Efficient Upper Bound Computation of Query Answers in Expressive Description Logics

Yujiao Zhou, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

1 Introduction

In recent years, there has been a growing interest in the problem of conjunctive query answering over description logic (DL) ontologies and large scale data sets. This problem is central to many applications, which often involve managing data sets consisting of hundreds of millions, or even billions of data assertions.

Meeting the scalability requirements of such applications is, however, a very challenging problem. Answering conjunctive queries over ontologies in expressive DLs is of high computational complexity; in fact, decidability is still an open problem for $\mathcal{SROIQ}$ [14] —the DL that underpins the standard ontology language OWL 2 [6]. Although small restriction on the ontology or query language can ensure decidability [26], worst case complexity is still very high (at least 2NExpTime) [15]. Several OWL/$\mathcal{SROIQ}$ reasoners, including HermiT [20], FaCT++ [28] and Racer [12], support query answering for restricted classes of conjunctive queries, but in spite of intensive efforts at optimisation, they can still only deal with small to medium size data sets [17, 13].

Scalability of query answering can be ensured by restricting the expressive power of the ontology language to the level that makes reasoning tractable. This has led to the development of three profiles of OWL 2, namely OWL 2 RL, OWL 2 EL, and OWL 2 QL [21]; these profiles are based on (families of) "lightweight" description logics, notably the DLP [10], DL-Lite [4], and the $\mathcal{EL}$ families of DLs [2], respectively. Query answering in all these lightweight DLs can be implemented in polynomial time w.r.t. the size of data (and even in LogSpace in the case of DL-Lite). Such appealing theoretical properties have spurred the development of specialised reasoning techniques [24, 22, 18, 4, 16].

Although allowing for efficient query answering, these lightweight DLs impose severe restrictions on the expressive power of the ontology language. In order to provide scalable query answering w.r.t. ontologies that cannot be captured by such lightweight DLs, Semantic Web researchers have developed reasoners that can process arbitrary OWL/$\mathcal{SROIQ}$ ontologies, but that guarantee completeness only for ontologies that fall within the fragment defined by the OWL 2 RL profile. Given the close connection between OWL 2 RL and datalog, these reasoners typically implement (deductive) database algorithms based on data materialisation. Examples of such systems include Jena [19] and OWLim [16].

All such materialisation-based reasoners are *sound*; that is, the answers they compute can be seen as a *lower bound* on the complete set of query answers.

For ontologies outside the OWL 2 RL profile, however, these reasoners are *incomplete*, and hence they are not guaranteed to compute all query answers. A possible approach to this problem is to investigate the behaviour of the reasoner on a given query Q and ontology $\mathcal{T}$ in an effort to show that it behaves as a complete reasoner w.r.t. Q, $\mathcal{T}$ and an arbitrary data set [7]. Providing such guarantees is, however, not always possible.

An alternative approach, which we investigate in this paper, is to devise a scalable procedure for computing complete but possibly unsound query answers. Such answers provide an *upper bound* to the set of all answers, which can complement the lower bounds efficiently computed by incomplete reasoners. The combined use of such lower and upper bounds has many interesting implications. First, the difference between the upper and lower bounds can be used as an optimisation for reducing the number of candidate answers; furthermore, it also provides a measure of the degree of incompleteness of a reasoner for a given input; finally, if both bounds match, we can efficiently compute all query answers without relying on potentially very expensive entailment tests.

In order to be useful, upper bounds should clearly be as tight as possible, and should also be efficiently computable. Obtaining such an upper bound is, however, not straightforward. The technique we use is to approximate $\mathcal{T}$ to give an ontology $\mathcal{T}'$ that is strictly "stronger" that $\mathcal{T}$ (i.e., $\mathcal{T}' \models \mathcal{T}$), and that is within a fragment for which query answers can be efficiently computed.

Knowledge approximation has been extensively studied in the literature, although mostly in the direction of weakening the ontology/theory [8, 23]. There has also been some work on strengthening approximations. For propositional logic, Horn theories can be used to both strengthen and weaken the original theory [27]; these Horn approximations can then be used to optimise reasoning by exploting the more efficient inference in the Horn theories. Finally, approximation is also related to the computation of least common subsumers [1].

Our technique exploits recent work on chase termination for existential rules, which introduces a so called Models-Summarising Acyclicity (MSA) check [5]. MSA is an approximation of existential rules (datalog$^{\pm}$) into datalog. As most $\mathcal{SROIQ}$ TBoxes can be translated into existential rules extended with disjunction (datalog$^{\pm,\vee}$) using model-preserving transformations, we can adapt MSA to produce a datalog approximation of a $\mathcal{SROIQ}$ TBox. Moreover, the resulting datalog rules can be translated back into an OWL 2 RL TBox for which complete query answers can be computed using materialisation based reasoners.

