Optimized Query Rewriting for 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 rewriting set of a user query, by avoiding blind and unnecessary inferences, as well as by reducing the need for extended subsumption checks. The evaluation shows that in several cases the algorithm achieves a significant improvement and better practical 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 some data [4,7]. Queries typically have the form of *conjunctive queries* (CQ) and terminologies the form of ontologies. Unfortunately, the problem of answering CQs in terms of ontologies axiomatized in expressive Description Logics 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 soundness or completeness have to be sacrificed.

Late research in description logics has introduced DL-Lite$_R$, a 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. Sound and complete CQ answering systems for DL-Lite$_R$ can follow a strategy that splits the procedure in two steps [7,1,8]: the query *rewriting*, in which the CQ is expanded into a union of CQs (UCQ), and the *execution* of the UCQ over the database. Apart from having the advantage of using the mature relational database technology, rewriting can be based on first order resolution-based reasoning algorithms [6], which are widely studied in the literature [2]. The main

* The work of this paper has been partially funded by EuropeanaConnect. Best Practice Network in the eContentplus Programme of the European Community. (ECP-2008-DILI-528001) www.europeanaconnect.eu.

restriction is that for large terminologies and/or large queries the exponential complexity in the query size may result in a very large number of rewritings.

Several CQ answering algorithms for DL-Lite$_R$ have been proposed in the literature. In [3,9], the rewriting strategy is based on reformulating the conjuncts of the query according to the taxonomic information of the ontology. Although the strategy is effective, some of the ontology axioms must be rewritten in terms of auxiliary roles, which may increase the ontology size. This restriction is relaxed in [6], which proposes a resolution-based rewriting strategy, called RQR. However, the non goal-oriented saturation strategy may get tangled in long inference paths leading either to unnecessary or non function free rewritings. Such rewritings are discarded in the end, but their participation in the inference process and the increased number of required subsumption checks degrades significantly performance. Another strategy is proposed in [8] which, instead of computing a set of CQs, builds a non-recursive datalog program, deferring thus the main source of complexity to the database system. A different approach is used in [5], which partially materializes the data in order to facilitate the rewriting process.

In this paper we improve on the pure query rewriting approach and introduce a new query rewriting algorithm called Rapid, which is optimized for queries posed over DL-Lite$_R$ ontologies. Its efficiency is owed to the goal-oriented organization of the resolution process. Instead of applying exhaustively the resolution rule, it exploits the query structure and performs a restricted sequence of inferences that lead directly to rewriting sets with, hopefully, no unnecessary rewritings. 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 unnecessary rewritings (that are subsumed by others) and the need to remove them by performing extended query subsumption checks, i.e. very costly operations. For simplicity, we restrict our study to user queries in which all body variables are reachable from a head variable through a role sequence. Although this assumption excludes some queries, e.g. 'boolean queries', it has little impact in practice, since such queries are not common in a typical semantic query answering system.

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* and $\mathcal{A}$ the assertional knowledge. 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 a conjunction), where atom $\mathbf{A}$ is the *head* and atoms $\mathbf{B}_i$ the *body* of Q. We assume that $\mathbf{B}_i$s 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 $\mathbf{B} \in \mathsf{body}\, Q$ are entities of $\mathcal{T}$ and have arities 1 or 2, if the entity is a concept or a role, respectively. Hence, $\mathbf{B}$ is a *concept atom* $B(t)$ or a *role atom* $B(t,s)$. $\mathsf{terms}\, \mathbf{B}$ ($\mathsf{vars}\, \mathbf{B}$, $\mathsf{cons}\, \mathbf{B}$) are the sets of terms (variables, constants) that appear in $\mathbf{B}$. For a set of atoms $\mathcal{B}$ we have that $\mathsf{terms}\, \mathcal{B} = \bigcup_{\mathbf{B} \in \mathcal{B}} \mathsf{terms}\, \mathbf{B}$, 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.

