Goal-oriented 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. We present an optimized query rewriting algorithm for OWL 2 QL that computes the rewriting set of a user query by avoiding unnecessary inferences and extended clause subsumption checks. The evaluation shows a significant performance improvement in comparison to other similar approaches. Alternatively, instead of a rewriting set, the algorithm can produce an equivalent non-recursive datalog program.

1 Introduction

The problem of answering *conjunctive queries* (CQ) over expressive DL ontologies suffers from high worst-case complexity. The DL-Lite$_R$ language [1], which underpins the OWL 2 QL profile, overcomes this problem by allowing a limited expressivity. In DL-Lite$_R$, the CQ answering problem is tractable from the data point of view, and can be solved by splitting the answering procedure in two steps [5, 1, 6]: 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 [4]. The main 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 [2, 7], 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. This restriction is lifted in [4], which proposes a resolution-based rewriting strategy, called RQR. However, its strategy may get tangled in long inference paths leading to unnecessary or to 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 performance. Another approach, called Presto, is proposed in [6] which, instead of a UCQ, computes a non-recursive datalog program, deferring thus part of the complexity to the database system, where it can be handled using disjunctive views.

In this paper we present Rapid$_f$, a goal-oriented query rewriting algorithm for queries posed over DL-Lite$_R$ ontologies. Instead of exhaustively performing resolution, it 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 and the need for extended query subsumption checks. Rapid$_f$ improves the Rapid algorithm [3] by supporting the full syntactic expressivity of DL-Lite$_R$ (i.e. axioms of the form $A \sqsubseteq \exists P.B$, $\exists P \sqsubseteq \exists S$, $\exists P \sqsubseteq \exists R.B$), all types of queries, and by further refining the unfolding step and reducing the need for subsumption checks. We describe also a simple modification of Rapid$_f$, called Rapid$_d$, which, instead of the complete set of rewritings, outputs an equivalent non recursive datalog program, similarly to [6].

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 shown in Table 1, where A, B are atomic concepts and P, S atomic roles. $\mathcal{A}$ is a finite set of *assertions* of the form $A(a)$ or $P(a, b)$, where a, b are individuals.

A CQ Q has the form $\mathbf{A} \leftarrow \mathcal{B}$, where atom $\mathbf{A}$ is the *head*, and the set of atoms $\mathcal{B}$ (seen as a conjunction) is the *body* of Q. We denote $\mathcal{B}$ 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* $S(t, s)$. $\mathsf{terms}\, \mathbf{B}$ ($\mathsf{vars}\, \mathbf{B}$, $\mathsf{cons}\, \mathbf{B}$) are the sets of terms (variables, constants) that appear in $\mathbf{B}$. For a set of atoms $\mathcal{B}$ we have $\mathsf{terms}\, \mathcal{B} = \bigcup_{\mathbf{B} \in \mathcal{B}} \mathsf{terms}\, \mathbf{B}$, for a CQ Q we have $\mathsf{terms}\, Q = \mathsf{terms}\, (\{\mathsf{head}\, Q\} \cup \mathsf{body}\, Q)$ and similarly for $\mathsf{vars}\, Q$, $\mathsf{cons}\, Q$. An atom or CQ is *function free* if it contains no functional terms. User queries are always function free. Given a function free CQ Q, a term $t \in \mathsf{terms}\, Q$ is called *distinguished* if it appears in $\mathsf{head}\, Q$, and *non distinguished* otherwise; *bound* if it is a constant, or a distinguished variable, or a variable that appears at least twice in $\mathsf{body}\, Q$, and *unbound* otherwise. We denote the set of bound terms, bound and unbound variables of Q by $\mathsf{terms}^{\mathsf{B}}\, Q$, $\mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}^{\mathsf{UB}}\, Q$, respectively. For an atom $\mathbf{A}$ we also write $\mathsf{vars}^{\mathsf{B}}\, \mathbf{A}$ and $\mathsf{vars}^{\mathsf{UB}}\, \mathbf{A}$ instead of $\mathsf{vars}\, \mathbf{A} \cap \mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}\, \mathbf{A} \setminus \mathsf{vars}^{\mathsf{B}}\, Q$, respectively, if it is clear that $\mathbf{A} \in \mathsf{body}\, Q$, for some 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 clausification of $\mathcal{O}$ into first order clauses (see Table 1, where it is assumed that each axiom introduces a distinct function f). 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 $\mathsf{cert}(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \bigcup_{Q' \in \mathcal{Q}} \mathsf{cert}(Q', \langle \emptyset, \mathcal{A} \rangle)$.