We have implemented our approach, and conducted preliminary experiments using LUBM [11]. Our preliminary results suggest that our bound could be tight for many queries, and it can be computed efficiently (or at least with similar efficiency to computing the lower bound).

2 Preliminaries

We assume basic familiarity with the DLs underpinning the ontology language OWL 2 and its profiles. We next introduce datalog$^{\pm,\vee}$ and datalog languages,

and define the syntax and semantics of conjunctive queries. In our definitions, we adopt standard first-order logic notions of variable, constant, term, substitution, atom, formula, and sentence. We assume all formulas to be function-free. We denote with $\bot$ the special atom evaluated as false in all interpretations, and we use $\approx$ to denote the special equality predicate in first-order logic. Finally, we also adopt the standard notions of satisfiability, unsatisfiability, and entailment.

Datalog Languages. A datalog$^{\pm,\vee}$ *rule* r is a formula of the form (1), where each B_j is an atom different from $\bot$ whose free variables are contained in $\boldsymbol{x}$, and

- $m = 1$ and $\varphi_1(\boldsymbol{x}, \boldsymbol{y}_1) = \bot$, or
- $m \geq 1$ and, for each $1 \leq i \leq m$, formula $\varphi_i(\boldsymbol{x}, \boldsymbol{y}_i)$ is a conjunction atoms different from $\bot$ whose free variables are contained in $\boldsymbol{x} \cup \boldsymbol{y}_i$.

$$\forall \boldsymbol{x}.[B_1 \wedge ... \wedge B_n] \rightarrow \bigvee_{i=1}^{m} \exists \boldsymbol{y}_i.\varphi_i(\boldsymbol{x}, \boldsymbol{y}_i) \tag{1}$$

A rule is safe if each variable in $\boldsymbol{x}$ also occurs in some B_j, and we consider only safe rules. For brevity, the quantifier $\forall \boldsymbol{x}$ is left implicit. The *body* of r is the set $\mathsf{body}(r) = \{B_1, \ldots, B_n\}$, and the *head* of r is the formula $\mathsf{head}(r) = \bigvee_{i=1}^{m} \exists \boldsymbol{y}_i.\varphi(\boldsymbol{x}, \boldsymbol{y}_i)$. A datalog$^{\pm,\vee}$ rule r is datalog$^{\pm}$ if $m = 1$ [3], and it is datalog if it is datalog$^{\pm}$ and the head is a single atom without existential quantifiers.

A *fact* is a ground atom, and an *instance* I is a finite set of facts. We denote with $\mathsf{ind}(I)$ the set of all individuals occurring in I.

For Σ a set of datalog rules and I an instance, the *saturation* of Σ w.r.t. I is the instance I' consisting of all facts entailed by $\Sigma \cup I$.

Most DL ontologies can be transformed into a set of datalog$^{\pm,\vee}$ rules and an instance by means of standard transformations. The rules and facts obtained via such transformations involve only unary and binary predicates; thus, we define an ABox $\mathcal{A}$ as an instance containing only unary and binary atoms.

Queries. A *conjunctive query* (CQ), or simply a *query*, is a formula $Q(\boldsymbol{x})$ of the form $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms. A tuple of individuals $\boldsymbol{a}$ is a *certain answer* to a query $Q(\boldsymbol{x})$ w.r.t. a set of first-order sentences $\mathcal{F}$ and an instance I if $\mathcal{F} \cup I \models Q(\boldsymbol{a})$. The set of answers of $Q(\boldsymbol{x})$ w.r.t. $\mathcal{F}$ and I is denoted as $\mathsf{cert}(Q, \mathcal{F}, I)$, where the free variables of $Q(\boldsymbol{x})$ are omitted for brevity.

3 Technical Approach

3.1 Overview

Given a TBox $\mathcal{T}$, an ABox $\mathcal{A}$ and a query Q, our goal is to compute a (hopefully tight) upper bound to the set $\mathsf{cert}(Q, \mathcal{T}, \mathcal{A})$ of answers. We poceed as follows:

1. Transform $\mathcal{T}$ into a set $\Sigma_{\mathcal{T}}$ of datalog$^{\pm,\vee}$ rules such that $\Sigma_{\mathcal{T}}$ is a conservative extension of $\mathcal{T}$.
2. Transform $\Sigma_{\mathcal{T}}$ into a set $\mathsf{approx}(\Sigma_{\mathcal{T}})$ of datalog rules s.t. $\mathsf{approx}(\Sigma_{\mathcal{T}}) \models \Sigma_{\mathcal{T}}$.