A term $t \in \mathsf{terms}\, Q$, where Q is a function free CQ is called *distinguished* if it appears in $\mathsf{head}\, Q$, and *non distinguished* otherwise; *bound* if it is either a constant, or a distinguished variable, or a variable that appears at least twice in $\mathsf{body}\, Q$, and *unbound* otherwise; and *disconnected* if there is a disconnected subgraph (V', E') of the graph $(\mathsf{terms}\, Q,\ \{\{t,s\} \mid R(t,s) \text{ or } R(s,t) \in \mathsf{body}\, Q\})$ such that $t \in V'$ and set V' contains no distinguished term. We denote the set of bound terms, and distinguished, bound and unbound variables of Q by $\mathsf{terms}^{\mathsf{B}}\, Q$, $\mathsf{vars}^{\mathsf{D}}\, Q$, $\mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}^{\mathsf{UB}}\, Q$, respectively. As noted in the introduction, we will assume that the user query Q is *connected*, i.e. it contains no disconnected terms. For simplicity and wlog we can also assume that Q contains 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 CQ Q posed over the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ iff $\Xi(\mathcal{O}) \cup \{Q\} \models C(\boldsymbol{a})$, where C is the predicate of $\mathsf{head}\, Q$ and $\Xi(\mathcal{O})$ the standard clausification of $\mathcal{O}$ into first order clauses. Each axiom of $\mathcal{O}$ adds either one or two clauses as shown in Table 1, and each axiom that contains an existential quantifier introduces a distinct function. The set that contains all answers of Q over $\mathcal{O}$ is denoted by $\mathsf{cert}\, (Q, \mathcal{O})$. It has been proved [7,1] 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 saturating Q and $\Xi(\mathcal{O})$ using first order resolution. We denote derivability under the first order resolution rule by $\vdash_{\mathcal{R}}$.

Table 1. Translation of DL-Lite$_R$ axioms into clauses of $\Xi(\mathcal{O})$ (reproduced from [6])

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

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.* If $\mathcal{Q}$ is a set of CQs and for some CQ Q there is a $Q' \in \mathcal{Q}$ equivalent to Q, we write $Q \mathrel{\hat{\in}} \mathcal{Q}$. We define also the operation $\mathcal{Q} \mathbin{\hat{\cup}} \{Q\} = \mathcal{Q} \cup \{Q\}$ if $Q \mathrel{\hat{\notin}} \mathcal{Q}$, and $\mathcal{Q} \mathbin{\hat{\cup}} \{Q\} = \mathcal{Q}$ otherwise. A set $\mathsf{rewr}\,(Q, \mathcal{O})$ is a *rewriting set* of the CQ Q over $\mathcal{O}$ iff for each rewriting Q' of Q over $\mathcal{O}$, either $Q' \mathrel{\hat{\in}} \mathsf{rewr}\,(Q, \mathcal{O})$ or there is a $Q'' \in \mathsf{rewr}\,(Q, \mathcal{O})$ such that $Q'' \rhd Q'$. 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 a minimal subset of $\mathsf{body}\,Q$ such that $Q \rhd Q'$, Q' is called *condensed* or a *condensation* of Q, and is denoted by $\mathsf{cond}\,Q$. Since a CQ is equivalent to its condensation, we can find $\mathsf{cert}\,(Q, \mathcal{O})$ by computing a rewriting set of Q that contains only condensed rewritings and that contains no two rewritings Q, Q' such that $Q \rhd Q'$. Hence, we say that Q' is a *core rewriting* of a CQ Q over $\mathcal{O}$, iff it is a rewriting of Q over O, it is condensed, and there is no (non equivalent) rewriting Q'' of Q over $\mathcal{O}$ such that $Q'' \rhd Q'$. The *core rewriting set* $\mathsf{rewr}^{\mathsf{C}}\,(Q, \mathcal{O})$ of Q over $\mathcal{O}$ is the set of all the core rewritings of Q over $\mathcal{O}$.

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 some tasks differently in order to reduce the inferences that lead to rewritings that will eventually be discarded because they are not function free or subsumed by others. The strategy of Rapid is based on the distinguishing property of the bound 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 Q apart from $\mathbf{A}$. This is not the case if $v \in \mathsf{vars}^{\mathsf{UB}}\,Q$, since unbound variables appear only once in Q. The main premise in the resolution rules in Rapid is always the user query 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 in resolution rule applications 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, some bound variables of the main premise may also be eliminated, not through the introduction and subsequent elimination of functional terms, but while condensing the conclusion. Obviously, the same can also happen at the shrinking step. (4) The *subsumption check* step, in which non core rewritings are removed. This step is in principle the same as in RQR, but is more efficient in two ways: First, the previous steps produce much fewer rewritings that are subsumed by others, and second not every pair of rewritings has to be checked for subsumption, because, as we will see, some sets of rewritings that are produced at the unfolding step are guaranteed not to contain rewritings that are subsumed by others.

