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. The basic idea of the combined approach to query answering in the presence of ontologies is to materialize the consequences of the ontology in the data and then use a limited form of query rewriting to deal with infinite materializations. While this approach is efficient and scalable for ontologies that are formulated in the basic version of the description logic DL-Lite, it incurs an exponential blowup during query rewriting when DL-Lite is extended with the popular role hierarchies. In this paper, we show how to replace the query rewriting with a filtering technique. This is natural from an implementation perspective and allows us to handle 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 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 very large data sets. The most popular ontology languages that have been considered for OBDA include the three OWL profiles OWL2 RL, OWL2 QL, and OWL2 EL, as well as various description logics and Datalog variants related to these profiles [2,3,5,14,17].

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 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 data $\mathcal{A}_{\mathcal{T}} \supseteq \mathcal{A}$ without any ontology. Thus, one can store $\mathcal{A}_{\mathcal{T}}$ in a 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 [9,11] 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, they can give spurious answers to conjunctive queries (CQs) that involve a cycle. To recover soundness, 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 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 [9,11].

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$_{\mathcal{R}}$ [8]. For the combined approach, an auxiliary query rewriting strategy for DL-Lite$_{\mathcal{R}}$ ontologies and CQs is presented in [9], 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 data is completed. Compared to the data completion for the original combined

approach [9], 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) [7] by introducing additional concepts into the ontology, modifying the data generator so that the produced data is incomplete, and replacing the original, very simple queries by more challenging ones.

(3) We have implemented our approach in a system called COMBO and carry out an experimental evaluation based on the modified LUBM benchmark, both to evaluate the feasibility of our approach and to compare it with the query rewriting approach. Our experiments show that the combined approach is significantly more robust than the rewriting approach when the number of (sub)classes in the ontology or the size of the data increases.

Some technical proofs and details of our experimental evaluation are presented in the appendix of the full version of this paper, available at `http://www.informatik.uni-bremen.de/~clu/combined/`. This paper is an extended version of the workshop paper [10]. In particular, the experimental evaluation carried out in this paper is much more comprehensive than the one in [10].

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.

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 used in $\mathcal{A}$, and write $P^-(a,b) \in \mathcal{A}$ instead of $P(b,a) \in \mathcal{A}$ if 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 α, we write $\mathcal{T} \models \alpha$ when α is a consequence of $\mathcal{T}$ (satisfied in all models of $\mathcal{T}$). Instead of

$\mathcal{T} \models R \sqsubseteq S$, we 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 N_V be a countably infinite set of *variables*. Taken together, the sets N_V and N_I form the set N_T of *terms*. A *conjunctive query (CQ)* takes the form $q = \exists \boldsymbol{y}\,\psi(\boldsymbol{y}, \boldsymbol{x})$, where ψ is a conjunction of *concept atoms* $A(t)$ and *role atoms* $P(t, t')$ where $t, t' \in N_T$. As in the case of ABox assertions, we do not distinguish between $P^-(t, t')$ and $P(t', t)$. The free variables $\boldsymbol{x}$ of φ are 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 *union of conjunctive queries (UCQ)* is a disjunction of CQs. We denote by $\mathsf{term}(q)$ the set of terms in q.

Let $q = \exists \boldsymbol{y}\,\psi(\boldsymbol{y}, \boldsymbol{x})$ be a k-ary CQ with $\boldsymbol{x} = x_1, \ldots, x_k$, and $\mathcal{I}$ an interpretation. A mapping $\pi \colon \mathsf{term}(q) \to \Delta^{\mathcal{I}}$ with $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{term}(q) \cap N_I$ is a *match* for q in $\mathcal{I}$ if $\mathcal{I}$ satisfies ψ 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 $\boldsymbol{a} = a_1, \ldots, a_k$, a match π 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 $a, b \in N_I$. 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 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

