Optimized Query Rewriting in OWL 2 QL

Alexandros Chortaras, Despoina Trivela, and Giorgos Stamou

School of Electrical and Computer Engineering,
National Technical University of Athens,
Zografou 15780, Athens, Greece
{achort,gstam}@cs.ntua.gr, despoina@image.ntua.gr

Abstract. The OWL 2 QL profile has been designed to facilitate query answering via query rewriting. This paper presents an optimized query rewriting algorithm which takes advantage of the special characteristics of the query rewriting problem via first-order resolution in OWL 2 QL and computes efficiently the set of the non redundant rewritings of a user query, by avoiding blind and redundant inferences, as well as by reducing the need for extended query subsumption checks. The evaluation shows that in several cases the algorithm achieves a significant improvement and better scalability if compared to other similar approaches.

Keywords: query answering, query rewriting, OWL 2 QL, DL-Lite.

1 Introduction

The use of ontologies in data access allows for semantic query answering, i.e. for answering user queries expressed in terms of terminologies linked to the data [3, 5]. Queries are typically expressed in the form of *conjunctive queries* (CQ) and terminologies in an ontological form. Unfortunately, the problem of answering CQs in terms of ontologies axiomatized in expressive Description Logics (DL) suffers from high worst-case complexity. The obvious way to overcome this obstacle and develop practical systems is to reduce the expressivity of the ontology language, otherwise either the soundness or the completeness of the algorithm have to be sacrificed.

Late research in description logics introduced DL-Lite$_R$, which is the DL ontology representation language that underpins the OWL 2 QL profile [1]. In DL-Lite$_R$, the CQ answering problem is tractable (from the data point of view). It is proved that sound and complete CQ answering systems for DL-Lite$_R$ can follow a strategy that splits the procedure into two independent steps [5, 1, 6]: the query *rewriting*, in which the CQ is expanded into a union of CQs (UCQ) using the ontology and the *execution* of the UCQ over the database. Apart from the obvious advantage of using the mature relational database technology, rewriting can be based on first order resolution-based reasoning algorithms [4] that are widely studied in the literature [7, Ch.2]. The main restriction is that for large terminologies and/or large queries, the algorithm becomes rather impractical, since the exponential complexity in the query size affects the efficiency of the system and may result to a very large number of rewritings.

Several CQ answering algorithms for DL-Lite$_R$ have been proposed in the literature. In [2, 8], the rewriting strategy, called PerfectRef, is based on reformulating the conjuncts of the query according to the taxonomic information of the ontology. Although the strategy is in general very effective, some of the axioms of the terminology must be syntactically rewritten in terms of auxiliary roles in order to cover the full expressivity of DL-Lite$_R$ which may significantly increase the size of the ontology. In [4] a resolution-based rewriting strategy, called RQR, is proposed, which relaxes the above restriction. However, the unstratified saturation strategy may get tangled in long inference paths leading either to redundant rewritings or to rewritings that are not in a CQ form (non function-free). Such rewritings are discarded in the end, however their participation in the inference process and the increased number of required subsumption checks degrades significantly the practical performance of the system (which already suffers from an exponential worst-case complexity in the query size). Finally, [6] follows a different approach, and instead of computing a set of CQs, a non-recursive datalog program is constructed, deferring thus the main source of complexity to the database system.

In this paper, we introduce a new query rewriting algorithm called Rapid, which is optimized for the query rewriting problem of CQs posed over DL-Lite$_R$ ontologies. Its efficiency is based on the selective and stratified application of resolution rules. Instead of applying indiscriminately the resolution rule saturating the user query, we take advantage of the query structure and apply a restricted sequence of resolutions that may lead to useful, hopefully redundant-free rewriting subsets. In this way, we avoid a large number of blind inference paths which can be the cause of scalability issues, as well as the production of many redundant rewritings and the need to perform extended query subsumption checks, i.e. very costly operations. The effectiveness of the algorithm is demonstrated in its practical evaluation, which shows clearly an optimized performance, especially in the most problematic cases of large queries or large terminologies.

2 Preliminaries

A DL-Lite$_R$ *ontology* is a tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the *terminology*, which represents the entities of the world, and $\mathcal{A}$ the assertional knowledge, which describes the world objects in terms of the entities of $\mathcal{T}$. Formally, $\mathcal{T}$ is a set of axioms of the form $C_1 \sqsubseteq C_2$ or $R_1 \sqsubseteq R_2$, where C_1, C_2 are concept descriptions and R_1, R_2 role descriptions, employing atomic concepts, atomic roles and individuals. $\mathcal{A}$ is a finite set of *assertions* of the form $A(a)$ or $R(a, b)$, where a, b are individuals, A an atomic concept and R an atomic role. A DL-Lite$_R$ concept can be either atomic or $\exists R.\top$. If it appears in the RHS, we assume that it may also be of the form $\exists R.A$. Negations of concepts can be used only in the RHS of subsumption axioms. A DL-Lite$_R$ role is either an atomic role R or its inverse R^-.

A CQ Q has the form $\mathbf{A} \leftarrow \{\mathbf{B}_i\}_{i=1}^n$ (the sequence is meant as a conjunction), where atom $\mathbf{A}$ is the *head* and atoms $\mathbf{B}_i$ the *body* of Q. Wlog we assume that all $\mathbf{B}_i$ are distinct and denote the set of $\mathbf{B}_i$s by $\mathsf{body}\, Q$, and $\mathbf{A}$ by $\mathsf{head}\, Q$. A CQ Q is

posed over an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ if the predicates of all atoms in $\mathsf{body}\, Q$ are entities of $\mathcal{T}$ and have arities 1 or 2, depending on whether the respective entity is a concept or a role. Hence, an atom $\mathbf{A} \in \mathsf{body}\, Q$ may be of the form $A(t)$ or $A(t, s)$, if it is a *concept* or *role atom*, respectively. $\mathsf{terms}\, \mathbf{A}$ ($\mathsf{vars}\, \mathbf{A}$, $\mathsf{cons}\, \mathbf{A}$) are the sets of terms (variables, constants) that appear in $\mathbf{A}$. For a set of atoms $\mathcal{B}$ we have that $\mathsf{terms}\, \mathcal{B} = \bigcup_{\mathbf{A} \in \mathcal{B}} \mathsf{terms}\, \mathbf{A}$, for a CQ Q that $\mathsf{terms}\, Q = \mathsf{terms}\, (\{\mathsf{head}\, Q\} \cup \mathsf{body}\, Q)$, and similarly for $\mathsf{vars}\, Q$ and $\mathsf{cons}\, Q$. An atom or CQ is *function free* if it contains no functional terms. User queries are always function free. The terms in a function free CQ Q can be characterized in some important ways: A term $t \in \mathsf{terms}\, Q$ is *distinguished* iff it appears in $\mathsf{head}\, Q$ and *non distinguished* otherwise, *bound* iff it is a constant or it is a variable that appears at least twice in $\{\mathsf{head}\, Q\} \cup \mathsf{body}\, Q$ and *unbound* otherwise, and *disconnected* iff there is a disconnected subgraph (V', E') of $(\mathsf{terms}\, Q,\ \{\{t, s\} \mid R(t, s) \text{ or } R(s, t) \in \mathsf{body}\, Q\})$ such that $t \in V'$ and V' contains no distinguished variable. We denote the set of distinguished and bound terms, and bound and unbound variables of Q by $\mathsf{terms}^{\mathsf{D}}\, Q$, $\mathsf{terms}^{\mathsf{B}}\, Q$, $\mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}^{\mathsf{UB}}\, Q$, respectively. For simplicity and wlog we assume that the user query Q contains no disconnected terms, no distinguished constants, and that all its distinguished variables appear also in $\mathsf{body}\, Q$.