Notwithstanding this general description, Rapid does not implement the shrinking and unfolding steps by applying directly the resolution rule. Instead, a shrinking and unfolding inference rule are defined, which combine a series of several successful resolution rule application steps into one. In this way, the resolution rule is used only if it eventually leads to a function free and hopefully also a core rewriting, and a large number of unnecessary inferences is avoided.

3.1 Atom Unfolding Sets

The closure of $\Xi(\mathcal{O})$ under the FOL resolution rule contains clauses of the form

$$\begin{array}{lll}
A(x) \leftarrow B(x), & A(x) \leftarrow B(x,y), & A(x,y) \leftarrow B(x,y), \\
A(x,f(x)) \leftarrow B(x), & & A(x,f(x)) \leftarrow B(x,y), \\
A(g(x),f(g(x))) \leftarrow B(x), & & A(g(x),f(g(x))) \leftarrow B(x,y), \\
A(g(h(x)),f(g(h(x)))) \leftarrow B(x), \ \ldots & & A(g(h(x)),f(g(h(x)))) \leftarrow B(x,y), \ \ldots
\end{array}$$

as well as the respective clauses with the role atom arguments inverted. We note that in the clauses of the first two rows, the non functional terms of the head appear also in the body. Based on this remark, and given that in the unfolding step we want that the 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* $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ *for some substitution* θ *on a subset of* $\mathsf{vars}\,\mathbf{A} \setminus T$ *to functional terms, 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, $\mathbf{B}\theta'$ is an unfolding of $\mathbf{A}$ w.r.t. T if it is the body of a clause inferrable from $\Xi(\mathcal{O})$ that has in its head an atom $\mathbf{A}'$ (of the same predicate as $\mathbf{A}$), and both $\mathbf{B}$ and $\mathbf{A}'$ contain unaltered all terms in T (which should contain the bound terms in $\mathbf{A}$). Since the variable renaming θ' contains no essential information, we define the *unfolding set* of atom $\mathbf{A}$ for T w.r.t. $\Xi(\mathcal{O})$ as the set $\mathcal{D}(\mathbf{A};T) = \{\mathbf{B} \mid \Xi(\mathcal{O}) \cup \{\mathbf{A}\} \vdash_{\mathcal{J}(T)} \mathbf{B}\}$, where $\mathcal{J}(T)$ are the inference rules shown in Fig. 1, in the form $\frac{\mathbf{A}\ C}{\mathbf{B}}$. Given T, $\mathbf{A}$ (the main premise) and a clause $C \in \Xi(\mathcal{O})$ (the side premise), by applying the respective rule we get atom $\mathbf{B}$ (the conclusion). We also define the set $\hat{\mathcal{D}}(\mathbf{A};T) = \mathcal{D}(\mathbf{A};T) \cup \{\mathbf{A}\}$. By using Table 2, which lists all possible cases, it is easy to prove that given $\mathbf{A}$ and $T \neq \emptyset$ we have that $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ iff $\mathbf{B}\theta' \in \mathcal{D}(\mathbf{A};T)$, for θ, θ' as defined in Def. 1.

3.2 Atom Function Sets

As we have already seen, the closure 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

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

Table 2. All possible cases for **A**, **B**, T, θ and θ' in Def. 1

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

the form $A(g(x), f(g(x))) \leftarrow B(x)$ and $A(g(x), f(g(x))) \leftarrow B(x,y)$. Unlike in the unfolding case, now we are interested in the behavior of the functional term $f(x)$, which appears in the head but not in the body, because if $f(x)$ appears in the body of some rewriting, it may be possible to eliminate it by using such clauses. Let $\mathsf{funcs}\, \Xi(\mathcal{O})$ be the set of all functions in $\Xi(\mathcal{O})$. According to Table 1, each DL-Lite$_R$ axiom that has an existential quantifier in the RHS introduces a distinct function f. Hence, 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. Let $\mathsf{cn}\, f$ denote the concept associated with f. We define the set of all functions that may appear in the place of a bound variable v of an atom $\mathbf{A}$ when resolving any of its unfoldings with a non function free clause in $\Xi(\mathcal{O})$ as follows:

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:*

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

It follows that, given a $T \neq \emptyset$ which represents the set of bound terms in $\mathbf{A}$, (a) if $\mathbf{A} \equiv A(v,t)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} A(f(t),s) \leftarrow (\mathsf{cn}\, f)(t)$, (b) if $\mathbf{A} \equiv A(t, v)$ then $f \in \mathcal{F}_v(\mathbf{A}; T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} A(s, f(t)) \leftarrow (\mathsf{cn}\, f)(t)$, 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 $\mathbf{A} \equiv A(v)$ then $f \in \mathcal{F}_v(\mathbf{A}; T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} A(f(t)) \leftarrow (\mathsf{cn}\, f)(t)$.

Example 1. Define the ontology $\mathcal{O} = \{B \sqsubseteq A, \exists R \sqsubseteq A, S \sqsubseteq R^-, C \sqsubseteq \exists R.A, \exists T^- \sqsubseteq C, D \sqsubseteq \exists S\}$, hence $\Xi(\mathcal{O}) = \{A(x) \leftarrow B(x), A(x) \leftarrow R(x, y), R(x, y) \leftarrow S(y, x), R(x, f_1(x)) \leftarrow C(x), A(f_1(x)) \leftarrow C(x), C(x) \leftarrow T(y, x), S(x, f_2(x)) \leftarrow D(x)\}$. Below we show the unfolding and function sets for the atoms $A(x)$, $C(x)$, $R(x, y)$ and $S(x, y)$ and some sets T. E.g. for $T = \{y\}$, main premise $R(x, y)$ and side premise $R(x, y) \leftarrow S(y, x)$, from Fig. 1 we get $S(y, x)$. Then (given that $x \notin T$), for main premise $S(y, x)$ and side premise $S(x, f_2(x)) \leftarrow D(x)$ we get $D(y)$. Because $R(x, f_1(x)) \leftarrow C(x) \in \Xi(\mathcal{O})$, we get that $\mathcal{F}_y(R(x, y); \{y\}) = \{f_1\}$.

$\mathbf{A}; T$	$A(x); \{x\}$	$C(x); \{x\}$	$R(x, y); \{x\}$	$R(x, y); \{y\}$	$R(x, y); \{x, y\}$	$S(x, y); \{x\}$
$\mathcal{D}(\mathbf{A}; T)$	$B(x)$ $R(x, z_1)$ $S(z_1, x)$ $C(x)$ $T(z_2, x)$	$T(z_3, x)$	$S(y, x)$ $C(x)$ $T(z_4, x)$	$S(y, x)$ $D(y)$	$S(y, x)$	$D(x)$
$\mathcal{F}_x(\mathbf{A}; T)$	$\{f_1, f_2\}$	$\emptyset$	$\{f_2\}$	$\emptyset$	$\{f_2\}$	$\emptyset$
$\mathcal{F}_y(\mathbf{A}; T)$	–	–	$\emptyset$	$\{f_1\}$	$\{f_1\}$	–

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 during the standard resolution-based inference process, subsequently it must be eliminated. However, we know that each function appears 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. Moreover, if Q contains the terms say $R(x, v)$ and $R(v, y)$ and v is eliminated this way by unifying with $f(x)$, given the form of $\Xi(O)$, variables x and y must be unified. If in place of x, y there are constants, these should coincide in order for the inference to be possible. 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 is as follows:*

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

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. Moreover, all variables in $\mathcal{X}$ are also merged into one. It is easy to prove that $\mathcal{S}$ is a sound inference rule, i.e. if $\Xi(\mathcal{O}) \cup \{Q\} \vdash_{\mathcal{S}} Q'$ then $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$, for any CQ Q'.