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 $Q \mathrel{\hat{\in}} \mathcal{Q}$. 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 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 $\mathcal{O}$.

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

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

3 Goal-oriented Query Rewriting

Rapid$_f$ computes $\mathsf{rewr}^{\mathsf{C}}\,(Q, \mathcal{O})$ for a user query Q in an efficient way. Its strategy 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 the query apart from $\mathbf{A}$.

Rapid$_f$ 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 happen at the shrinking step. (4) The *subsumption check* step, in which subsumed rewritings are removed.

For efficienty, Rapid$_f$ 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 non subsumed rewriting, and a large number of unnecessary inferences is avoided. The rules of Rapid$_f$ make use of the unfolding and function sets of atom.

3.1 Atom Unfolding Sets

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

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

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 the bound variables not to 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 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 (possibly empty) 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* $v\theta' \notin \mathsf{vars}\,\mathbf{A}$.

In the above, $\vdash_{\mathcal{R}}$ denotes derivability under the first-order resolution rule. 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 collect all unfoldings and 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 Table. 2, 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 $\hat{\mathcal{D}}(\mathbf{A}; T) = \mathcal{D}(\mathbf{A}; T) \cup \{\mathbf{A}\}$. It is easy to prove that given $\mathbf{A}$ and $T \neq \emptyset$ we have $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ iff $\mathbf{B}\theta' \in \mathcal{D}(\mathbf{A}; T)$, for θ, θ' as defined in Def. 1.

T	rule	T	rule
$\{\}$, $\{t\}$	$\frac{A(t) \quad A(x) \leftarrow B(x)}{B(t)}$	$\{\}$	$\frac{A(t) \quad A(f(x)) \leftarrow B(x)}{B(z)}$
$\{\}$	$\frac{A(t) \quad A(f(x)) \leftarrow S(x,y)}{S(z,w)}$	$\{\}$	$\frac{A(t) \quad A(f(x)) \leftarrow S(y,x)}{S(w,z)}$
$\{\}$, $\{t\}$	$\frac{A(t) \quad A(x) \leftarrow S(x,y)}{S(t,z)}$	$\{\}$, $\{t\}$	$\frac{A(t) \quad A(x) \leftarrow S(y,x)}{S(z,t)}$
$\{\}$, $\{t\}$	$\frac{P(t,v) \quad P(x,f(x)) \leftarrow B(x)}{B(t)}$	$\{\}$, $\{t\}$	$\frac{P(v,t) \quad P(f(x),x) \leftarrow B(x)}{B(t)}$
$\{\}$, $\{t\}$	$\frac{P(t,v) \quad P(x,f(x)) \leftarrow S(x,y)}{S(t,z)}$	$\{\}$, $\{t\}$	$\frac{P(v,t) \quad P(f(x),x) \leftarrow S(x,y)}{S(t,z)}$
$\{\}$, $\{t\}$	$\frac{P(t,v) \quad P(x,f(x)) \leftarrow S(y,x)}{S(z,t)}$	$\{\}$, $\{t\}$	$\frac{P(v,t) \quad P(f(x),x) \leftarrow S(y,x)}{S(z,t)}$
$\{t\}$, $\{s\}$, $\{\}$, $\{t,s\}$	$\frac{P(t,s) \quad P(x,y) \leftarrow S(x,y)}{S(t,s)}$	$\{t\}$, $\{s\}$, $\{\}$, $\{t,s\}$	$\frac{P(t,s) \quad P(x,y) \leftarrow S(y,x)}{S(s,t)}$

Table 2. The $\mathcal{J}(T)$ inference rules (w and z are any newly introduced variables).