A tuple of constants $\boldsymbol{a}$ is a *certain answer* of a user CQ Q posed over the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ iff $\mathcal{O} \cup \{Q\} \models C(\boldsymbol{a})$, where $C(\boldsymbol{a})$ is subsumed by $\mathsf{head}\, Q$. The set that contains all answers of Q over $\mathcal{O}$ is denoted by $\mathsf{cert}\, (Q, \mathcal{O})$. It has been proved [5, 1] that for any CQ Q and DL-Lite$_R$ ontology $\mathcal{O}$, there is a set $\mathcal{Q}$ of function free CQs (called query rewritings) such that $\mathsf{cert}(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \bigcup_{Q' \in \mathcal{Q}} \mathsf{cert}(Q', \langle \emptyset, \mathcal{A} \rangle)$. The set of these rewritings may be computed by first clausifying $\mathcal{O}$, i.e. translating $\mathcal{O}$ into a set of first order clauses $\Xi(\mathcal{O})$ according to table 1, and then performing resolution operations on $\Xi(\mathcal{O})$ and Q.

Table 1. Translation of DL-Lite$_R$ axioms into clauses of $\Xi(\mathcal{O})$ (reproduced from [4]).

Axiom	Clause	Axiom	Clause
$A \sqsubseteq B$	$B(x) \leftarrow A(x)$		
$P \sqsubseteq S$ $P^- \sqsubseteq S^-$	$S(x, y) \leftarrow P(x, y)$	$P \sqsubseteq S^-$ $P^- \sqsubseteq S$	$S(x, y) \leftarrow P(y, x)$
$\exists P \sqsubseteq A$	$A(x) \leftarrow P(x, y)$	$\exists P^- \sqsubseteq A$	$A(x) \leftarrow P(y, x)$
$A \sqsubseteq \exists P$	$P(x, f^A_P(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P^-$	$P(f^A_{P^-}(x), x) \leftarrow A(x)$
$A \sqsubseteq \exists P.B$	$P(x, f^A_{P.B}(x)) \leftarrow A(x)$ $B(f^A_{P.B}(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P^-.B$	$P(f^A_{P^-.B}(x), x) \leftarrow A(x)$ $B(f^A_{P^-.B}(x)) \leftarrow A(x)$

Formally, a function free CQ Q' is a *rewriting* of a CQ Q posed over ontology $\mathcal{O}$, iff Q and Q' have the same head predicate and $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$. Nevertheless, not all possible rewritings are needed for the complete computation of $\mathsf{cert}\, (Q, \mathcal{O})$, since some of them may be equivalent or subsumed by others. We say that a CQ Q *subsumes* a CQ Q' (or Q' *is subsumed by* Q) and write $Q \rhd Q'$, iff there is a substitution θ such that $\mathsf{head}\, (Q\theta) = \mathsf{head}\, Q'$ and $\mathsf{body}\, (Q\theta) \subseteq \mathsf{body}\, Q'$. If

Q and Q' are mutually subsumed they are *equivalent*, and we write $Q \hat{=} Q'$. If $\mathcal{Q}$ is a set of CQs and for some CQ Q there is a $Q' \in \mathcal{Q}$ such that $Q \hat{=} Q'$ we write $Q \hat{\in} \mathcal{Q}$. We define also the operation $\mathcal{Q} \hat{\cup} \{Q\} = \mathcal{Q} \cup \{Q\}$ if $Q \hat{\notin} \mathcal{Q}$, and $\mathcal{Q} \hat{\cup} \{Q\} = \mathcal{Q}$ otherwise. Given a CQ Q, a set $\mathcal{Q}$ is a *set of rewritings* of Q over $\mathcal{O}$ and is denoted by $\mathsf{rewr}(Q, \mathcal{O})$ iff for each rewriting Q'' of Q over $\mathcal{O}$ there is a unique $Q' \in \mathcal{Q}$ with $Q' \hat{=} Q''$. If for some CQ Q there is a $Q' \in \mathcal{Q}$ for which $Q' \rhd Q$, Q is called to be *redundant* in $\mathcal{Q}$. Given a CQ Q, let Q' be the CQ $\mathsf{head}\, Q \leftarrow \{\mathbf{B}\}_{\mathbf{B} \in \mathcal{B}}$ for some $\mathcal{B} \subseteq \mathsf{body}\, Q$. If $\mathcal{B}$ is the smallest subset of $\mathsf{body}\, Q$ for which $Q' \rhd Q$, the atoms $\mathsf{body}\, Q \setminus \mathcal{B}$ are called *redundant* and Q' *minimal*. We denote the minimal CQ obtained from Q by $\mathsf{min}\, Q$. Obviously, $\mathsf{min}\, Q$ always subsumes Q. Because the redundant atoms of a non minimal CQ can be removed without affecting its answers, in order to compute $\mathsf{cert}(Q, \mathcal{O})$, we are interested in minimal, non redundant rewritings of Q. We say that Q' is a *core rewriting* of a CQ Q over $\mathcal{O}$, iff $Q' \hat{=} \mathsf{min}\, Q''$ for some $Q'' \hat{\in} \mathsf{rewr}(Q, \mathcal{O})$ (obviously we may have $Q' \hat{=} Q''$ if Q'' is already minimal) and there is no CQ $Q''' \hat{\in} \mathsf{rewr}(Q, \mathcal{O})$ such that $\mathsf{min}\, Q''' \rhd Q'$. The *set of the core rewritings* $\mathsf{rewr}^{\mathsf{C}}(Q, \mathcal{O})$ of Q over $\mathcal{O}$ is defined in the same way as $\mathsf{rewr}(Q, \mathcal{O})$ by including in it all the core rewritings.

3 The Rapid Algorithm

Rapid computes $\mathsf{rewr}^{\mathsf{C}}(Q, \mathcal{O})$ for a user query Q in an efficient way. Its structure is similar to that of RQR, but it introduces several optimizations and organizes differently some tasks so as to reduce the inferences that lead to queries that will eventually be discarded because they are not function free or redundant. The strategy of Rapid is based on the distinguishing property of the bound and unbound variables, namely that whenever a CQ Q is used as the main premise in a resolution rule in which an atom $\mathbf{A} \in \mathsf{body}\, Q$ unifies with the head of the side premise and the mgu θ contains a binding v/t for some variable $v \in \mathsf{vars}^{\mathsf{B}}\, Q$, the application of θ affects several atoms of the query apart from $\mathbf{A}$. This is not the case if $v \in \mathsf{vars}^{\mathsf{UB}}\, Q$, since unbound variables appear only once in a CQ. The main premise in the resolution rules in Rapid is always Q or a rewriting of it.