3.4 Query Unfolding

Let $\mathcal{S}^*(Q)$ be the closure of $\mathsf{cond}\, Q$ under application of the inference rule $\mathcal{S}$, for any CQ Q. By construction, $\mathcal{S}^*(Q)$ 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. 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 by computing, for each $Q' \in \mathcal{S}^*(Q)$, all CQs that can be inferred from Q' by replacing one or more of its atoms by one of their 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 *is defined as follows:*

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

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. if $\Xi(\mathcal{O}) \cup \{Q\} \vdash_{\mathcal{U}} Q'$ then $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$. Rule $\mathcal{U}$ replaces one atom of Q by one of its unfoldings, and can be applied iteratively on the conclusion in order to produce more rewritings. In order to facilitate the optimization of such a sequential application of the $\mathcal{U}$ rule on some rewriting, we define the *combined unfolding 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* combined unfolding rule $\mathcal{W}$*:*

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

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* $x \in \mathsf{vars}\, \mathbf{C}_i \setminus \mathsf{terms}^{\mathsf{B}}\, Q$*.*

Let $\mathcal{W}^*(Q)$ be the closure of $\mathsf{cond}\, Q$ under application of the inference rule $\mathcal{W}$, for any CQ Q. The strategy by which Rapid computes the core rewriting set of a user query Q is justified by the following theorem:

Theorem 1. *Let Q be a connected 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 the final conclusion of a sequence of resolutions with main premises Q, Q_1, …, Q_{l-1}, we must show that there is a sequence of shrinking rule applications with main premises Q, Q_1^s, …, $Q_{l_s-1}^s$ and final conclusion $Q_{l_s}^s$, and a sequence of unfolding rule applications with main premises $Q_{l_s}^s$, Q_1^u, …, $Q_{l_u-1}^u$ and final conclusion Q'. $Q_1, \ldots, Q_{l-1}$ may contain functional terms of the form $f_1(\cdots f_k(t))$ for $k > 0$ (k is called depth*). Since Q and Q' are both function free, any functional terms eventually are eliminated. The result can be proved by induction on the maximum depth d of the intermediate CQs $Q_1, \ldots, Q_{l-2}$, by showing that the sequence of resolutions that led to the introduction of a functional term of depth d can be rearranged so that only functional terms of depth 1 are introduced. Because the shrinking rule considers by definition all functional terms that may be introduced at any step of the resolution process, it can be applied first, thus giving rise to the shrinking rule application sequence, on whose final conclusion the unfolding rule is then applied, in order to get Q'.*

Note that if we wanted to lift the restriction to connected queries, we should take into account atoms containing only unbound variables. Such a variable may unify with a functional term and give its place to a new unbound variable. Hence, we should allow empty sets T in Def. 1 and include the appropriate rules in $\mathcal{J}(T)$, e.g. rule $\frac{A(t)\ A(f(x)) \leftarrow B(x)}{B(z)}$ for $T = \emptyset$, which 'replaces' variable t by z.

4 Implementation

The implementation of Rapid includes additional optimizations at the unfolding step that reduce the number of non core rewritings that are produced and hence the need for extended subsumption checks. Rapid (Algorithm 3) uses procedures Shrink and Unfold (Algorithms 1 and 2). Shrink computes the closure $\mathcal{S}^*(Q)$ by iteratively applying the shrinking rule. Each rewriting produced by Shrink is processed by Unfold, which computes two disjoint sets of rewritings. We will now discuss their contents and explain the optimizations that have been used.

If we apply exhaustively the $\mathcal{W}$ rule on a CQ Q in order to get $\mathcal{W}^*(Q)$, we may end up with many rewritings subsumed by others. Since this is undesired, we have two options: to compute all rewritings and then remove the subsumed ones, or else try to apply $\mathcal{W}$ in a cleverer way, so as to get only non subsumed 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 $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, find the CQs that are conclusions of all possible applications of $\mathcal{W}$ on Q and are not subsumed by others. For convenience, define $\mathcal{B}_i = \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, Q \cap \mathsf{terms}\, \mathbf{B}_i)$, 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

Algorithm 1 The query shrinking procedure

```
procedure Shrink(CQ $Q$, ontology $\mathcal{O}$)
    $\mathcal{Q}_r \leftarrow \{Q\}$
    for all unconsidered $Q' \in \mathcal{Q}_r$ do
        mark $Q'$ as considered
        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
            $\theta \leftarrow \bigcup_{x \in \mathcal{X}} \{x/t\}$
            $\mathcal{Q}_r \leftarrow \mathcal{Q}_r \hat{\bigcup}_{f \in \mathcal{F}} \{\mathsf{cond}\, (\mathsf{head}\, Q'\theta \leftarrow (\mathsf{cf}\, f)(t), \{\mathbf{B}\theta\ \}_{\mathbf{B} \in \mathcal{A}})\}$
        end for
    end for
    return $\mathcal{Q}_r$
end procedure
```

the substitutions γ_i that appear in the definition of $\mathcal{W}$ by assuming that if a member of a set $\mathcal{B}_j$ has been obtained by an inference that introduced a new variable, this variable does not appear elsewhere in $\bigcup_{i=1}^{n} \mathcal{B}_i$. If the sets $\mathcal{B}_i$ are not disjoint, simply taking all possible combinations of their elements so as to form the unfoldings of Q, will certainly result in rewritings subsumed by others.