$$
\begin{array}{rcl}
\mathsf{Student} \sqsubseteq \mathsf{Person} & \rightsquigarrow & \mathsf{Student}(x) \rightarrow \mathsf{Person}(x) \\
\mathsf{RA} \sqsubseteq \mathsf{Student} & \rightsquigarrow & \mathsf{RA}(x) \rightarrow \mathsf{Student}(x) \\
\mathsf{RA} \sqsubseteq \exists \mathsf{works}.\mathsf{Group} & \rightsquigarrow & \mathsf{RA}(x) \rightarrow \exists y.[\mathsf{works}(x,y) \wedge \mathsf{Group}(y)] \\
\mathsf{Group} \sqsubseteq \mathsf{Org} & \rightsquigarrow & \mathsf{Group}(x) \rightarrow \mathsf{Org}(x) \\
 & & \mathsf{Emp}(x) \rightarrow \mathsf{Person}(x) \\
\mathsf{Emp} \equiv \mathsf{Person} \sqcap \exists \mathsf{works}.\mathsf{Org} & \rightsquigarrow & \mathsf{Emp}(x) \rightarrow \exists y.[\mathsf{works}(x,y) \wedge \mathsf{Org}(y)] \\
 & & \mathsf{Person}(x) \wedge \mathsf{works}(x,y) \wedge \mathsf{Org}(y) \rightarrow \mathsf{Emp}(x) \\
\mathsf{works} \sqsubseteq \mathsf{memberOf} & \rightsquigarrow & \mathsf{works}(x,y) \rightarrow \mathsf{memberOf}(x,y) \\
\mathsf{Student} \sqsubseteq \mathsf{Grad} \sqcup \mathsf{Undergrad} & \rightsquigarrow & \mathsf{Student}(x) \rightarrow \mathsf{Grad}(x) \vee \mathsf{Undergrad}(x) \\
\mathsf{func}(\mathsf{works}) & \rightsquigarrow & \mathsf{works}(x,y) \wedge \mathsf{works}(x,z) \rightarrow y \approx z
\end{array}
$$

Fig. 1. Transforming $\mathcal{T}_{\mathsf{ex}}$ into datalog$^{\pm,\vee}$ rules $\Sigma_{\mathcal{T}_{\mathsf{ex}}}$

3. Transform $\mathsf{approx}(\Sigma_{\mathcal{T}})$ into an OWL 2 RL TBox $\mathcal{T}'$ for which we have $\mathsf{cert}(Q, \mathsf{approx}(\Sigma_{\mathcal{T}}), \mathcal{A}) = \mathsf{cert}(Q, \mathcal{T}', \mathcal{A})$ for every query Q and ABox $\mathcal{A}$.

Our transformation depends only on $\mathcal{T}$, and satisfies the following property:

$$\mathsf{cert}(Q, \mathcal{T}, \mathcal{A}) \subseteq \mathsf{cert}(Q, \mathcal{T}', \mathcal{A}) \quad \text{for every query } Q \text{ and ABox } \mathcal{A}$$

Given the OWL 2 RL TBox $\mathcal{T}'$ we can then use $\mathcal{T}$ and $\mathcal{T}'$ with a materialisation based reasoner rl that is sound for OWL 2 and complete for OWL 2 RL (such as OWLim) to respectively compute a lower and an upper bound to query answers for the given Q and $\mathcal{A}$. More precisely, we have:

$$\mathsf{rl}(Q, \mathcal{T}, \mathcal{A}) \subseteq \mathsf{cert}(Q, \mathcal{T}, \mathcal{A}) \subseteq \mathsf{rl}(Q, \mathcal{T}', \mathcal{A}) \quad \text{for every } Q \text{ and } \mathcal{A}$$

3.2 Computing the Upper Bound

We next describe in detail the transformations in Steps 1-3. As a running example, we use the TBox $\mathcal{T}_{\mathsf{ex}}$ in Figure 1.

From DL TBoxes to Datalog$^{\pm,\vee}$ Rules. The first step is to transform the DL TBox $\mathcal{T}$ into a set $\Sigma_{\mathcal{T}}$ of datalog$^{\pm,\vee}$ rules such that $\Sigma_{\mathcal{T}}$ is a conservative extension of $\mathcal{T}$. For a wide range of DLs, this can be achieved by first using the well-known correspondence between DL axioms and first-order logic formulas and then applying standard structural transformation rules, which may involve introducing new predicates. The transformation of our example $\mathcal{T}_{\mathsf{ex}}$ into datalog$^{\pm,\vee}$ rules $\Sigma_{\mathcal{T}_{\mathsf{ex}}}$ is also shown in Figure 1; here, $\mathcal{T}_{\mathsf{ex}}$ and $\Sigma_{\mathcal{T}_{\mathsf{ex}}}$ are equivalent.