Rapid consists of the following steps: (1) The *clausification* step, in which O is transformed into $\Xi(\mathcal{O})$. (2) The *shrinking* step, in which the clauses of $\Xi(\mathcal{O})$ are selectively used as side premises of resolution inference rules in order to compute rewritings which differ from the user query Q in that they do not contain one or more variables in $\mathsf{vars}^{\mathsf{B}}\, Q$, because the application of the resolution rule led to their unification with a functional term which subsequently was eliminated. (3) The *unfolding* step, which uses the results of the previous step to compute the remaining rewritings of Q, by applying the resolution rule without that the bound variables of the main premise are affected. In principle, only unbound variables are eliminated or introduced at this step. However, eventually some bound variables of the main premise may also be eliminated, not through the introduction and subsequent elimination of functional terms, but through the removal of redundant atoms, i.e. while converting the conclusion to its minimal form. (4) The *redundancy removal* step, in which redundant rewritings are re-

moved. This step is in principle the same as in RQR, but is more efficient in two ways: First, the previous steps produce much fewer redundant rewritings, and second not every pair of rewritings has to be checked for subsumption, because, as we will see in the sequel, some sets of rewritings that are produced at the unfolding step are guaranteed to be redundant free.

Notwithstanding the above general description, Rapid optimizes the shrinking and unfolding steps by not employing the resolution rule directly in its standard form. Instead, a shrinking and unfolding inference rule are defined, which condensate into one a series of several successful resolution rule application steps. In this way, we achieve that the resolution rule is applied only if it eventually leads to a function free and hopefully also non redundant rewriting, avoiding thus a large number of useless inferences. The shrinking and unfolding inference rules are based on the notion of the unfolding and function set of an atom.

3.1 Atom Unfolding Sets

The saturation of $\Xi(\mathcal{O})$ w.r.t. the resolution rule contains clauses of the form

$$A(x) \leftarrow B(x),\ A(x) \leftarrow B(x,y),\ A(x,y) \leftarrow B(x,y),$$
$$A(x, f(x)) \leftarrow B(x),\ A(x, f(x)) \leftarrow B(x,y),$$
$$A(g(x), f(g(x))) \leftarrow B(x),\ A(f(x), f(g(x))) \leftarrow B(x,y),$$
$$A(g(h(x)), f(g(h(x)))) \leftarrow B(x),\ A(g(h(x)), f(g(h(x)))) \leftarrow B(x,y),\ \ldots$$

as well as the respective clauses with the role atom arguments inverted. We note that in the first two rows the non functional arguments appear both in the LHS and the RHS of the clauses. Based on this remark and given that in the unfolding step we want that bound variables do not unify with functional terms but be preserved in the conclusion, we define the *unfolding* of an atom as follows:

Definition 1. *Let* $\mathbf{A}$ *be a function free atom and* T *a non empty subset of* $\mathsf{terms}\,\mathbf{A}$*. Atom* $\mathbf{B}\theta'$ *is an* unfolding *of* $\mathbf{A}$ *w.r.t.* T *iff* $\Xi(\mathcal{O}) \models_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ *for some substitution* θ *on* $\mathsf{vars}\,\mathbf{A} \setminus T$*, where* θ' *is a renaming of* $\mathsf{vars}\,\mathbf{B} \setminus T$ *such that for* $v \in \mathsf{vars}\,\mathbf{B} \setminus T$ *we have that* $v\theta' \notin \mathsf{vars}\,\mathbf{A}$*.*

Essentially, $\mathbf{B}$ (up to the variable renaming θ') is an unfolding of $\mathbf{A}$ w.r.t. T if $\mathbf{B}$ is the body of a clause that can be inferred from $\Xi(\mathcal{O})$ having in its head an atom $\mathbf{A}'$ (of the same predicate as $\mathbf{A}$) such that both $\mathbf{B}$ and $\mathbf{A}'$ contain unaltered all terms in T (which will normally be the bound variables of the CQ in question). The renaming θ' contains no essential information, so in order to collect all *distinct* unfoldings of $\mathbf{A}$ w.r.t. some T, it suffices to collect only $\mathbf{B}\theta'$ for any convenient θ' for each $\mathbf{B}$ such that $\Xi(\mathcal{O}) \models_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$. Having this in mind, we define the *unfolding set* of atom $\mathbf{A}$ for T w.r.t. $\Xi(\mathcal{O})$ as the set $\mathcal{D}(\mathbf{A};T) = \{\mathbf{B} \mid \Xi(\mathcal{O}) \cup \{\mathbf{A}\} \models_{\mathcal{J}} \mathbf{B}\}$. We also define the set $\dot{\mathcal{D}}(\mathbf{A};T) = \mathcal{D}(\mathbf{A};T) \cup \{\mathbf{A}\}$. By using table 2, it is easy to prove that given $\mathbf{A}$ and T we have that $\Xi(\mathcal{O}) \models \mathbf{A}\theta \leftarrow \mathbf{B}$ iff $\mathbf{B}\theta' \in \mathcal{D}(\mathbf{A};T)$. In the definition of $\mathcal{D}$, $\mathcal{J}$ are the following atom production rules:

T	rule	T	rule
$\{t\}$	$\dfrac{A(t) \quad A(x) \leftarrow B(x)}{B(t)}$		
$\{t\}$	$\dfrac{A(t) \quad A(x) \leftarrow P(x,y)}{P(t,z)}$	$\{\mathrm{t}\}$	$\dfrac{A(t) \quad A(x) \leftarrow P(y,x)}{P(z,t)}$
$\{t\}$	$\dfrac{P(t,v) \quad P(x,f(x)) \leftarrow B(x)}{B(t)}$	$\{t\}$	$\dfrac{P(v,t) \quad P(f(x),x) \leftarrow B(x)}{B(t)}$
$\{t\}$, $\{s\}$ or $\{t,s\}$	$\dfrac{P(t,s) \quad P(x,y) \leftarrow R(x,y)}{R(t,s)}$	$\{t\}$, $\{s\}$ or $\{t,s\}$	$\dfrac{P(t,s) \quad P(x,y) \leftarrow R(y,x)}{R(s,t)}$

Fig. 1. The $\mathcal{J}(T)$ inference rules.

$\mathbf{A}$	T	$\mathbf{B}\theta'$	$\mathbf{A}\theta \leftarrow \mathbf{B}$	θ	θ'
$A(t)$	$\{t\}$	$B(t)$	$A(t) \leftarrow B(t)$	$\emptyset$	$\emptyset$
$A(t)$	$\{t\}$	$P(t,y)$ / $P(y,t)$	$A(t) \leftarrow P(t,z)$ / $\leftarrow P(z,t)$	$\emptyset$	$\{z/y\}$
$P(t,v)$	$\{t\}$	$B(t)$	$P(t,f(t)) \leftarrow B(t)$	$\{v/f(t)\}$	$\emptyset$
$P(t,v)$	$\{t\}$	$R(t,y)$ / $R(y,t)$	$P(t,f(t)) \leftarrow R(t,z)$ / $\leftarrow R(z,t)$	$\{v/f(t)\}$	$\{z/y\}$
$P(t,t')$	$\{t\}$	$R(t,t')$ / $R(t',t)$	$P(t,t') \leftarrow R(t,t')$ / $\leftarrow R(t',t)$	$\emptyset$	$\emptyset$
$P(v,t)$	$\{t\}$	$B(t)$	$P(f(t),t) \leftarrow B(t)$	$\{v/f(t)\}$	$\emptyset$
$P(v,t)$	$\{t\}$	$R(t,y)$ / $R(y,t)$	$P(f(t),t) \leftarrow R(t,z)$ / $\leftarrow R(z,t)$	$\{v/f(t)\}$	$\{z/y\}$
$P(t',t)$	$\{t\}$	$R(t',t)$ / $R(t,t')$	$P(t',t) \leftarrow R(t',t)$ / $\leftarrow R(t,t')$	$\emptyset$	$\emptyset$
$P(t,s)$	$\{t,s\}$	$R(t,s)$ / $R(s,t)$	$P(t,s) \leftarrow R(t,s)$ / $\leftarrow R(s,t)$	$\emptyset$	$\emptyset$