For any $\mathbf{B} \in \bigcup_{i=1}^{n} \mathcal{B}_i$, define the set $\mathsf{ind}\, \mathbf{B} = \{j \mid \mathbf{B} \in \mathcal{B}_j\}$ 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) $\bigcup_{i=1}^{k} \mathsf{ind}\, \mathbf{A}_i = \mathbb{N}_n$ (where $\mathbb{N}_n \doteq \{1, \ldots, n\}$), and (b) $\mathsf{ind}\, \mathbf{A}_i \setminus \mathsf{ind}\, \mathbf{A}_j \neq \emptyset$ for all $i, j \in \mathbb{N}_k$, i.e. if $\mathcal{A}$ contains at least one atom from each unfolding set and no two sets $\mathsf{ind}\, \mathbf{A}_i$ overlap fully. 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 subsumed rewritings. We call a selection $\mathcal{A}$ 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 we have that $\mathsf{ind}\, \mathbf{A}_i \setminus \left(\bigcup_{j=1, j \neq i}^{k} \mathsf{ind}\, \mathbf{A}_j \right) \neq \emptyset$ for all $i \in \mathbb{N}_k$, i.e. if all the atoms $\mathbf{A}_i$ need to be present in $\mathcal{A}$ in order for $\bigcup_{i=1}^{k} \mathsf{ind}\, \mathbf{A}_i = \mathbb{N}_n$ to hold. If this were not the case for some $\mathbf{A}_i$, we could form the selection $\mathcal{A}' = \{\mathbf{A}_1, \ldots, \mathbf{A}_{j-1}, \mathbf{A}_{j+1}, \mathbf{A}_k\} \subset \mathcal{A}$, hence $\mathcal{A}$ would not be minimal.

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 $\mathbf{A} \in \mathcal{B}_i$ do
        for $j = 1 \ldots n, j \neq i$ and for all role atoms $\mathbf{A}' \in \mathcal{B}_j$ 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 $\mathsf{ind}\, \mathbf{C}_i \setminus \bigcup_{j=1 \ldots k, j \neq i} \mathsf{ind}\, \mathbf{C}_j \neq \emptyset$ for all $i$ then
            if $\exists j$ such that $\mathbf{C}_i \in \hat{\mathcal{B}}_j$ for some $i$ then
                $\hat{\mathcal{Q}} \leftarrow \hat{\mathcal{Q}} \hat{\cup} \{Q\}$
            else
                $\mathcal{Q} \leftarrow \mathcal{Q} \hat{\cup} \{Q\}$
            end if
        end if
    end for
    return $[\mathcal{Q}, \hat{\mathcal{Q}}]$
end procedure
```

The unfolding step in Rapid computes efficiently the minimal selections for a CQ Q by finding the common elements of the unfolding sets $\mathcal{B}_i$ and enforcing the above conditions. Although the set of unfoldings obtained by the minimal selections for Q contains no subsumed rewrittings, in general, the same will not hold for the union of the unfoldings of two distinct CQs Q_1 and Q_2 obtained at the shrinking step. The need for subsumption checks remains, however their number is much less, since the unfoldings of Q_1 have to be checked only against the unfoldings of Q_2 and vice versa, and not also against the unfoldings of Q_1.

The computation of the minimal selections as described above takes into account the equality between the elements of the unfolding sets, but not subsumption relations. However, an unfolding set $\mathcal{B}_i$ may contain an atom with an unbound variable that unifies with an atom of another set $\mathcal{B}_j$ that contains only bound variables. In order to address this issue we compute all such bindings in advance and include the respective atoms in the sets $\hat{\mathcal{B}}_i$, defined for this purpose. 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 $\hat{\mathcal{B}}_j$. We call the minimal selections for Q that contain an atom that appears in some $\hat{\mathcal{B}}_j$ *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 an unfolding that subsumes or is subsumed by an unfolding given by another minimal selection for Q. For this reason, UNFOLD

Algorithm 3 The Rapid algorithm