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

correctly returns each a with $\mathsf{Student}(a) \in \mathcal{A}$ as a certain answer.
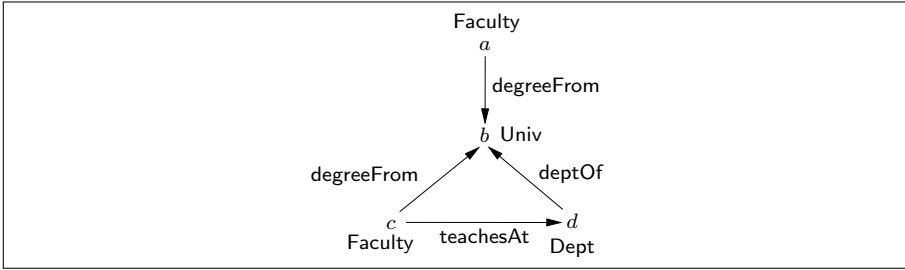
Fig. 1. Completed ABox for Example 3

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

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

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

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

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

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

$$q_2(x) = \exists y, z \; \text{Faculty}(x) \wedge \text{degreeFrom}(x, y) \wedge \text{Univ}(y) \wedge$$
$$\text{deptOf}(z, y) \wedge \text{Dept}(z) \wedge \text{teachesAt}(x, z)$$

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 answering queries; 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 [9,11]. 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 $\mathcal{K}$ [9]. 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* $\mathcal{K}$ if there exist an $a \in \mathsf{Ind}(\mathcal{A})$ and $R_0, \ldots, R_n \in \mathsf{rol}(\mathcal{T})$ 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 [9]: in some cases, we introduce two fresh individuals for a given role instead of only a single one. This helps to avoid choices in the elimination of spurious answers (see Example 8), which are 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

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

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:

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

Note that the slightly more straightforward version of canonical models defined in [9] can be obtained from our canonical models by identifying all elements $c_{R,0}$ and $c_{R,1}$.
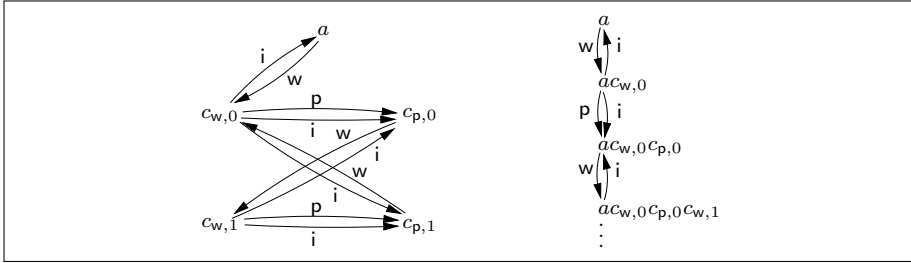
Fig. 2. Canonical model $\mathcal{I}_{\mathcal{K}}$ and unraveled canonical model $\mathcal{U}_{\mathcal{K}}$ for Example 5

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 $\mathcal{T}$ [9].

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 [9]. 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 for the ABox {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 {worksFor, paysSalaryOf}*:*

$$\text{Employee} \sqsubseteq \exists \text{worksFor} \qquad\qquad \exists \text{worksFor}^- \sqsubseteq \text{Employer} \qquad\qquad (6)$$

$$\text{Employer} \sqsubseteq \exists \text{paysSalaryOf} \qquad \exists \text{paysSalaryOf}^- \sqsubseteq \text{Employee} \qquad (7)$$

$$\text{worksFor}^- \sqsubseteq \text{isAffiliatedWith} \qquad \text{paysSalaryOf} \sqsubseteq \text{isAffiliatedWith}. \qquad (8)$$

A part of the canonical model for the ABox {Employee(a)} *and the TBox (6)-(8) with* paysSalaryOf $\prec$ 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.*

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 $R \in \mathsf{N}_R^-$, and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N}_R^-$, $1 \leq i < n$. We denote by $\mathsf{tail}(\sigma)$ the last element of the path σ. The unraveled canonical model $\mathcal{U}_{\mathcal{K}}$ is then defined by taking:

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

$$R^{\mathcal{U_K}} = \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \exists S : S(a, b) \in \mathcal{A} \text{ and } S \sqsubseteq_{\mathcal{T}}^{*} R\} \cup$$
$$\{(\sigma, \sigma d) \mid \sigma d \in \Delta^{\mathcal{U_K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_R d\} \cup$$
$$\{(\sigma d, \sigma) \mid \sigma d \in \Delta^{\mathcal{U_K}} \text{ and } \mathsf{tail}(\sigma) \rightsquigarrow_{R^-} d\}.$$

As an example, the canonical model $\mathcal{U_K}$ for the KB from Example 5 is shown on the right-hand side of Figure 2. The following result shows that, as one would expect, $\mathcal{U_K}$ does not suffer from spurious answers.

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

The proof of Theorem 1 is standard and omitted, see [9] for a similar proof.

4   Filtering

To remove spurious answers, we install a filtering procedure as a user-defined function in the RDBMS. In this approach, calls to the filtering procedure are delegated to the RDBMS in hopes that the query optimizer will eliminate spurious answers as early as possible in the execution plan. The procedure takes as input a match of the query in the canonical model $\mathcal{I_K}$ 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 $\mathcal{U_K}$ and Theorem 1.

Let $\mathcal{K}$ be a KB and $q(\boldsymbol{x})$ a CQ. A match π of q in $\mathcal{I_K}$ *is reproduced by* a match τ of q in $\mathcal{U_K}$ if for all $t \in \mathsf{term}(q)$, we have $\pi(t) = \mathsf{tail}(\tau(t))$. We say that π is *spurious* if it is not reproduced by any match τ of q in $\mathcal{U_K}$. The following lemma, which is an immediate consequence of Theorem 1, shows that $\mathcal{I_K}$ 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 a non-spurious $\boldsymbol{a}$-match π of q in $\mathcal{I_K}$.*

We want to show that it can be decided in time polynomial in the size of q and $\mathcal{T}$ (and without accessing $\mathcal{A}$ at all) whether a given match in $\mathcal{I_K}$ is spurious. Clearly, it is enough to test for each maximally connected component of q whether the match is spurious on that component. We thus 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 $d_1, \dots, d_n \in \Delta^{\mathcal{I_K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $d_i \rightsquigarrow_R d_{i+1}$ for some $R \in \mathsf{N_R^-}, 1 \leq i < n$. We use Paths to denote the set of all paths, both anonymous and non-anonymous. A *root configuration for q given π* is a set $\rho \subseteq \mathsf{term}(q)$ such that one of the following conditions is true:

- ρ is the set of those $t \in \mathsf{term}(q)$ such that $\pi(t) \in \mathsf{N_I}$ and this set is non-empty;
- the above set is empty and ρ contains exactly one term (actually a variable).

The filtering procedure immediately returns "false" if some answer variable is mapped to an element of $\Delta^{\mathcal{I_K}}$ that is not from $\mathsf{Ind}(\mathcal{A})$ (based on the name of
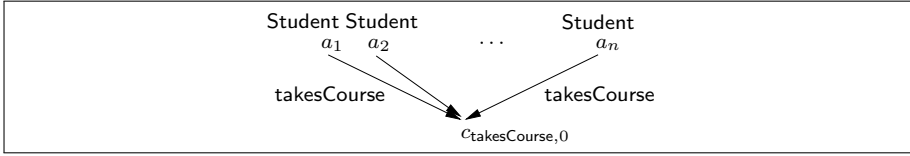
Fig. 3. Canonical model $\mathcal{I}_{\mathcal{K}}$ for Example 6

the element, i.e., whether it is of the form $c_{R,i}$). Then the procedure iterates through all root configurations ρ. For each ρ, it constructs a sequence $S_\rho^0, S_\rho^1, \ldots$ of relations $S_\rho^i \subseteq \mathsf{term}(q) \times \mathsf{Paths}$ as follows:

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

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

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

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 π, 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 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 [9], the filtering procedure follows a simple idea for reproducing the input match π in $\mathcal{I}_\mathcal{K}$ as a match τ 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 σ' 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

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

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

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

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

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

which is functional, showing that the match π is not spurious. In a canonical model $\mathcal{I}_\mathcal{K}$ where $c_{\mathsf{w},0}$ and $c_{\mathsf{w},1}$ 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 $\pi(s) \leadsto_R \pi(t)$ and $\pi(t) \leadsto_{R^-} \pi(s)$, 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 $\pi(s) \leadsto_R \pi(t)$ 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 $\pi(s) \leadsto_R \pi(t)$ holds by the construction of $\mathcal{I}_\mathcal{K}$ since π 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 $\pi(s) \leadsto_R \pi(t)$ can be checked by using only π and $\mathcal{T}$ based on the definition of "$\leadsto_R$".

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 ρ and then need to compute S_ρ, 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 the appendix of the full version.

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

5   Implementation and Experiments

We have implemented our approach in the Combo system, a collection of tools that support the user in setting up the tables of a relational database system to store ABoxes and their completion, implements the actual data completion via querying, and allows to compile an ontology into a filter that takes the form of a user defined function. The preferred relational database system for use with Combo is IBM DB2.

We use this combination to carry out an experimental evaluation of our approach, and to compare it to the query rewriting approach. The experiments are based on a modified version of the ontology from the Lehigh University Benchmark (LUBM) [7] and on ABoxes produced by a modified version of the LUBM data generator. We use six queries that were hand-crafted specifically for our experiments. The mentioned modifications aim at making the LUBM suite more realistic for OBDA evaluation. We believe that this setup might be interesting also for future experiments and provide it online at http://www.informatik.uni-bremen.de/~clu/combined/.

Regarding the query rewriting approach, we use Rapid (v0.3) [4] and Presto (version March 25th 2013) [15] as typical examples of state-of-the-art rewriting tools. Both Rapid and Presto are able to generate rewritings into UCQs and into non-recursive Datalog (DLog), and they use various optimizations to generate as small rewritings as possible.

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 carefully chosen 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 the appendix 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 many real-world 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 the appendix 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.

We use the six queries in Figure 4 that we have hand-crafted specifically for our experiments. Note that cq_3 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. Requiem test queries are commonly used for benchmarking

query rewriting systems [12]. We did not include those queries since they are too simple for our purposes. In fact, they are answered effortlessly both by our approach and by the query rewriting approach.

5.2   Results

We report on two experiments: in the first experiment we vary the complexity of the ontology by increasing the number of subclasses (the parameter n of the ontology $LUBM_n^\exists$) and in the second experiment we vary the data size by increasing the number of universities that are generated by the modified LUBM data generator. It turned out that, in general, the degree of incompleteness had only very limited effect on the execution time of queries. We therefore do not vary the degree of incompleteness but use 5% incompleteness in the data for both experiments. 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 Express-C version 9.7.5.

```
cq1(x,z)<-Student(x), takesCourse(x,y), Subj1Course(y), teacherOf(z,y),
         Professor(z), headOf(z,w), Department(w), memberOf(x,w)
cq2(x)  <-Faculty(x), degreeFrom(x,y), University(y),
         subOrganizationOf(z,y), Department(z), memberOf(x,z)
cq3(x,y)<-Professor(z), memberOf(z,x), Subj3Department(x),
         publicationAuthor(w,z), Professor(v), memberOf(v,y),
         Subj4Department(y), publicationAuthor(w,v)
cq4(x,y)<-Department(x), memberOf(z,x), Student(z), takesCourse(z,v),
         teacherOf(w,v), Professor(w), memberOf(w,y), Department(y)
cq5(x)  <-Person(x), worksFor(x,y), Department(y), takesCourse(x,z),
         Course(z)
cq6(x)  <-Student(x), publicationAuthor(y,x), Publication(y),
         teachingAssistantOf(x,z), Course(z)
```