Table 2. Unfolding table

3.2 Atom Function Sets

As we have mentioned before, the saturation of $\Xi(\mathcal{O})$ contains clauses of the form $A(x, f(x)) \leftarrow B(x)$, $A(f(x), x) \leftarrow B(x)$, and $A(f(x)) \leftarrow B(x)$, as well as of the form $A(g(x), f(g(x))) \leftarrow B(x)$ and $A(g(x), f(g(x))) \leftarrow B(x, y)$. In contrast to the case of the unfoldings, now we are interested in the behavior of the functional term $f(x)$, which is present in the LHS but not in the RHS of the clause. Hence, if $f(x)$ appears in an atom $\mathbf{A}$ of some rewriting, it could probably be eliminated by using such clauses. Let $\mathsf{funcs}\,\Xi(\mathcal{O})$ be the set of all functions that appear in $\Xi(\mathcal{O})$. According to table 1, each axiom that has an existential quantifier in the RHS introduces a distinct function f in $\Xi(\mathcal{O})$. It follows that each function $f \in \mathsf{funcs}\,\Xi(\mathcal{O})$ is uniquely associated with the concept A that appears in the LHS of the axiom that introduces f. We denote the concept associated with function f by $\mathsf{cn}\,f$. In order to group the functional term elimination clauses through the use of $\mathsf{cn}\,f$, we provide the following definition:

Definition 2. *Let* $\mathbf{A}$ *be a function free atom,* T *a non empty subset of* $\mathsf{terms}\,\mathbf{A}$ *and* v *a variable in* $\mathsf{vars}\,\mathbf{A} \cap T$*. The* function set $\mathcal{F}_v(\mathbf{A}; T)$ *of all functions associated with* $\mathbf{A}$ *in variable* v *w.r.t.* T *is defined as follows:*

$$\mathcal{F}_v(\mathbf{A};T) = \begin{array}{l} \{f \mid B(v) \in \hat{\mathcal{D}}(\mathbf{A};T) \text{ and } B(f(x)) \leftarrow (\mathsf{cn}\,f)(x) \in \Xi(\mathcal{O})\} \cup \\ \{f \mid B(v,t) \in \hat{\mathcal{D}}(\mathbf{A};T) \text{ and } B(f(x),x) \leftarrow (\mathsf{cn}\,f)(x) \in \Xi(\mathcal{O})\} \cup \\ \{f \mid B(t,v) \in \hat{\mathcal{D}}(\mathbf{A};T) \text{ and } B(x,f(x)) \leftarrow (\mathsf{cn}\,f)(x) \in \Xi(\mathcal{O})\} \end{array}$$

It follows immediately that (a) if $\mathbf{A}$ is of the form $A(v,t)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \models_{\mathcal{R}} A(f(t),s) \leftarrow (\mathsf{cn}\, f)(t)$, (b) if $\mathbf{A}$ is of the form $A(t,v)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \models_{\mathcal{R}} A(s,f(t)) \leftarrow (\mathsf{cn}\, f)(t)$, where in both cases $s = t$ if $t \in T$ otherwise either $s = t$, or $s = g(f(t))$ for some function g, and (c) if $\mathbf{A}$ is of the form $A(v)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \models_{\mathcal{R}} A(f(t)) \leftarrow (\mathsf{cn}\, f)(t)$.

Example 1. Consider the ontology $\mathcal{O} = \{B \sqsubseteq A, \exists R \sqsubseteq A, S \sqsubseteq R^-, C \sqsubseteq \exists R.A, \exists T^- \sqsubseteq C, D \sqsubseteq \exists S\}$, from which we get $\Xi(\mathcal{O}) = \{A(x) \leftarrow B(x), A(x) \leftarrow R(x,y), R(x,y) \leftarrow S(y,x), R(x,f_1(x)) \leftarrow C(x), A(f_1(x)) \leftarrow C(x), C(x) \leftarrow T(y,x), S(x,f_2(x)) \leftarrow D(x)\}$. Table 3 shows the unfolding and function sets for atoms $A(x)$, $C(x)$, $R(x,y)$ and $S(x,y)$ and several sets T.

$\mathbf{A};T$	$A(x);\{x\}$	$C(x);\{x\}$	$R(x,y);\{x\}$	$R(x,y);\{y\}$	$R(x,y);\{x,y\}$	$S(x,y);\{x\}$
$\mathcal{D}(\mathbf{A};T)$	$B(x)$ $R(x,z_1)$ $S(z_1,x)$ $C(x)$ $T(z_2,x)$	$T(z_1,x)$	$S(y,x)$ $C(x)$ $T(z_1,x)$	$S(y,x)$ $D(y)$	$S(y,x)$	$D(x)$
$\mathcal{F}_x(\mathbf{A};T)$	$\{f_1,f_2\}$	$\emptyset$	$\{f_2\}$	$\emptyset$	$\{f_2\}$	$\emptyset$
$\mathcal{F}_y(\mathbf{A};T)$	–	–	$\emptyset$	$\{f_1\}$	$\{f_1\}$	–

Table 3. Unfolding and function sets for example 1.

3.3 Query Shrinking

The shrinking step computes rewritings that can be inferred from the user query Q by eliminating one or more of its bound variables through their unification with a functional term. Given that the rewritings in $\mathsf{rewr}\,(Q,\mathcal{O})$ are function free, if a function is introduced in some rewriting at some point in the standard resolution-based inference process, subsequently it must be eliminated. However, we know that each function appears only in at most two clauses of $\Xi(\mathcal{O})$, both of which have as body the atom $(\mathsf{cn}\, f)(x)$. Now, $f(x)$ can be introduced in a CQ only if some inference led to the substitution of a bound variable v by $f(x)$. Hence, in order for $f(x)$ to be eliminated, all atoms in which $f(x)$ has been introduced must contain f in their function sets, for the appropriate argument. This is the intuition behind the following shrinking inference rule:

Definition 3. *Let Q be a CQ and v a non distinguished bound variable of Q. Write Q in the form $\mathbf{A} \leftarrow \mathbf{B}_1,\ldots,\mathbf{B}_k,\mathbf{C}_1,\ldots,\mathbf{C}_n$, where $\mathbf{B}_i$ are the atoms in $\mathsf{body}\, Q$ that contain v, and $\mathbf{C}_i$ the remaining atoms. Let also $\mathcal{C} = \bigcup_{i=1}^k \mathsf{cons}\, \mathbf{B}_i$ and $\mathcal{X} = \bigcup_{i=1}^k (\mathsf{vars}^{\mathsf{B}}\, Q \cap \mathsf{vars}\, \mathbf{B}_i) \setminus v$. The shrinking rule $\mathcal{S}$ on Q has as follows:*

$$\frac{\mathbf{A} \leftarrow \mathbf{B}_1,\ldots,\mathbf{B}_k,\mathbf{C}_1,\ldots,\mathbf{C}_n \quad f \in \bigcap_{i=1}^k \mathcal{F}_v(\mathbf{B}_i;\mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B}_i) \wedge |\mathcal{C}| \leq 1}{\mathsf{min}\,(\mathbf{A}\theta \leftarrow (\mathsf{cn}\, f)(t),\mathbf{C}_1\theta,\ldots,\mathbf{C}_n\theta)}$$

where $\theta = \bigcup_{x \in \mathcal{X}} \{x/t\}$, and $t = a$ if $\mathcal{C} = \{a\}$ otherwise t is a variable $\notin \mathsf{vars}\, Q$.