```
procedure Rapid(connected CQ $Q$, ontology $\mathcal{O}$)
    $\mathcal{Q}_f = \emptyset$
    for all $Q_s \in$ Shrink($Q, \mathcal{O}$) do
        $[\mathcal{Q}, \hat{\mathcal{Q}}] \leftarrow$ Unfold($Q_s, \mathcal{O}$); $\mathcal{Q}_t \leftarrow \emptyset$
        for all $Q' \in \mathcal{Q}$ do
            if $\mathsf{cond}\, Q'$ coincides with $Q'$ then
                $Q_t \leftarrow Q_t \cup \{Q'\}$
            else
                $Q_f \leftarrow Q_f \cup \{\{\mathsf{cond}\, Q'\}\}$
            end if
        end for
        $Q_f \leftarrow Q_f \cup \{Q_t\} \cup \bigcup_{Q' \in \hat{\mathcal{Q}}} \{\{\mathsf{cond}\, Q'\}\}$
    end for
    return CheckSubsumption($Q_f$)
end procedure
```

distinguishes between the two sets of unfoldings and returns them in the sets $\mathcal{Q}$ and $\hat{\mathcal{Q}}$, which contain the unfoldings resulting from the pure and impure minimal selections, respectively.

The final step of Rapid is the check for subsumed rewritings within the results of Unfold. The check is done after first grouping the results into sets that are known not to contain subsumed rewritings. These are the sets of pure unfoldings returned by Unfold, excluding the unfoldings that do not coincide with their condensations. The condensation of each such query, as well as each impure unfolding forms a separate set. These sets are processed by CheckSubsumption which checks for subsumption across sets only. We also note that Unfold applies 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 become unbound or eliminated only at the condensation step; this is the last step 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 Q, unifies with a functional term of a clause in $\Xi(\mathcal{O})$ and hence is eventually eliminated. If this is the case, variable v would have been eliminated also earlier at an application of the shrinking rule, hence again completeness is not affected. The algorithm terminates because when computing the unfolding sets or applying the $\mathcal{S}$ rule, atoms or clauses equivalent to already computed ones are not considered again.

Example 2. Consider the CQ $Q(x) \leftarrow A(x), R(x,y), A(y), S(x,z)$ posed over the ontology of Ex. 1. We have $\mathsf{vars}^{\mathsf{D}}\, Q = \{x\}$, $\mathsf{vars}^{\mathsf{B}}\, Q = \{x, y\}$ and $\mathsf{vars}^{\mathsf{UB}}\, Q = \{z\}$. From Ex. 1 we know 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 procedures 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
$\mathcal{B}_i$	$A(x)$ $B(x)$ $R(x,*)$ $S(*,x)$ $C(x)^{\{1,2\}}$ $T(*,x)^{\{1,2\}}$	$C(x)^{\{1,2\}}$ $T(*,x)^{\{1,2\}}$	$S(x,*)$ $D(x)$
$\hat{\mathcal{B}}_i$			

E.g. we add $R(x,y)$ to $\hat{\mathcal{B}}_1$ because $\mathcal{B}_1$ contains $R(x,*)$ which subsumes $R(x,y)$ in $\mathcal{B}_2$. For the atoms $\mathbf{A}$ for which $|\mathsf{ind}\,\mathbf{A}| > 1$ the tables show the sets $\mathsf{ind}\,\mathbf{A}$ in superscript. Since in both Q_1 and Q_2, for all atoms $\mathbf{A}$ in $\mathcal{B}_2$ we have $\mathsf{ind}\,\mathbf{A} = \{1,2\}$, Rapid computes no unfoldings with atoms that appear only in $\mathcal{B}_1$, because they are subsumed (e.g. for Q_2 the CQ $Q(x) \leftarrow A(x), C(x), S(x,z)$ is subsumed by $Q(x) \leftarrow C(x), S(x,z)$). So, we get $24\,(= 2 \cdot 6 \cdot 2)$ rewritings from Q_1 and $4\,(= 2{\cdot}2)$ from Q_2. The unfoldings of Q_1 are all impure (they contain atoms in $\hat{\mathcal{B}}_1$) while those of Q_2 are all pure. All of them are finally checked for subsumption, but the check within the set of the unfoldings of Q_2 is skipped because we know that it contains no subsumed rewritings. Finally, we get 28 core rewritings.

5 Evaluation

We evaluated Rapid by comparing it with Requiem, the implementation of RQR. We used the same datasets as in [6], 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 are obtained by rewriting U, A and P5 without qualified existential restrictions). The results (for a Java implementation on a 3GHz processor PC) are shown in Table 3. T_A is the rewriting computation time without the final subsumption check step, and T_R the total time including this step. Similarly, R_A is the size of the (non core) rewriting set when omitting the subsumption check 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 more evident if we compare the results before and after the subsumption check step. In most cases, the number of rewritings discarded as subsumed in this step is small. Hence, by omitting it we would still get a 'good' result, but gain possibly a lot in time. The subsumption check 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 less than 1 min, while Requiem needs about 2 hours. Moreover, Rapid needs only to 2.1 sec to compute a set containing only 35 non core rewritings and