From Datalog$^{\pm,\vee}$ to Datalog. Next, we approximate the datalog$^{\pm,\vee}$ rules $\Sigma_{\mathcal{T}}$ by a datalog program $\mathsf{approx}(\Sigma_{\mathcal{T}})$ that entails $\Sigma_{\mathcal{T}}$. Intuitively, this transformation can be described in two steps:

– Rewrite each datalog$^{\pm,\vee}$ rule r into a set of datalog$^{\pm}$ rules by transforming disjunctions in the head of r into conjunctions and splitting the conjunctions into different datalog$^{\pm}$ rules. Such a way to eliminate disjunction in the head has been used in OWL reasoning with logic program [25].

$$\begin{array}{rcl}
\mathsf{RA}(x) \to \exists y.[\mathsf{works}(x,y) \wedge \mathsf{Group}(y)] & \rightsquigarrow & \mathsf{RA}(x) \to \mathsf{works}(x,c_1) \wedge \mathsf{Group}(c_1) \\
\mathsf{Emp}(x) \to \exists y.[\mathsf{works}(x,y) \wedge \mathsf{Org}(y)] & \rightsquigarrow & \mathsf{Emp}(x) \to \mathsf{works}(x,c_2) \wedge \mathsf{Org}(c_2) \\
\mathsf{Student}(x) \to \mathsf{Grad}(x) \vee \mathsf{Undergrad}(x) & \rightsquigarrow & \mathsf{Student}(x) \to \mathsf{Grad}(x) \wedge \mathsf{Undergrad}(x)
\end{array}$$

Fig. 2. Transforming $\Sigma_{\mathcal{T}_{\mathsf{ex}}}$ into $\mathsf{approx}(\Sigma_{\mathcal{T}_{\mathsf{ex}}})$

– Transform the resulting datalog$^{\pm}$ rules into datalog using fresh individuals to skolemise existentially quantified variables.

Figure 2 illustrates this process for our example. The figure shows only the rules in $\Sigma_{\mathcal{T}_{\mathsf{ex}}}$ that need changing; note that c_1 and c_2 are fresh individuals used to skolemise existentially quantified variables.[1]

Definition 1. *For each datalog$^{\pm,\vee}$ rule r of the form (1) and each variable $y_{ij} \in \boldsymbol{y}_i$, let c^r_{ij} be a fresh individual unique for r and y_{ij}. Then, $\mathsf{approx}(r)$ is the following set of rules, where for each $1 \leq i \leq m$, θ^r_i is a substitution mapping each variable $y_{ij} \in \boldsymbol{y}_i$ to c^r_{ij}:*

$$\mathsf{approx}(r) = \{B_1 \wedge ... \wedge B_n \to \varphi_i(\boldsymbol{x}, \theta^r_i(\boldsymbol{y}_i)) \mid 1 \leq i \leq m\} \tag{2}$$

For Σ a set of datalog$^{\pm,\vee}$ rules, $\mathsf{approx}(\Sigma)$ is the smallest set that contains $\mathsf{approx}(r)$ for each rule $r \in \Sigma$.

We next show that $\mathsf{approx}(\Sigma_{\mathcal{T}})$ entails $\Sigma_{\mathcal{T}}$. The following proposition provides sufficient and necessary conditions for a datalog$^{\pm,\vee}$ rule to be entailed by an arbitrary set of first-order sentences (the proof is rather straightforward and can be found in [7]).

Proposition 1. *Let $\mathcal{F}$ be a set of first-order sentences and r be a datalog$^{\pm,\vee}$ rule of the form (1). Then, for each substitution σ mapping the free variables of r to distinct individuals not occurring in $\mathcal{F}$ or r, we have $\mathcal{F} \models r$ if and only if*

$$\mathcal{F} \cup \{\sigma(B_1), \ldots, \sigma(B_n)\} \models \bigvee_{i=1}^{m} \exists \boldsymbol{y}_i.\varphi_i(\sigma(\boldsymbol{x}), \boldsymbol{y}_i)$$

Proposition 1 can be used to show that each datalog$^{\pm,\vee}$ rule r in $\Sigma_{\mathcal{T}}$ is entailed by $\mathsf{approx}(r)$, and hence $\mathsf{approx}(\Sigma_{\mathcal{T}})$ strengthens $\Sigma_{\mathcal{T}}$.