The shrinking rule changes the structure of Q, in the sense that it eliminates a bound variable, and hence the atoms that contained it. It is easy to prove that $\mathcal{S}$ is a sound inference rule, i.e. that if $\Xi(\mathcal{O}) \cup \{Q\} \models_{\mathcal{S}} Q'$ then $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$.

3.4 Query Unfolding

Let $\mathcal{S}^*(Q)$ be the reflexive transitive closure of Q under the inference rule $\mathcal{S}$. From its computation, it follows that $\mathcal{S}^*(Q)$ contains a 'representative' for all query structures that can result from Q by eliminating one or more variables in $\mathsf{vars}^{\mathsf{B}}\, Q$ by using functional terms. Moreover, this representative can be considered as a 'top' query, in the sense that in can produce several more CQs with no further structural changes due to bindings of bound variables with functional terms. Hence, the remaining rewritings can be obtained from $\mathcal{S}^*(Q)$ by computing for each $Q' \in \mathcal{S}^*(Q)$ all CQs inferred from Q' by replacing one or more of its atoms by one of its unfoldings. In this way we can eventually compute all rewritings of Q. This can be achieved by applying the following unfolding inference rule:

Definition 4. *Let Q be the CQ $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$. The unfolding rule $\mathcal{U}$ on Q has as follows:*

$$\frac{\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n \qquad \mathbf{C} \in \mathcal{D}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B}_i)}{\mathsf{min}\,(\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{C}\gamma, \mathbf{B}_{i+1}, \ldots, \mathbf{B}_n)}$$

where γ is a renaming of $\mathsf{vars}\,\mathbf{C} \setminus \mathsf{vars}^{\mathsf{B}}\, Q$ such that $x\gamma \notin \bigcup_{j=1, j\neq i}^{n} \mathsf{vars}\,\mathbf{B}_j$ for all $x \in \mathsf{vars}\,\mathbf{C} \setminus \mathsf{vars}^{\mathsf{B}}\, Q$.

It follows immediately that $\mathcal{U}$ is a sound inference rule, i.e. that if $\Xi(\mathcal{O}) \cup \{Q\} \models_{\mathcal{U}} Q'$ then $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$. Rule $\mathcal{U}$ replaces one atom of Q by one of its unfoldings. $\mathcal{U}$ can iteratively be applied on the conclusion to get more rewritings. In order to facilitate the optimization of the combined application of $\mathcal{U}$ on more than one atoms at the same time, we define the *combined unfolding* inference rule $\mathcal{W}$ which can replace in one step more than one atoms of Q by one of their unfoldings. In this way, any *unfolding* of Q can be obtained in one step.

Definition 5. *An unfolding of CQ $Q : \mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, is the conclusion of any application of the following $\mathcal{W}$ rule:*

$$\frac{\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n \qquad \mathbf{C}_i \in \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B}_i) \text{ for } i = 1 \ldots n}{\mathsf{min}\,(\mathbf{A} \leftarrow \mathbf{C}_1\gamma_1, \ldots, \mathbf{C}_n\gamma_n)}$$

where γ_i is a renaming of $\mathsf{vars}\,\mathbf{C_i} \setminus \mathsf{terms}^{\mathsf{B}}\, Q$ such that $x\gamma_i \notin \bigcup_{j=1, j\neq i}^{n} \mathsf{vars}\,(\mathbf{C}_j\gamma_j)$ for all $x \in \mathsf{vars}\,\mathbf{C}_i \setminus \mathsf{terms}^{\mathsf{B}}\, Q$.

In the above definitions we have always taken the minimal form of the conclusion in order to remove possible redundant atoms. In the sequel, we will call *raw* an unfolding given by $\mathcal{U}$ or $\mathcal{W}$ before converting it to the minimal form.

We denote the reflexive transitive closure of all possible applications of the inference rule $\mathcal{W}$ on Q by $\mathcal{W}^*(Q)$. By using the definitions of the shrinking and unfolding rules we get the following theorem, which justifies the strategy followed by Rapid to compute the set of non redundant rewritings of a user query Q.

Theorem 1. *Let Q be a CQ over a DL-Lite$_R$ ontology $\mathcal{O}$. We have that if $Q' \in \bigcup_{Q'' \in \mathcal{S}^*(Q)} \mathcal{W}^*(Q'')$ then $Q' \hat{\in} \mathsf{rewr}(Q, \mathcal{O})$ (soundness), and that if $Q' \in \mathsf{rewr}^{\mathsf{C}}(Q, \mathcal{O})$ then $Q' \hat{\in} \bigcup_{Q'' \in \mathcal{S}^*(Q)} \mathcal{W}^*(Q'')$ (completeness).*

Proof (Sketch). Soundness follows from the soundness of the $\mathcal{S}$ and $\mathcal{W}$ rules. For completeness, if Q' is inferred by a sequence of resolution rule applications with main premises Q, Q_1, ..., Q_{l-1} and conclusions Q_1, Q_2, ..., Q', it must be shown that there is a sequence of shrinking rule applications with main premises Q, Q^s_1, ..., $Q^s_{l_s-1}$ and conclusions Q^s_1, Q^s_2, ..., $Q^s_{l_s}$, and a sequence of unfolding rule applications with main premises $Q^s_{l_s}$, Q^u_1, ..., $Q^u_{l_u-1}$ and conclusions Q^u_1, Q^u_2, ..., Q'. CQs $Q_1, \ldots, Q_{l-1}$ may contain functional terms of the form $f_1(\cdots f_k(t))$ for $k \geq 0$ (k is called depth). But Q and Q' are both function free, hence any functional terms that appear in the intermediate steps are eventually eliminated. The result can be proved by induction on the maximum depth d of the intermediate queries $Q_1, \ldots, Q_{m-1}$, by showing that the sequence of resolution rule applications that led to the introduction of the functional term of depth d can be rearranged so that only functional terms of depth 1 are introduced. Moreover, these rules can be applied first, thus giving rise to the shrinking rules sequence, on whose final conclusion the unfolding rules in order to get Q' are then applied.

4 Implementation

The practical implementation of Rapid follows the above strategy, however the unfolding step includes additional optimizations that reduce the number of redundant rewritings and hence the need for extended subsumption checks. The Rapid system (algorithm 3) uses procedures Shrink and Unfold (algorithms 1 and 2). Shrink computes the closure $\mathcal{S}^*(Q)$ by iteratively applying the shrinking inference rule. Each rewriting computed by Shrink is processed by Unfold, which computes two disjoint sets of rewritings. We will now discuss their contents, explaining at the same time the optimizations that have been employed.

If we apply exhaustively the $\mathcal{W}$ rule on a CQ Q in order to compute $\mathcal{W}^*(Q)$, we may end up with many redundant rewritings. Given that these are undesired, we have two options: either to compute all rewritings and then remove the redundant ones, or else try to apply $\mathcal{W}$ in a cleverer way, so as to get only non redundant rewritings, or at least as few as possible. Because the subsumption check operation is very costly, we choose the second option, i.e. we have to solve the following problem: Given a CQ Q of the form $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, compute only the non redundant CQs that are conclusions of all possible applications of $\mathcal{W}$ on Q. For convenience, define $\mathcal{B}_i = \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B})$, so that we have the sequence of the possibly non disjoint unfolding sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$. For simplicity, we can drop the substitutions γ_i that appear in the definition of $\mathcal{W}$ by assuming that if some member of a set $\mathcal{B}_j$ has been obtained by an inference that introduced a new variable, that variable appears in no other element of $\bigcup_{i=1}^{n} \mathcal{B}_i$. If the sets $\mathbf{B}_i$ are not disjoint, simply taking all possible combinations of their elements in order to form the unfoldings of Q, will certainly result in several redundant rewritings, which we should then check for subsumption.