3.2 Atom Function Sets

As we have already seen, the closure of $\Xi(\mathcal{O})$ contains clauses of the form $P(x, f(x)) \leftarrow B(x)$, $P(f(x), x) \leftarrow B(x)$ and $A(f(x)) \leftarrow B(x)$, as well as of the form $P(g(x), f(g(x))) \leftarrow B(x)$ and $P(g(x), f(g(x))) \leftarrow S(x, y)$. 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 a 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 or role that appears in the LHS of the axiom that introduces f. Let $\mathsf{atom}\,f[x]$ denote the atom that (a) has as predicate the entity associated with f, (b) has the variable x in the place of the bound variable of the respective axiom of Table 1 which introduces f, and (c) has a distinct variable (not elsewhere used, as needed) in the place of the unbound variable, if any. E.g. if the axiom $A \sqsubseteq \exists P.B$ introduces function f_1 then $\mathsf{atom}\,f_1[x]$ is the atom $A(x)$, and if the axiom $\exists S^- \sqsubseteq \exists P.B$ introduces function f_2 then $\mathsf{atom}\,f_2[x]$ is the atom $S(z, x)$, where z is some variable not used elsewhere. 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:*

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

It follows that (a) if $\mathbf{A} \equiv P(v,t)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} P(f(t),s) \leftarrow \mathsf{atom}\, f[t]$, (b) if $\mathbf{A} \equiv P(t,v)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} P(s,f(t)) \leftarrow \mathsf{atom}\, f[t]$, 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 $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} A(f(t)) \leftarrow \mathsf{atom}\, f[t]$. Again, T stands for the set of bound terms in $\mathbf{A}$.

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 $\mathsf{atom}\, f[x]$. Now, the functional term $f(x)$ 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 $P(x,v)$ and $P(v,y)$ and v is eliminated this way by unifying with $f(x)$, 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 $\mathcal{X} = \bigcup_{i=1}^{k}(\mathsf{vars}^{\mathsf{B}}\, \mathbf{B}_i) \setminus v$. The* shrinking rule $\mathcal{S}$ *on Q is defined as follows:*

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

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

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

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

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

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

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

3.5 Query Unfolding Optimization

If we apply exhaustively the $\mathcal{W}$ rule in order to compute $\mathcal{W}^*(Q)$, we may end up with many subsumed rewritings. Because the subsumption check operation needed to remove them is very costly, Rapid$_f$ applies $\mathcal{W}$ in a cleverer way, so as to get as few as possible subsumed rewritings. In fact, it restates the unfolding problem as follows: Given a CQ Q of the form $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, find the non subsumed CQs that are conclusions of all possible applications of $\mathcal{W}$ on Q. For convenience, define $\mathcal{B}_i = \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\,\mathbf{B}_i)$, so that we get the sequence of the possibly non disjoint unfolding sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$. For simplicity, we drop the substitutions γ_i in Def. 4 by assuming that each time a rule of $\mathcal{J}(T)$ that introduces a new variable is applied, this variable does not appear elsewhere.

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 unfolding sets that contain $\mathbf{B}$. We call the set $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ with $k \leq n$ a *selection* for Q iff (a) $\bigcup_{i=1}^{k} \mathsf{ind}\,\mathbf{C}_i = \mathbb{N}_n$, where $\mathbb{N}_n \doteq \{1, \ldots, n\}$, and (b) $\mathsf{ind}\,\mathbf{C}_i \setminus \mathsf{ind}\,\mathbf{C}_j \neq \emptyset$ for all $i, j \in \mathbb{N}_k$, i.e. if $\mathcal{C}$ contains at least one atom from each unfolding set and no two sets $\mathsf{ind}\,\mathbf{C}_i$ overlap fully. Clearly, a selection corresponds to an unfolding of Q, in particular to $\mathbf{A} \leftarrow \mathcal{C}$. However, of interest are the *minimal selections*, which can produce non subsumed rewritings. We call a selection $\mathcal{C}$ for Q minimal, iff there is no selection $\mathcal{C}'$ for Q such that $\mathcal{C}' \subset \mathcal{C}$, i.e. if condition (b) above is replaced by the stronger condition $\mathsf{ind}\,\mathbf{C}_i \setminus \left(\bigcup_{j=1,j\neq i}^{k} \mathsf{ind}\,\mathbf{C}_j\right) \neq \emptyset$ for all $i \in \mathbb{N}_k$, i.e. if all atoms $\mathbf{C}_i$ need to be present in set $\mathcal{C}$ in order for $\bigcup_{i=1}^{k} \mathsf{ind}\,\mathbf{C}_i = \mathbb{N}_n$ to hold. If this were not the case, we could form the selection $\mathcal{C}' = \{\mathbf{C}_1, \ldots, \mathbf{C}_{i-1}, \mathbf{C}_{i+1}, \mathbf{C}_k\} \subset \mathcal{C}$, hence $\mathcal{C}$ would not be minimal.