Proposition 2. *For Σ a set of datalog$^{\pm,\vee}$ rules, we have $\mathsf{approx}(\Sigma) \models \Sigma$.*

Proof. It suffices to show that, for each rule $r \in \Sigma$ of the form (1) and each $r_i \in \mathsf{approx}(r)$, we have $r_i \models r$. Let σ be a substitution mapping the free variables in r to fresh individuals; by Proposition 1, we have

$$r_i \models r \Leftrightarrow r_i \cup \{\sigma(B_1), \ldots, \sigma(B_n)\} \models \bigvee_{i=1}^{m} \exists \boldsymbol{y}_i.\varphi_i(\sigma(\boldsymbol{x}), \boldsymbol{y}_i) \tag{3}$$

[1] Note that, although $\mathsf{approx}(\Sigma_{\mathcal{T}_{\mathsf{ex}}})$ is strictly speaking not datalog (we have conjunction of atoms in the head), it is trivially equivalent to a set of datalog rules.

Since r_i and r have the same body atoms, the definition of r_i in (2) implies

$$r_i \cup \{\sigma(B_1), \ldots, \sigma(B_n)\} \models \varphi_i(\sigma(\boldsymbol{x}), \theta_i^r(\boldsymbol{y}_i)) \tag{4}$$

Since substitution θ_i^r maps variables to constants, the following conditions clearly hold by the semantics of first-order logic:

$$\varphi_i(\sigma(\boldsymbol{x}), \theta_i^r(\boldsymbol{y}_i)) \models \exists \boldsymbol{y}_i.\varphi_i(\sigma(\boldsymbol{x}), \boldsymbol{y}_i) \models \bigvee_{i=1}^{m} \exists \boldsymbol{y}_i.\varphi_i(\sigma(\boldsymbol{x}), \boldsymbol{y}_i) \tag{5}$$

But then, conditions (3), (4) and (5) immediately imply $r_i \models r$, as required. □

The fact that $\mathsf{approx}(\Sigma)$ entails Σ implies that query answers w.r.t. $\mathsf{approx}(\Sigma)$ are an upper bound to those w.r.t. Σ.

Proposition 3. *For each set of datalog$^{\pm,\vee}$ rules Σ, each ABox $\mathcal{A}$ and each query Q, we have* $\mathsf{cert}(Q, \Sigma, \mathcal{A}) \subseteq \mathsf{cert}(Q, \mathsf{approx}(\Sigma), \mathcal{A})$.

From Datalog to OWL 2 RL. The last step is to transform $\mathsf{approx}(\Sigma_\mathcal{T})$ into an OWL 2 RL TBox $\mathcal{T}'$. Although arbitrary datalog rules cannot be transformed into OWL 2 RL, the rules in $\mathsf{approx}(\Sigma_\mathcal{T})$ have been obtained from a DL TBox and are therefore tree-shaped, which makes this transformation possible.

There is, however, a technical consideration related to the fresh skolem constants in $\mathsf{approx}(\Sigma_\mathcal{T})$. In particular, the following rule in our running example (see Figure 2) does not directly correspond to an OWL 2 RL axiom:

$$\mathsf{RA}(x) \rightarrow \mathsf{works}(x, c_1) \wedge \mathsf{Group}(c_1) \tag{6}$$

This issue can be easily addressed by introducing fresh atomic roles. More precisely, rule (6) can be transformed into the following three OWL 2 RL axioms, where R' is a fresh atomic role:

$$\mathsf{RA} \sqsubseteq \exists R'.\{c_1\} \qquad \top \sqsubseteq \forall R'.\mathsf{Group} \qquad R' \sqsubseteq \mathsf{works}$$

3.3 Additional Considerations

Transformation of Disjunctive Rules. The proof of Proposition 2 suggests that we can easily weaken $\mathsf{approx}(\Sigma_\mathcal{T})$ given in Definition 1 such that $\Sigma_\mathcal{T}$ is still entailed. In particular, when transforming a disjunctive rule in $\Sigma_\mathcal{T}$ into datalog by replacing disjunctions with conjunctions, it suffices to keep only one of the conjuncts. For example, given the transformation

$$\mathsf{Student}(x) \rightarrow \mathsf{Grad}(x) \vee \mathsf{Undergrad}(x) \rightsquigarrow \mathsf{Student}(x) \rightarrow \mathsf{Grad}(x) \wedge \mathsf{Undergrad}(x)$$

we can replace in $\mathsf{approx}(\Sigma_\mathcal{T})$ the rule $\mathsf{Student}(x) \rightarrow \mathsf{Grad}(x) \wedge \mathsf{Undergrad}(x)$ with either $\mathsf{Student}(x) \rightarrow \mathsf{Grad}(x)$, or $\mathsf{Student}(x) \rightarrow \mathsf{Undergrad}(x)$. Any of these choices will lead to a (different) upper bound. In practice, one can choose to process any number of such different options, or simply devise suitable heuristics to choose the most promising one.