Algorithm 1 The query shrinking procedure.

procedure SHRINK(CQ Q, ontology $\mathcal{O}$)
  $\mathcal{Q}_{res} \leftarrow \emptyset$; $\mathcal{Q}_r \leftarrow \{Q\}$
  for all $Q' \in \mathcal{Q}_r$ **do**
    $\mathcal{Q}_{res} \leftarrow \mathcal{Q}_{res} \cup \{Q'\}$; $\mathcal{Q}_r \leftarrow \mathcal{Q}_r \setminus \{Q'\}$
    for all $v \in \mathsf{vars}^{\mathsf{B}}\, Q' \setminus \mathsf{vars}^{\mathsf{D}}\, Q'$ **do**
      $\mathcal{F} \leftarrow \mathsf{funcs}\, \Xi(\mathcal{O})$; $\mathcal{X} \leftarrow \emptyset$; $\mathcal{C} \leftarrow \emptyset$; $\mathcal{A} \leftarrow \emptyset$
      for all $\mathbf{B} \in \mathsf{body}\, Q'$ **do**
        if $v \in \mathsf{vars}\, \mathbf{B}$ **then**
          $\mathcal{F} \leftarrow \mathcal{F} \cap \mathcal{F}_v(\mathbf{B}; \mathsf{terms}^{\mathsf{B}}\, Q' \cap \mathsf{terms}\, \mathbf{B})$
          $\mathcal{X} \leftarrow \mathcal{X} \cup (\mathsf{vars}^{\mathsf{B}}\, Q' \cap \mathsf{vars}\, \mathbf{B})$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathsf{cons}\, \mathbf{B}$
        else
          $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{B}\}$
        end if
      end for
      if $|\mathcal{C}| > 1$ **then**
        continue
      else if $|\mathcal{C}| = \{a\}$ **then**
        $t \leftarrow a$
      else
        $t \leftarrow$ a new variable not in $\mathsf{vars}\, Q'$
      end if
      $\theta \leftarrow \bigcup_{x \in \mathcal{X}} \{x/t\}$
      $\mathcal{Q}_r \leftarrow \hat{\bigcup}_{f \in \mathcal{F}} \{\mathsf{min}\,(\mathsf{head}\, Q'\theta \leftarrow (\mathsf{cf}\, f)(t), \{\mathbf{B}\theta\}_{\mathbf{B} \in \mathcal{A}})\}$
    end for
  end for
  return $\mathcal{Q}_{res}$
end procedure

For any $\mathbf{B} \in \bigcup_{i=1}^{n} \mathcal{B}_i$, define the set $\mathsf{idx}\, \mathbf{B} = \{j \mid \mathbf{B} \in \mathcal{B}_j\}$ of the indices of all the unfolding sets that contain $\mathbf{B}$. We call the set $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_k\}$ with $k \leq n$ a *selection* for Q iff (a) $\bigcup_{i=1}^{k} \mathsf{idx}\, \mathbf{A}_i = \mathbb{N}_n$, and (b) $\mathsf{idx}\, \mathbf{A}_i \setminus \mathsf{idx}\, \mathbf{A}_j \neq \emptyset$ for all $i, j \in \mathbb{N}_k$, i.e. if $\mathcal{A}$ contains at least one atom from each unfolding set and we can consider that each unfolding set is represented in $\mathcal{A}$ by a single atom. Clearly, a selection corresponds to an unfolding of Q, in particular to $\mathsf{head}\, Q \leftarrow \mathbf{A}_1, \ldots, \mathbf{A}_k$. However, we are interested in *minimal selections*, which correspond to non redundant rewritings. We call a selection for Q minimal, iff there is no selection $\mathcal{A}'$ for Q such that $\mathcal{A}' \subset \mathcal{A}$, i.e. if in addition to the above conditions, we have that $\mathsf{idx}\, \mathbf{A}_i \setminus \left(\bigcup_{i=1, j \neq i}^{k} \mathsf{idx}\, \mathbf{A}_j\right) \neq \emptyset$ for all $i \in \mathbb{N}_k$, i.e. if all atoms $\mathbf{A}_i$s need to be present in $\mathcal{A}$ in order for $\bigcup_{i=1}^{k} \mathsf{idx}\, \mathbf{A}_i = \mathbb{N}_n$ to hold. If this not was the case, we could form the selection $\mathcal{A}' = \{\mathbf{A}_1, \ldots, \mathbf{A}_{i-1}, \mathbf{A}_{i+1}, \mathbf{A}_k\} \subset \mathcal{A}$, hence $\mathcal{A}$ would not be minimal.

The unfolding step in Rapid computes efficiently the minimal selections for a CQ Q by examining the contents of the sets $\mathsf{idx}\, \mathbf{B}_i$ for each candidate raw unfolding given by the $\mathcal{W}$ rule. However, although the set of minimal selections

Algorithm 2 The query unfolding procedure

procedure UNFOLD(CQ Q of the form $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, ontology $\mathcal{O}$)
  $\mathcal{Q} \leftarrow \emptyset$; $\hat{\mathcal{Q}} \leftarrow \emptyset$
  for $i = 1 \ldots n$ **do**
    $\mathcal{B}_i \leftarrow \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B}_i)$; $\hat{\mathcal{B}}_i \leftarrow \emptyset$
  end for
  for $i = 1 \ldots n$ **and for all** role atoms $\mathbf{A} \in \hat{\mathcal{B}}_i$ **do**
    for $j = 1 \ldots n, j \neq i$ **and for all** role atoms $\mathbf{A}' \in \hat{\mathcal{B}}_j$ **do**
      if $\exists \theta$ on $\mathsf{vars}\, \mathbf{A}' \setminus \mathsf{vars}^{\mathsf{B}}\, Q$ such that $\mathbf{A}'\theta = \mathbf{A}$ **then**
        $\hat{\mathcal{B}}_j \leftarrow \hat{\mathcal{B}}_j \cup \{\mathbf{A}\}$
      end if
    end for
  end for
  for all selections $\mathbf{C}_1, \ldots, \mathbf{C}_k$ from $\mathcal{B}_1 \cup \hat{\mathcal{B}}_1, \ldots, \mathcal{B}_n \cup \hat{\mathcal{B}}_n$ **do**
    if $\mathsf{idx}\, \mathbf{C}_i \setminus \bigcup_{j=1\ldots k, j\neq i} \mathsf{idx}\, \mathbf{C}_j \neq \emptyset$ **for all** i, j **then**
      if $\mathbf{C}_i \in \mathcal{B}_i$ **for all** i **then**
        $\mathcal{Q} \leftarrow \mathcal{Q} \hat{\cup} \{Q\}$
      else
        $\hat{\mathcal{Q}} \leftarrow \hat{\mathcal{Q}} \hat{\cup} \{Q\}$
      end if
    end if
  end for
  return $[\mathcal{Q}, \hat{\mathcal{Q}}]$
end procedure

for some CQ Q corresponds to the set of all its non redundant unfoldings and hence there is no need to perform subsumption checks within this set, the union of the sets of the non redundant unfoldings of two distinct CQs Q_1 and Q_2 (e.g. obtained at the shrinking step) may not be redundant free. The need for subsumption check remains, however the number of required checks is much less, since the unfoldings of Q_1 have to be checked for subsumption only against the unfoldings of Q_2 and vice versa, and not also against the unfoldings of Q_1.