In this computation of minimal selections only equality between the elements of the sets $\mathcal{B}_i$ is taken into account, and not subsumption relations. However, an unfolding set may contain an atom with an unbound variable (e.g. $P(x, *)$, where $*$ is unbound) which unifies with an atom of another unfolding set that contains only bound variables (e.g. $P(x, y)$). The unfoldings of a CQ Q resulting after such unifications are made may subsume or be subsumed by several of the unfoldings given directly by other minimal selections for Q. In order to take into account atom subsumption relations, we compute all possible bindings for the unbound variables that appear in the sets $\mathcal{B}_i$ in advance and enrich the respective sets $\mathcal{B}_i$ with the respective atoms, before computing the minimal selections. In particular, if for some $i, j \in \mathbb{N}_n$ we have $\mathbf{C} \in \mathcal{B}_i$ and $\mathbf{C}' \in \mathcal{B}_j$ and there is a substitution θ on $\mathsf{vars}^{\mathsf{UB}}\,\mathbf{C}'$ such that $\mathbf{C}'\theta = \mathbf{C}$, we add $\mathbf{C}$ to $\mathcal{B}_j$. The presence of any such two atoms $\mathbf{C}'$ and $\mathbf{C}$ in any pair $\mathcal{B}_i, \mathcal{B}_j$, regardless of whether they were present from the beginning or introduced at the enrichment phase, establishes a *parent-child* relationship between $\mathbf{C}'$ and $\mathbf{C}$. Let $\mathsf{parents}\,\mathbf{C}$ and $\mathsf{children}\,\mathbf{C}$ denote the set of parent and child atoms of atom $\mathbf{C}$ across all the sets $\mathcal{B}_i$. Obviously, a child can have several parents in different unfolding sets, possibly distinct between each other, and the same holds for the children of a parent.

In order to avoid the production of subsumed rewritings due to such atom subsumptions, for each candidate unfolding $Q : \mathbf{A} \leftarrow \mathcal{C}$ obtained from a minimal selection $\mathcal{C}$, Rapid$_f$ performs two checks: (1) For each parent $\mathbf{C}$ of an atom in $\mathcal{C}$, it constructs a candidate rewriting with body $\mathcal{C}' = \{\mathbf{C}\} \cup (\mathcal{C} \setminus \mathsf{children}\,\mathbf{C})$, i.e it replaces all children of $\mathbf{C}$ by their parent. If $\mathcal{C}'$ is a minimal selection, then Q is discarded because it is subsumed by $\mathbf{A} \leftarrow \mathcal{C}'$. E.g. $Q(x) \leftarrow S(x, y), T(x, y)$ is subsumed by $Q(x) \leftarrow S(x, *), T(x, z)$ where $S(x, y)$ is a child of $S(x, *)$. (2) For each atom $\mathbf{C}$ that is a child of an atom in $\mathcal{C}$ it constructs the candidate body $\mathcal{C}' = \{\mathbf{C}\} \cup \{\mathbf{D} \mid \mathbf{D} \in \mathcal{C} \text{ and } \mathsf{ind}\,\mathbf{D} \nsubseteq \mathsf{ind}\,\mathbf{C}\}$, i.e. it replaces the parent by its child $\mathbf{C}$ and keeps all the remaining atoms of $\mathcal{C}$ that are not ‘covered’ (in terms of their indices) by $\mathbf{C}$. If $\mathsf{cond}\,(\mathbf{A} \leftarrow \mathcal{C}') \rhd Q$ then Q is discarded because it is subsumed. E.g. $Q(x, y) \leftarrow R(x, y), S(y, w), T(y, z), S(v, z)$ is subsumed by $Q(x, y) \leftarrow R(x, y), S(y, z)$, where $S(y, z)$ is child of both $S(y, w)$ and $S(v, z)$.