Unsatisfiability Issues. For given TBox $\mathcal{T}$ and ABox $\mathcal{A}$, it might be the case that $\mathsf{approx}(\Sigma_\mathcal{T}) \cup \mathcal{A}$ is unsatisfiable, whereas $\Sigma_\mathcal{T} \cup \mathcal{A}$ is satisfiable. Then, the upper bound we obtain is the trivial one for each query. For example, if we extend $\mathcal{T}_\mathsf{ex}$ in Figure 1 with the axiom $\mathsf{Grad} \sqcap \mathsf{Undergrad} \sqsubseteq \bot$, we obtain a rule with $\bot$ in the head in $\Sigma_\mathcal{T}$, namely $\mathsf{Grad}(x) \wedge \mathsf{Undergrad}(x) \rightarrow \bot$. For $\mathcal{A}_\mathsf{ex} = \{\mathsf{RA}(a)\}$ we then have that $\mathcal{T}_\mathsf{ex} \cup \mathcal{A}_\mathsf{ex}$ is satisfiable, but $\mathsf{approx}(\Sigma_{\mathcal{T}_\mathsf{ex}}) \cup \mathcal{A}_\mathsf{ex}$ is unsatisfiable; thus, for a given Q, any tuple of individuals of the appropriate arity must be included in the upper bound.

A way to address this issue is to remove from $\mathsf{approx}(\Sigma_\mathcal{T})$ all rules with $\bot$ in the head. Although it is then no longer the case that $\mathsf{approx}(\Sigma_\mathcal{T}) \models \Sigma_\mathcal{T}$, we can still guarantee completeness provided that $\Sigma_\mathcal{T} \cup \mathcal{A}$ is satisfiable.

Proposition 4. *Let Σ be a set of datalog$^{\pm,\vee}$ rules and let Σ' the result of removing from $\mathsf{approx}(\Sigma)$ all rules containing $\bot$ in the head. Then, the following condition holds for each ABox $\mathcal{A}$ and each query Q: if $\Sigma \cup \mathcal{A}$ is satisfiable, then $\mathsf{cert}(Q, \Sigma, \mathcal{A}) \subseteq \mathsf{cert}(Q, \Sigma', \mathcal{A})$.*

In practice, checking the satisfiability of $\mathcal{T} \cup \mathcal{A}$, which is equisatisfiable with $\Sigma_\mathcal{T} \cup \mathcal{A}$, is easier than (and a prerrequisite for) query answering. Even if satisfiability of $\mathcal{T} \cup \mathcal{A}$ cannot be checked in practice using a complete reasoner for a very large $\mathcal{A}$, we can still compute an upper bound "modulo satisfiability".

Why Translating Back into OWL 2 RL? Our last step was to transform $\mathsf{approx}(\Sigma_\mathcal{T})$ into OWL 2 RL TBox $\mathcal{T}'$. Note, however, that one could obtain the upper bound directly from $\mathsf{approx}(\Sigma_\mathcal{T})$ by first computing the saturation $\mathcal{A}'$ of $\mathsf{approx}(\Sigma_\mathcal{T})$ w.r.t. $\mathcal{A}$ and then computing $\mathsf{cert}(Q, \emptyset, \mathcal{A}')$.

The saturation $\mathcal{A}'$ can be computed in polynomial time [5]; indeed, the rules in $\mathsf{approx}(\Sigma_\mathcal{T})$ are tree-shaped, which can be exploited to obtain a polynomial time saturation procedure. This could lead to better performance in the computation of our upper bound —an interesting topic for future work.

Our current approach, however, does have some advantages. In particular, one can use the same off-the-shelf RL reasoner (such as OWLim) to compute both the lower and upper bound, which is convenient in practice. Furthermore, as suggested by our experiments, reasoners such as OWLim are quite efficient for large data sets.

4 Experiments

We have implemented our approach in Java and used the latest version of OWLim-Lite as an OWL 2 RL reasoner. All experiments have been performed on a desktop computer with 4Gb of RAM, of which 3000Mb were assigned as maximum heap size in Java.