The above procedure computes sets of non redundant, raw unfoldings, but does not take into account possible redundant atoms that may appear in a raw unfolding Q' of Q and are removed in $\mathsf{min}\, Q'$. The removal of possible redundant atoms, so as to get a candidate core rewriting, may change the structure of Q' and make it subsume several other raw unfoldings given by the minimal selections for Q. Because the redundant atoms removal process may involve the binding of unbound variables to bound ones, this issue can be addressed by computing all such bindings in advance and include them in the sets $\mathcal{B}_i$. In particular, if for some $i, j \in \mathbb{N}_n$ we have that $\mathbf{A} \in \mathcal{B}_i$, $\mathbf{A}' \in \mathcal{B}_j$ and there is a substitution θ on $\mathsf{vars}\, \mathbf{A}' \setminus \mathsf{vars}^{\mathsf{B}}\, Q$ such that $\mathbf{A}'\theta = \mathbf{A}$, we add $\mathbf{A}$ to $\mathcal{B}_j$. We call the minimal selections for Q that contain at least such an atom *impure*. Their inclusion in the result does not affect soundness, since we have only replaced an unbound variable of $\mathbf{A}'$ by a bound variable of $\mathbf{A}$. However, an impure selection may result in a minimal unfolding that is subsumed by an unfolding given by another minimal

Algorithm 3 The Rapid algorithm

procedure RAPID(CQ Q, ontology $\mathcal{O}$)
  $\mathcal{Q}_f = \emptyset$
  for all $Q_s \in$ SHRINK(Q) **do**
    $[\mathcal{Q}, \hat{\mathcal{Q}}] \leftarrow$ UNFOLD(Q_s); $\mathcal{Q}_t \leftarrow \emptyset$
    for all $Q' \in \mathcal{Q}_p$ **do**
      if $\mathsf{min}\, Q' = Q'$ **then**
        $Q_t \leftarrow Q_t \cup \{Q'\}$
      else
        $Q_f \leftarrow Q_f \cup \{\{\mathsf{min}\, Q'\}\}$
      end if
    end for
    $Q_f \leftarrow Q_f \cup Q_t \cup \bigcup_{Q' \in \hat{\mathcal{Q}}} \{\{\mathsf{min}\, Q'\}\}$
  end for
  return REMOVEREDUNDANT(Q_f)
end procedure

selection for Q. For this reason, UNFOLD distinguishes between the two sets of unfoldings and returns them in the sets $\mathcal{Q}$ and $\hat{\mathcal{Q}}$, which contain the unfoldings that have resulted from the pure and impure minimal selections, respectively.

The final step of Rapid is the check for redundancies in the results of UNFOLD. The check is done after first grouping the results into sets that are known not to contain redundant rewritings. These are the sets of the pure unfoldings returned by UNFOLD, excluding the unfoldings who did not coincide with their minimal form. The minimal form of each of these queries, as well as each impure unfolding forms a separate set. These sets are processed by REMOVEREDUNDANT which removes any redundant rewritings by checking for subsumption across the several sets only. We should also note that UNFOLD applies the combined unfolding rule $\mathcal{W}$ only on its input Q and not iteratively also on the conclusions. This does not affect completeness, because at the application of $\mathcal{W}$ bound terms may be converted to unbound or eliminated only due to removal of redundant atoms. This redundancy removal is the last step of the application of the $\mathcal{W}$ rule, hence no rewriting is lost. However, it may be the case that an unbound variable v of such a conclusion (which was bound in the main premise) can unify with a functional term of some clause in $\Xi(\mathcal{O})$ and hence be eventually eliminated. If this is the case, variable v would have been eliminated also earlier at some application of the shrinking rule, hence again completeness is not affected.

Example 2. Consider the CQ $Q(x) \leftarrow A(x), R(x,y), A(y), S(x,z)$ posed over the ontology of example 1. We have $\mathsf{vars}^{\mathsf{D}}\, Q = \{x\}$, $\mathsf{vars}^{\mathsf{B}}\, Q = \{x,y\}$ and $\mathsf{vars}^{\mathsf{UB}}\, Q = \{z\}$. From table 3 we get that $\mathcal{F}_y(R(x,y);\{x,y\}) \cap \mathcal{F}_y(A(y);\{y\}) = \{f_1\}$ and given that $\mathsf{cn}\, f_1 = C$, the SHRINK procedure returns the rewritings $Q_1 : Q(x) \leftarrow A(x), R(x,y), A(y), S(x,z)$ (the initial CQ) and $Q_2 : Q(x) \leftarrow A(x), C(x), S(x,z)$, which are subsequently processed by the UNFOLD procedure. We have that $\mathsf{vars}^{\mathsf{B}}\, Q_1 = \{x,y\}$ and $\mathsf{vars}^{\mathsf{B}}\, Q_2 = \{x\}$. The sets $\mathcal{B}_i$ and $\hat{\mathcal{B}}_i$ for Q_1 and Q_2 are shown below (for convenience unbound variables have been replaced by $*$).

i	1	2	3	4
$\mathcal{B}_i$	$A(x)$ $B(x)$ $R(x,*)$ $S(*,x)$ $C(x)$ $T(*,x)$	$R(x,y)^{\{1,2\}}$ $S(y,x)^{\{1,2\}}$	$A(y)$ $B(y)$ $R(y,*)$ $S(*,y)$ $C(y)$ $T(*,y)$	$S(x,*)$ $D(x)$
$\hat{\mathcal{B}}_i$	$R(x,y)^{\{1,2\}}$ $S(y,x)^{\{1,2\}}$			

i	1	2	3
$\mathcal{B}_i$	$A(x)$ $B(x)$ $R(x,*)$ $S(*,x)$ $C(x)^{\{1,2\}}$ $T(*,x)^{\{1,2\}}$	$C(x)^{\{1,2\}}$ $T(*,x)^{\{1,2\}}$	$S(x,*)$ $D(x)$
$\hat{\mathcal{B}}_i$			

For the atoms $\mathbf{A}$ for which $|\mathsf{idx}\,\mathbf{A}| > 1$ the above tables show the sets $\mathsf{idx}\,\mathbf{A}$ (in supersript). Because in both Q_1 and Q_2 for all atoms $\mathbf{A}$ in column 2 we have $\mathsf{idx}\,\mathbf{A} = \{1,2\}$, Rapid computes no unfoldings with atoms that appear only in column 1, because they would be redundant (e.g. for Q_2 the CQ $Q(x) \leftarrow A(x), C(x), S(x,z)$ is subsumed by $Q(x) \leftarrow C(x), S(x,z)$. In this way $24\,(= 2*6*2)$ rewritings are obtained from Q_1 and $4\,(= 2*2)$ from Q_2. The unfoldings of Q_1 are all impure (because they contain atoms in $\hat{\mathcal{B}}_1$) while the ones of Q_2 are all pure. All of them are finally checked for subsumptions, however the check within the set of the rewritings of Q_2 is skipped because we know that it contains no redundant rewritings. Finally, we get 28 core rewritings.