The only case an unfolding Q' of Q obtained in this way may subsume another unfolding of Q is when the condensation of Q' does not contain one or more of the variables in $\mathsf{vars}^{\mathsf{B}}\,Q$; this implies that a structural change has happened to $\mathsf{cond}\,Q'$. To cover this case, we always compute the condensation of each unfolding given by the above procedure. If the condensation does not contain a bound variable of Q it is marked as *impure*, otherwise as *pure*. Given that the unfolding step is executed for each rewriting produced by the shrinking step, the final step is the check for subsumed rewritings within the results of the entire unfolding process. The check is done after first grouping the results into sets that are known not to contain subsumed rewritings. As explained, these are the sets of pure unfoldings obtained during the unfolding step for each rewriting given by the shrinking step. Each impure unfolding is considered to be a separate set.

The overall structure of Rapid$_f$ for a user query Q is shown in Algorithm 1. Procedure Shrink computes $\mathcal{S}^*(Q)$, by iteratively applying Def 3. For each rewriting computed by Shrink, procedure Unfold computes its minimal selections and discards any subsumed unfoldings as described above. Finally, the unfoldings, grouped into sets of pure unfoldings and singleton sets of impure unfoldings, are processed by procedure CheckSubsumption, which checks for subsumptions across sets only and removes any subsumed rewritings.

Algorithm 1 The Rapid$_f$ algorithm

```
procedure RAPID$_f$(CQ $Q$, ontology $\mathcal{O}$)
    $\mathcal{Q}_f = \emptyset$
    for all $Q_s \in$ SHRINK$(Q, \mathcal{O})$ do
        $\mathcal{Q}_t \leftarrow \emptyset$
        for all $Q' \in$ UNFOLD$(Q_s, \mathcal{O})$ do
            if $\mathsf{vars}^{\mathsf{B}}\, Q \subseteq \mathsf{vars}\,(\mathsf{cond}\, Q')$ then
                $Q_t \leftarrow Q_t \cup \{Q'\}$
            else
                $Q_f \leftarrow Q_f \cup \{\{\mathsf{cond}\, Q'\}\}$
            end if
        end for
        $Q_f \leftarrow Q_f \cup \{Q_t\}$
    end for
    return CHECKSUBSUMPTION$(Q_f)$
end procedure
```

3.6 Rapid$_d$: Rewritings as a non-recursive Datalog program

The side premises $\mathbf{C}_i \in \hat{\mathcal{D}}(\mathbf{B}_i, \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i)$ of the unfolding rule $\mathcal{W}$ may be seen as the clauses $\mathbf{C}_i \leftarrow \mathbf{D}_i$ for some $\mathbf{D}_i \in \hat{\mathcal{D}}(\mathbf{B}_i, \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i)$. Hence, similarly to [6], given a user CQ Q, instead of applying exhaustively the unfolding rule on $\mathcal{S}^*(Q)$, in order to produce all unfoldings and then check for subsumptions among them so as to get $\mathsf{rewr}^{\mathsf{C}}\,(Q, \mathcal{O})$, we can produce a non-recursive datalog program $\mathcal{P}_Q$, which contains the rewritings produced at the shrinking step plus the side premises of the $\mathcal{W}$ rules that can possibly be applied. Rapid$_d$ works exactly this way: The clausification and shrinking steps are as in Rapid$_f$, but the unfolding and subsumption check steps are replaced by a single step which rewrites the unfolding of all atoms that appear in the body of the rewritings in $\mathcal{S}^*(Q)$ in the form of a set of clauses $\mathcal{U}$, which are then appended to the set of rewritings obtained at the shrinking step so that $\mathcal{P}_Q$ is produced. Before doing this however, the rewritings in $\mathcal{S}^*(Q)$ need to be modified in two ways, as in [6].

First, we can remove from the body of the several $Q' \in \mathcal{S}^*(Q)$ the atoms that will certainly produce only subsumed rewritings (in the case we were to apply exhaustively the $\mathcal{W}$ rule, as before, in order to compute all rewritings); this happens if there are two atoms $\mathbf{A}, \mathbf{B} \in \mathsf{body}\, Q'$ and a θ such that $\mathbf{A} = \mathbf{C}\theta$ for some $\mathbf{C} \in \mathcal{D}(\mathsf{terms}^{\mathsf{B}}\, \mathbf{B})$, i.e. if $\mathbf{A}$ is subsumed by $\mathbf{C}$. In this case we just remove $\mathbf{B}$ from $\mathsf{body}\, Q'$. Let $\mathsf{rr}\,(\mathcal{S}^*(Q))$ be the set of clauses obtained in this way from $\mathcal{S}^*(Q)$ and also after removing from it any subsumed clauses.