In our experiments, we have used LUBM extended with additional synthetically generated queries. The LUBM ontology contains 93 TBox axioms, out of which 8 contain existential quantification, and 39 are domain and range axioms. The LUBM ontology, however, does not contain disjunction or negation; thus,

Table 1. Number of answers for the 14 LUBM queries

Query	1	2	3	4	5	6	7
Lower Bound	4	0	6	34	719	7356	67
Upper Bound	4	0	6	34	719	7356	67
Query	8	9	10	11	12	13	14
Lower Bound	7356	194	4	212	14	1	5594
Upper Bound	7356	194	4	212	14	1	5594

Table 2. Synthetic queries for which OWLim is incomplete

Query	Lower Bound	Upper Bound	HermiT's answers
3	540	1087	1087
51	0	547	547
67	540	1087	1087
69	0	547	547

the corresponding issues discussed in Section 3.3 do not apply to our experiments. To test how tight our upper bounds are for different queries, we have used LUBM(1,0), the smallest data set in the benchmark, since the complete DL reasoner HermiT can answer all test queries for this data set. LUBM(1,0) contains data for one university and 14 departments, with a total of $100,543$ ABox assertions about $17,174$ different individuals.

4.1 Results for LUBM(1,0)

Standard Queries. LUBM provides 14 queries. As shown in Table 1, the lower and upper bounds computed using OWLim coincide, which allows us to conclude that OWLim is complete for all these queries and the LUBM(1,0) data set. Hence, in these cases, one wouldn't need to use a complete DL reasoner at all.

Additional Synthetic Queries. We have used a synthetic query generation tool [9] to compute 78 additional queries for LUBM. For all these queries, except those in Table 2, lower and upper bounds also coincide. Furthermore, for all queries in Table 2 the upper bound is tight.

Observe, however, that the lower bound is missing those answers which require taking into account LUBM's axioms with existential quantifiers, for which OWLim is not complete. For example, consider the following query

$$Q_{51}(x) = \exists y.(\mathsf{memberOf}(x, y) \land \mathsf{Group}(y))$$

LUBM's ontology contains all the axioms in $\mathcal{T}_{\mathsf{ex}}$, except for the last two axioms in Figure 1; furthermore, LUBM(1,0) contains many instances of RA. Since LUBM's ontology implies that each RA works for (and hence is a member of) some group, all these instances are answers to Q_{51}, which are not computed by OWLim.

Additional Query. For all previous test queries, we have obtained a tight upper bound. To show that this is not always the case, we have manually created the additional query given next.

$$Q(x_1, x_2) = \exists y.\mathsf{works}(x_1, y) \wedge \mathsf{works}(x_2, y) \wedge \mathsf{Group}(y)$$

Since this query is not tree-like, it cannot be answered using HermiT. To obtain the complete answers, we have used REQUIEM[2] to compute a (complete) UCQ rewriting U of Q w.r.t. LUBM's ontology; then, we used OWLim to compute the answers to U w.r.t. the LUBM(1,0) data. For this query, the lower and upper bounds contained zero and 299, 209 answers, respectively; in contrast, we obtained 547 answers using REQUIEM and hence the upper bound is not tight.

As shown in Figure 2, our transformation implies that all RAs work for the same group (represented by the skolem constant c_1); since there are many instances of RA in LUBM(1,0), all pairs of RA instances will be included in the upper bound. Clearly, however, many of these RAs work for different groups.

Note, however, that even for this query we managed to reduce the number of candidate answers from $17,174^2 \sim 3 \times 10^8$ to 299, 209.

4.2 Scalability Tests

To test scalability of upper bound computation, we have performed additional experiments using LUBM data sets of increasing size (from 1 to 10 universities). For each data set and each of the 78 synthetic queries, we have computed lower and upper bounds (using OWLim) and the complete answers (using HermiT). The results are summarised in Figure 4.2, where the time refers to the total query answering time for all test queries.

We can observe similar query answering times and scalability behaviour for lower and upper bound computation. Furthermore, we can observe that HermiT's performance is similar to OWLim's for relatively small data sets. HermiT, however, does not scale well for the largest LUBM data sets. In particular, it takes 178s to complete the tests for 9 universities and runs out of memory for 10 universities; in contrast, OWLim computation times increase more regularly.

5 Conclusion and Future Work

In this paper, we have proposed a novel technique for efficiently computing an upper bound to CQ answers for ontologies given in a expressive DL. Our preliminary experiments on LUBM show that this upper bound might be tight in many cases. Furthermore, we identified cases were lower and upper bounds coincide and hence it is not necessary to use a fully-fledged DL reasoner such as HermiT to compute query answers (HermiT is able to answer rollable queries).