5 Evaluation

We evaluated Rapid by comparing it with Requiem (its greedy strategy), the implementation of RQR. We used the same data sets as in [4], namely the V, S, U, A, P5, UX, AX, P5X ontologies. (V models European history, S European financial institutions, and A information about abilities, disabilities and devices. U is a DL-Lite$_R$ version of the LUBM benchmark ontology. P5 is synthetic and models graphs with paths of length 5. UX, AX and P5X were obtained from U, A and P5, by rewriting them without qualified existential restrictions.) Rapid was implemented in Java, as is Requiem. Tests were performed on a 3GHz processor PC and the results are shown in table 4. For both Rapid and Requiem two times (in *hh:mm:ss.msec* format) and rewriting sizes are given. T_A is the rewriting computation time without the final redundancy removal step, and T_R is the total time including this last step. Similarly, R_A is the size of the rewriting set when omitting the redundancy removal step, and R_F the size of the core rewriting set. As expected, both systems compute the same number of core rewritings.

The results show clearly the efficiency of Rapid. It is always faster and in several cases the improvement is very significant. The efficiency is even more evident if we compare the results before and after the redundancy removal step. In most cases, the number of rewritings discarded as redundant in this step is small. Hence, by omitting it we would still get a 'good' result, but gain possibly a lot in time. The redundancy removal step is very expensive, although our optimizations significantly reduced its cost too. The most striking case is ontology AX and query 5, in which Rapid completes the computation of the 32,921 core

rewritings in about 43 seconds, while Requiem needs about 2 hours. Moreover, Rapid needs only to 2.1 sec to compute a set containing only 35 redundant rewritings and then about 41.2 sec to detect them, while Requiem computes 43,111 redundant rewritings and needs 1.5 hours to detect and remove them.

		Rapid				Requiem (greedy strategy)			
$\mathcal{O}$	Q	T_A	T_F	R_A	R_F	T_A	T_F	R_A	R_F
V	1	.001	.001	15	15	.001	.001	15	15
	2	.001	.001	10	10	.001	.001	10	10
	3	.001	.001	72	72	.016	.016	72	72
	4	.015	.015	185	185	.031	.062	185	185
	5	.016	.016	30	30	.001	.015	30	30
S	1	.001	.001	6	6	.001	.001	6	6
	2	.001	.001	2	2	.031	.062	160	2
	3	.001	.001	4	4	.187	.515	480	4
	4	.001	.001	4	4	.406	1.047	960	4
	5	.001	.001	8	8	5.594	17.984	2,880	8
U	1	.001	.001	2	2	.001	.001	2	2
	2	.001	.001	1	1	.031	.047	148	1
	3	.001	.001	4	4	.047	.109	224	4
	4	.001	.001	2	2	.625	2.031	1,628	2
	5	.001	.001	10	10	2.187	7.781	2,960	10
A	1	.001	.001	27	27	.031	.047	121	27
	2	.001	.001	54	50	.031	.047	78	50
	3	.016	.016	104	104	.047	.063	104	104
	4	.031	.031	320	224	.078	.156	304	224
	5	.062	.078	624	624	.188	.610	624	624
P5	1	.001	.001	6	6	.001	.001	6	6
	2	.001	.001	10	10	.015	.015	10	10
	3	.001	.001	13	13	.047	.047	13	13
	4	.015	.015	15	15	.688	.688	15	15
	5	.015	.015	16	16	16.453	16.453	16	16
P5X	1	.001	.001	14	14	.001	.001	14	14
	2	.001	.001	25	25	.031	.031	77	25
	3	.015	.031	112	58	.125	.297	390	58
	4	.062	.109	561	179	2.453	7.375	1,953	179
	5	.344	1.313	2,805	718	1:10.141	3:48.690	9,766	718
UX	1	.001	.001	5	5	.001	.001	5	5
	2	.001	.001	1	1	.031	.078	240	1
	3	.001	.001	12	12	.391	1.125	1,008	12
	4	.001	.001	5	5	5.187	19.375	5,000	5
	5	.015	.015	25	25	15.125	57.672	8,000	25
AX	1	.001	.001	41	41	.047	.063	132	41
	2	.093	.140	1,546	1,431	.703	2.781	1,632	1,431
	3	.297	.672	4,466	4,466	6.484	29.109	4,752	4,466
	4	.219	.625	4,484	3,159	5.282	23.516	4,960	3,159
	5	2.140	43.374	32,956	32,921	27:04.006	1:56:21.585	76,032	32,921

Table 4. Evaluation results.

We comment on two cases that illustrate best the efficiency of the shrinking and unfolding steps in Rapid. In ontology $P5$, where query i asks for nodes from which paths of length i start, the performance of Rapid is essentially unaffected by the query size, unlike Requiem which is not scalable. This is due to the efficiency of the shrinking inference rule, which fires only if it leads to a valid rewriting. In RQR the saturation is performed undiscriminately, leading to a large number of non function free rewritings (exponential in the size of $\mathsf{funcs}\,\Xi(\mathcal{O})$) that are eventually discarded. In ontology U, the superior per-

formance of Rapid is due to the efficiency of the unfolding step, in particular to the non computation of redundant unfoldings. In query 5, at the end of the unfolding step Rapid has computed only 8 rewritings, which are the final core rewritings. In contrast, Requiem computes 2,880, which need to be checked for subsumption. In the general case the superior performance of Rapid is due to the combined efficiency of the shrinking and unfolding steps.

Before concluding this section we should note, however, that Requiem, being an $\mathcal{EL}$ reasoner, is not optimized for DL-Lite$_R$. Nevertheless, in [4] which compares Requiem with CGLLR, an implementation of the authors of the PerfectRef algorithm, Requiem shows already a better performance.

6 Conclusions

We have presented Rapid, a new algorithm for the efficient computation of query rewritings over DL-Lite$_R$ ontologies. The general structure of Rapid is inspired by a resolution-based process, however its steps are optimized and the application of the first order resolution rule is replaced by specialized shrinking and unfolding rules, which save the algorithm from many redundant rewritings, many subsumption checks, as well as from blind resolution paths. The specialized inference rules differentiate Rapid from pure resolution-based algorithms, however it remains committed to the computation of query rewriting sets, unlike recent approaches [6] which circumvent the exponential complexity of the query rewriting problem by computing datalog programs, deferring thus the complexity to the underlying database systems. The experimental evaluation of Rapid showed a significant performance benefit if compared to RQR, which in several practical cases can alleviate the exponential behavior of the system. An interesting direction for future work is to apply the idea to more expressive DLs, like $\mathcal{ELHI}$.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. J. of Artificial Intelligence Research, 36–69, (2009).
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite Family. J. of Automated Reasoning, (2007).
3. B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic SHIQ. J. of Artificial Intelligence Research, 31:157–204, (2008).
4. H. Perez-Urbina, I. Horrocks, and B. Motik. Efficient query answering for OWL 2. In Procs of ISWC 2009, LNCS 5823:489–504, (2009).
5. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. J. on Data Semantics, 133–173, (2008).
6. R. Rosati and A. Almatelli, Improving Query Answering over DL-Lite Ontologies. In Procs of KR 2010, (2010)
7. J. A. Robinson and A. Voronkov, editors. Handbook of Automated Reasoning (in 2 volumes), Elsevier and MIT Press, (2001).
8. M. Stocker, and M. Smith. Owlgres: A scalable OWL reasoner. In Procs of OWLED 2008, (2008).