Fig. 4. Queries cq_1 to cq_6

#Univ.	individuals	Original ABox			Data Completion		
		concepts	roles	time	concepts	roles	time
200	4M	7M	12M	7m30s	12M	22M	16m55s
500	10M	17M	31M	39m06s	31M	55M	70m48s
1000	21M	35M	63M	43m17s	63M	111M	146m39s

Fig. 5. Size original and completed ABox (in million) and load and completion time

The size of the test data for the experiments is detailed in Figure 5, where we give (for 20 subclasses) the number of individuals in the original ABox (there are only about 200 additional individuals in the completion), the number of concept and role assertions (in the original ABox and in its completion), and the load time for the original and the completed ABox (including the completion time).

Test	System	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
1000.10	Rap-DLog	23.54	TO	40.88	TO	TO	50.41
	Pre-DLog	23.61	TO	43.66	75.33	TO	15.69
	Combo	24.42	393.97	TO	267.55	23.67	38.93
1000.20	Rap-DLog	TO	TO	33.80	TO	TO	54.12
	Pre-DLog	22.86	TO	32.48	TO	TO	17.24
	Combo	18.13	460.16	587.76	266.97	28.30	38.72
1000.40	Rap-DLog	TO	TO	76.69	TO	TC	65.45
	Pre-DLog	23.21	TO	75.71	TO	TO	18.27
	Combo	13.56	456.84	279.09	270.12	28.31	37.43
1000.80	Rap-DLog	TO	TO	UM	TC	TC	TC
	Pre-DLog	TO	TO	UM	TC	TC	17.81
	Combo	7.10	448.69	152.55	268.86	28.07	39.26

Fig. 6. Run time for varying number of subclasses

Test	System	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
200.20	Rap-DLog	6.72	4.68	3.35	13.37	14.36	9.22
	Pre-DLog	5.84	55.55	5.79	146,26	11.76	2.78
	Combo	4.75	22.65	25.58	51.17	6.50	4.07
500.20	Rap-DLog	14.32	343.47	15.04	TO	TO	25.58
	Pre-DLog	11.32	344.36	14.96	TO	TO	8.18
	Combo	14.14	116.34	161.16	135.36	11.59	19.46
1000.20	Rap-DLog	TO	TO	33.80	TO	TO	54.12
	Pre-DLog	22.86	TO	32.48	TO	TO	17.24
	Combo	18.13	460.16	587.76	266.97	28.30	38.72

Fig. 7. Run time for varying number of universities

	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
Rap-UCQ	57984	15120	14880	162288	1950	1702
Rap-DLog	85	68	39	81	118	105
Pre-UCQ	TO	15120	14880	TO	1950	1702
Pre-DLog	85	68	39	81	86	63

Fig. 8. Number of disjuncts in UCQ and rules in Datalog program

For our experiments, summarized in Figures 6 and 7, we report the execution time (in seconds; TO stands for 600s timeout, TC for the DB2 output "The statement is too long or too complex", and UM for the DB2 output "Unexpect maxNumBrunch") for the Datalog rewritings generated by Rapid and Presto (transformed into SQL by unfolding them into positive existential queries) and for the Combo filtering approach. We do not report execution times for any UCQ rewritings because they are excessively large and DB2 fails to execute them in all of our experiments, see Figure 8.[1]

---

[1] We are not aware of any experimental evaluation of the query execution time of rewritings into non-recursive Datalog. Our experiments show that Datalog rewritings can be significantly more efficient than UCQ rewritings.

	cq_1	cq_2	cq_3	cq_4	cq_5	cq_6
spurious answers	0	2	600K	35K	0	0
valid answers	32K	2K	0	20K	0	465K

Fig. 9. Number of answers for 1000 universities, 20 subclasses

The main outcomes of our experiments are as follows:

1. the COMBO filtering approach is significantly more robust than the rewriting approaches when both the complexity of the ontology and the size of the data increase. We observe only one timeout for the filtering approach (cq_3 for 1000 universities and 10 subclasses) but the rewriting approach eventually fails for all queries with the exception of cq_6.
2. For smaller data sets or for simple class hierarchies (e.g., 200 universities and 20 subclasses), the filtering and rewriting approaches are comparable.
3. In contrast to the rewriting approach, the performance of the filtering approach does not depend significantly on the number of subclasses.[2]