Our work so far, however, is only very preliminary, and there are plenty of possibilities for future work. We are planning to perform experiments with ontologies involving disjunction and negation, and address the issues discussed in

[2] http://www.cs.ox.ac.uk/isg/tools/Requiem/

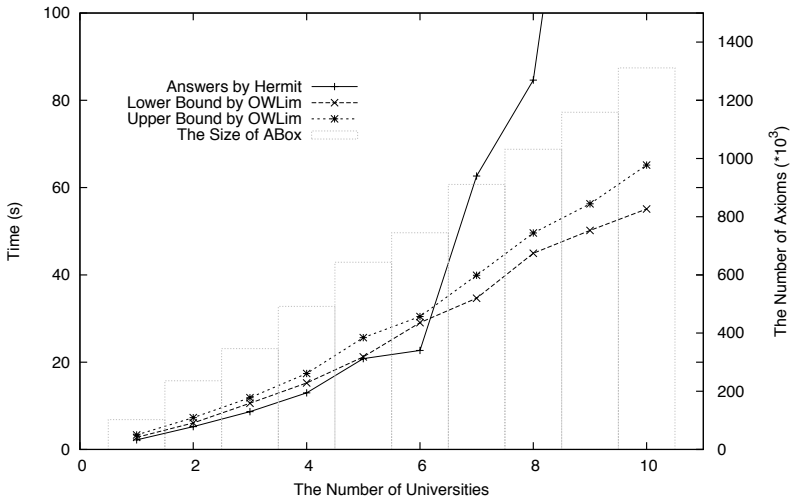

Fig. 3. Running Time Comparison

Section 3.3. Furthermore, we will implement a dedicated datalog engine that can compute the saturation in polynomial time for tree-shaped datalog rules; this might allow us to improve performance as well as to simultaneously compute lower and upper bounds. We will also develop techniques for identifying during upper bound computation the fragments of the ABox and TBox that are sufficient to determine, using a complete reasoner, which of the answers in the upper bound are actual answers; we expect that these techniques will provide additional room for optimisation. As a result, we can integrate all these techniques in HermiT to optimise query answering.

Acknowledgements. This work was supported by the Royal Society, the EU FP7 project SEALS and the EPSRC projects ConDOR, ExODA, and LogMap.

References

1. Baader, F., Sertkaya, B., Turhan, A.: Computing the least common subsumer wrt a background terminology. J. of Applied Logic (JAL) 5(3), 392–420 (2007)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI (2005)
3. Cali, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: Proc. of LICS (2010)
4. 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 (JAR) 39(3), 385–429 (2007)

5. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity conditions and their application to query answering in description logics. In: Proc. of KR (2012)
6. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Semamtics (JWS) 6(4), 309–322 (2008)
7. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Completeness guarantees for incomplete ontology reasoners: Theory and practice. J. of Artificial Intelligence Research (JAIR) 43 (2012)
8. Del Val, A.: First order lub approximations: characterization and algorithms. Artificial Intelligence 162(1-2), 7–48 (2005)
9. Grau, B.C., Stoilos, G.: What to ask to an incomplete semantic web reasoner? In: Proc. of IJCAI. pp. 419–476 (2011)
10. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. of WWW (2003)
11. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for OWL knowledge base systems. J. Web Semantics (JWS) 3(2-3), 158–182 (2005)
12. Haarslev, V., Möller, R.: RACER system description. J. of Automated Reasoning (JAR) pp. 701–705 (2001)
13. Haarslev, V., Möller, R., Wessel, M.: Querying the semantic web with RACER+NRQL. In: Proc. of ADL (2004)
14. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR (2006)
15. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of KR (2008)
16. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM–a pragmatic semantic repository for OWL. In: Proc. of WISE Workshops (2005)
17. Kollia, I., Glimm, B., Horrocks, I.: Query answering over sroiq knowledge bases with SPARQL. In: Proc. of DL (2011)
18. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. of IJCAI. vol. 9 (2009)
19. McBride, B.: Jena: Implementing the rdf model and syntax specification. In: Semantic Web Workshop. vol. 40 (2001)
20. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research (JAIR) 36(1), 165–228 (2009)
21. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (2009)
22. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. of Applied Logic (JAL) 8(2), 186–209 (2010)
23. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for tbox reasoning. In: Proc. of AAAI (2010)
24. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Proc. of DL. vol. 46 (2007)
25. Rudolph, S., Krötzsch, M., Hitzler, P., Sintek, M., Vrandecic, D.: Efficient owl reasoning with logic programs–evaluations. Web Reasoning and Rule Systems (2007)
26. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! J. Artif. Intell. Res. (JAIR) 39, 429–481 (2010)
27. Selman, B., Kautz, H.: Knowledge compilation and theory approximation. J. of the ACM (JACM) 43(2), 193–224 (1996)
28. Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. J. Automated Reasoning (JAR) pp. 292–297 (2006)