Table 3. Evaluation results. *The greedy modality provided by the Requiem system applies forward query subsumption, dependency graph pruning and greedy unfolding.

		Rapid				Requiem (greedy modality*)			
$\mathcal{O}$	Q	T_A	T_F	R_A	R_F	T_A	T_F	R_A	R_F
	1	.001	.001	15	15	.001	.001	15	15
	2	.001	.001	10	10	.001	.001	10	10
V	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
	1	.001	.001	6	6	.001	.001	6	6
	2	.001	.001	2	2	.031	.062	160	2
S	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
	1	.001	.001	2	2	.001	.001	2	2
	2	.001	.001	1	1	.031	.047	148	1
U	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
	1	.001	.001	27	27	.031	.047	121	27
	2	.001	.001	54	50	.031	.047	78	50
A	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
	1	.001	.001	6	6	.001	.001	6	6
	2	.001	.001	10	10	.015	.015	10	10
P5	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
	1	.001	.001	14	14	.001	.001	14	14
	2	.001	.001	25	25	.031	.031	77	25
P5X	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
	1	.001	.001	5	5	.001	.001	5	5
	2	.001	.001	1	1	.031	.078	240	1
UX	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
	1	.001	.001	41	41	.047	.063	132	41
	2	.093	.140	1,546	1,431	.703	2.781	1,632	1,431
AX	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

then some 40 seconds to detect them, while Requiem computes 43,111 non core rewritings and needs 1.5 hours to detect and remove them.

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 useful, function free rewriting. In RQR, resolution is performed exhaustively, leading to a large number of non function free rewritings that are eventually discarded. In ontology U, the superior performance of Rapid is due to the efficiency of the unfolding step, in particular to the non computation of subsumed 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 [6] 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 the core rewriting set of connected queries posed over DL-Lite$_R$ ontologies. Rapid optimizes the inference process by replacing the application of the first order resolution rule by specialized shrinking and unfolding rules, which save the algorithm from many unnecessary rewritings, subsumption checks and blind inference paths. The experimental evaluation of Rapid showed a significant performance benefit if compared to RQR, which in several practical cases can alleviate the exponential behavior. The specialized inference rules differentiate Rapid from pure resolution-based algorithms, however it remains committed to the computation of set of rewritings (i.e. of a UCQ), unlike recent approaches [8] which compute non recursive datalog programs, deferring thus the complexity to the underlying database system. The performance benefit that may result from computing and executing a non recursive datalog program instead of a rewriting set equivalent to a UCQ largely depends on the way relational engines deal with such datalog programs and needs to be better explored. Another interesting direction for future work is to apply the idea to more expressive DLs, like $\mathcal{ELHI}$.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Handbook of Automated Reasoning (in 2 volumes), vol. 1, pp. 19–99. Elsevier, MIT Press (2001)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39, 385–429 (2007)
4. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. J. of Artificial Intelligence Research 31, 157–204 (2008)
5. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Procs. of KR 2010, pp. 247–357 (2010)
6. 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)
7. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics 10, 133–173 (2008)
8. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Procs. of KR 2010, pp. 290–300 (2010)
9. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: Procs. of OWLED 2008. CEUR-WS.org, vol. 432 (2008)