time of queries. Again, $q_2$ is an exception as the number of spurious answers increases dramatically from 3M for 1% incompleteness to 125M for 20% incompleteness. The number of subclasses also has essentially no effect on query execution times (in contrast to the pure query rewriting approach for which a non-trivial number subclasses can dramatically increase the size of the rewritten query). Note that the execution time of $q_2$ becomes shorter with an increasing number of subclasses because the number of spurious matches decreases from 6.5M for 5 subclasses to 211K for 100 subclasses: this is due to the atoms Subj3Student and Subj4Student in $q_2$ and the fact that the number of assertions for these two concepts decreases as the number of subject areas increases.

## 6 Conclusion

We have modified the combined approach to OBDA by replacing the query rewriting part with a filtering technique. This step is natural from an implementation perspective and allows to circumvent an exponential blowup of the query. Based on experiments with an improved version of the LUBM ontology, we have demonstrated the scalability of our approach.

As future work, we plan to extend the combined approach with filtering to other description logics for which, until now, it is unknown how to avoid an exponential blowup. For example, we believe that polytime filtering is possible for the extension of $\mathcal{EL}$ with transitive roles, as found in the OWL2 EL profile. It would also be interesting to better understand the impact of modifying the canonical model on query answering, both from a theoretical and from a

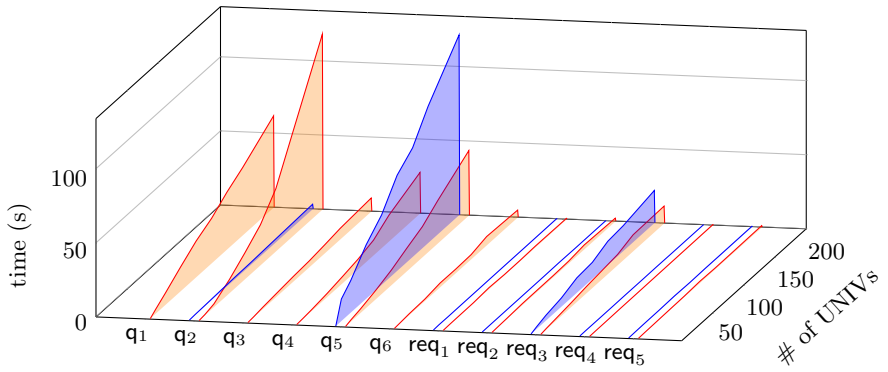


**Fig. 7.** Query run times for varying numbers of Universities.

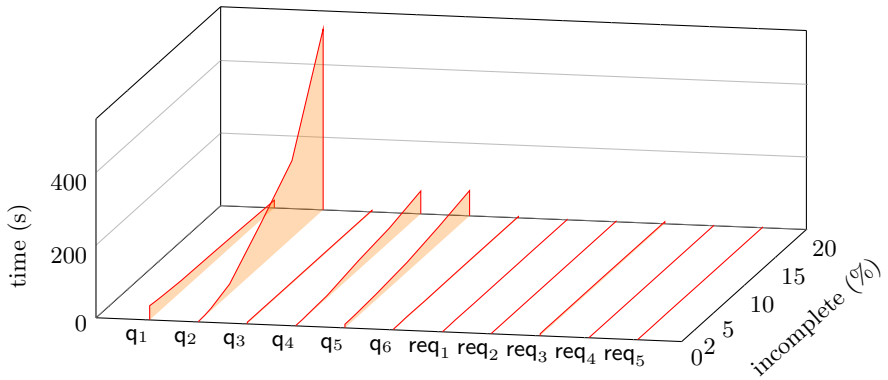


**Fig. 8.** Query run times for varying incompleteness (in %).

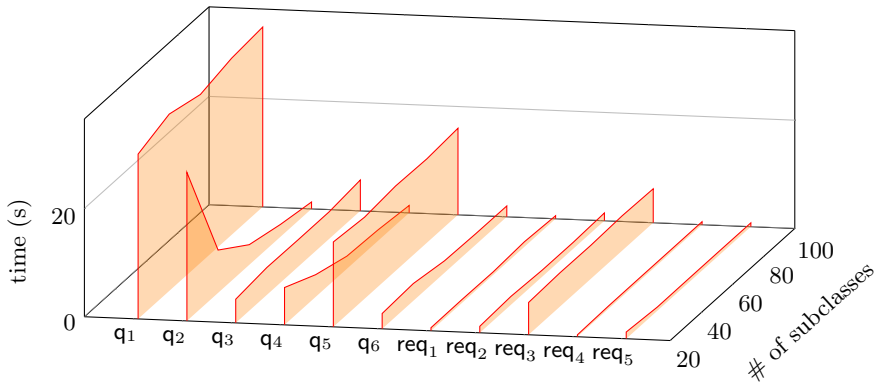


**Fig. 9.** Query run times for varying number of subclasses.

practical perspective. For example, we do not know whether the more natural canonical model obtained by identifying all individuals $c_{R,0}$ and $c_{R,1}$ admits polytime filtering. Moreover, as discussed above it is conceivable that versions of the canonical model that are *less* economic regarding individual reuse result in better runtime in practice.

We also plan to compare the performance of our approach more thoroughly with the performance of pure query rewriting, using other state-of-the-art query rewriting tools such as Quest [12], Presto [13], OWLgres [14], CLIPPER [4]. In this context, it is interesting to note that promising new optimization techniques have recently been developed in [11] and implemented in the Quest system. While some of them (such as the exploitation of ABox integrity constraints) aim specifically at the query rewriting approach, others (such as semantic indexing) can easily be combined with the filtering approach proposed in this paper.

## References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
2. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: AAAI (2011)
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. J. of Automated Reasoning 39(3), 385–429 (2007)
4. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In: AAAI (2012)
5. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Towards practical query answering for Horn-$\mathcal{SHIQ}$. In: Description Logics (2012)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
7. Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: ICALP (2). pp. 263–274 (2012)
8. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR (2010)
9. Lutz, C., Wolter, F., Toman, D.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database systems. In: IJCAI. pp. 2070–2075 (2009)
10. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: International Semantic Web Conference. pp. 489–504 (2009)
11. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR (2012)
12. Rodriguez-Muro, M., Calvanese, D.: Quest, an OWL 2 QL reasoner for ontology-based data access. In: OWLED (2012)
13. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR (2010)
14. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: OWLED (2008)
15. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR (2012)