The poor performance of the rewriting approach for complex ontologies and large data is due to the fact that for complex queries (the SQL queries corresponding to the datalog rewriting) the DB2 query optimizer realizes that the use of data structures, such as B-tree indices, becomes imperative and attempts to distribute (index) joins into unions to take advantage of these indices. Such an attempt, however, commonly leads to exhausting the resources available for query optimization (DB2 then aborts by outputting TC or UM).

The poor performance of the filtering approach for query cq_3 is due to the large number of spurious answers, see Table 9 for an overview of the number of valid and spurious answers for all queries. It is possible to avoid this behavior at the cost of slight increase of the size of the canonical model: by duplicating the anonymous parts of the canonical model so that no two (or few) individuals in the original ABox 'share' an anonymous part of the canonical model.

We close by commenting on the data loading and completion times reported in Figure 5. Here the completion time is spent almost exclusively on loading the data into the DBMS: indeed, loading large amounts of data into a relational DBMS can be time consuming since the system needs to build up indexes and other auxiliary data structures. Note, however, that bulk-loading data is rare in most applications: standard workloads typically add and remove few tuples at a time. In our case adding (removing), e.g., 100 concept/role assertions into (from) the original ABox results in adding and removing less than 500 tuples from the completed data (for the LUBM ontology), yielding an essentially instant update. Moreover, the changes to the completed data can be efficiently computed given the original ABox and the update request using incremental view maintenance technology [6].

---

[2] The increase of performance for cq_1 and cq_3 when the number of subclasses grows is due to each subclass becoming less populated as the data size is fixed.

6   Conclusion

We have modified the combined approach to OBDA by replacing the query rewriting part with a filtering technique. This is natural from an implementation perspective and allows to avoid an exponential blowup of the query. We have implemented our approach in the COMBO system and generated an improved version of the LUBM benchmark that aims at evaluating OBDA approaches. Our experiments demonstrate the scalability and robustness of our approach.

In the future 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 of the query. 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. Note that, based on the workshop predecessor [10] of this paper, the combined approach with filtering has already been picked up to implement OBDA for an extension of $\mathcal{EL}$ (but without transitive roles) [16].

From an applied perspective, it would be interesting to compare our approach also with the promising new optimization techniques that have recently been developed in [13,14]. 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 used also for the combined approach. We did not include those optimizations and systems in our evaluation because all available implementations seem to require prerequisites that are not satisfied in our tests (such as the availability of mappings from a relational database to the ontology or the storage of the ABox in an in-memory database).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
2. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: AAAI, pp. 1541–1546 (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. Journal of Automated Reasoning 39(3), 385–429 (2007)
4. Chortaras, A., Trivela, D., Stamou, G.B.: Optimized query rewriting for OWL 2 QL. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 192–206. Springer, Heidelberg (2011)
5. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., Xiao, G.: Towards practical query answering for Horn-$\mathcal{SHIQ}$. In: DL, pp. 158–168 (2012)
6. Griffin, T., Libkin, L.: Incremental maintenance of views with duplicates. In: ICMD, pp. 328–339 (1995)
7. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 158–182 (2005)
8. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 263–274. Springer, Heidelberg (2012)

 9. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR, pp. 247–257 (2010)
10. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: SSWS+HPCSW, pp. 16–31 (2012)
11. 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)
12. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)
13. Pinto, F.D., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: EDBT, pp. 561–572 (2013)
14. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR, pp. 308–318 (2012)
15. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR, pp. 290–300 (2010)
16. Stefanoni, G., Motik, B., Horrocks, I.: Introducing nominals to the combined query answering approaches for $\mathcal{EL}$. In: DL, pp. 962–974 (2013)
17. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR, pp. 96–106 (2012)