Next, we must construct the set of clauses $\mathcal{U}$. This is straightforward, but we must take into account the different bindings that bound and unbound variables can have during the unfolding. So, for all atoms $\mathbf{A}$ that appear in the clauses of $\mathsf{rr}\,(\mathcal{S}^*(Q))$ we compute the set $\mathcal{D}(\mathbf{A}; \mathsf{vars}^{\mathsf{B}}\, \mathbf{A})$ and then we construct from $\mathbf{A}$ a new atom $\mathbf{A}'$ by removing from the arguments of $\mathbf{A}$ all unbound variables and replacing the predicate p of $\mathbf{A}$ by a new predicate p_{t_1} or $p_{t_1 t_2}$, if $\mathbf{A}$ is a concept or role atom, respectively, and $t_i = 0$ if the i-th argument of $\mathbf{A}$ is unbound and

$t_i = 1$ otherwise. Finally we *normalize* the clauses in $\mathsf{rr}\,(\mathcal{S}^*(Q))$ by replacing all appearances of $\mathbf{A}$ by $\mathbf{A}'$, and add to $\mathcal{U}$ the clause $\mathbf{A}' \leftarrow \mathbf{A}$ as well as the clause $\mathbf{A}' \leftarrow \mathbf{D}$ for all $\mathbf{D} \in \mathcal{D}(\mathbf{A}; \mathsf{vars}^{\mathsf{B}}\,\mathbf{A})$.

$\mathcal{P}_Q$ is the union of $\mathcal{U}$ and the normalized version of all clauses in $\mathsf{rr}\,(\mathcal{S}^*(Q))$.

4 Evaluation

We evaluated Rapid$_f$ by comparing it with Rapid and Requiem, the implementation of RQR. We used the same datasets 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 are obtained by rewriting U, A and P5 without qualified existential restrictions). The results are shown in Table 3. T is the rewriting computation time and R the number of rewritings. For Rapid$_f$ (Rap$_f$), Rapid (Rap) and Requiem (Req), one number is given for the rewritings, since all these algorithms compute the same core rewriting set. For Rapid$_d$ (Rap$_d$), the column R is the number of clauses in $\mathcal{P}_Q$.

		Rap$_f$	Rap	Req		Rap$_d$				Rap$_f$	Rap	Req		Rap$_d$	
$\mathcal{O}$	Q	T	T	T	R	T	R	$\mathcal{O}$	Q	T	T	T	R	T	R
	1	.001	.001	.001	15	.001	16		1	.001	.001	.001	6	.001	7
	2	.001	.001	.001	10	.001	13		2	.001	.001	.015	10	.001	16
V	3	.001	.015	.016	72	.001	29	**P5**	3	.001	.001	.047	13	.001	19
	4	.015	.031	.062	185	.001	44		4	.015	.015	.688	15	.001	21
	5	.016	.016	.015	30	.001	13		5	.015	.015	16.453	16	.001	22
	1	.001	.001	.001	6	.001	7		1	.001	.001	.001	14	.001	15
	2	.001	.001	.062	2	.001	3		2	.001	.001	.031	25	.001	31
S	3	.001	.001	.515	4	.001	5	**P5X**	3	.016	.031	.297	58	.001	34
	4	.001	.001	1.047	4	.001	5		4	.078	.172	7.375	179	.001	36
	5	.001	.001	17.984	8	.001	7		5	1.234	2.625	3:48.690	718	.001	37
	1	.001	.001	.001	2	.001	4		1	.001	.001	.001	5	.001	7
	2	.001	.001	.047	1	.001	2		2	.001	.001	.078	1	.001	2
U	3	.001	.001	.109	4	.001	8	**UX**	3	.001	.001	1.125	12	.001	10
	4	.001	.001	2.031	2	.001	3		4	.001	.001	19.375	5	.001	6
	5	.001	.001	7.781	10	.001	8		5	.001	.015	57.672	25	.001	11
	1	.001	.001	.047	27	.001	54		1	.001	.015	.063	41	.001	69
	2	.001	.001	.047	50	.001	33		2	.109	.141	2.781	1,431	.001	51
A	3	.016	.016	.063	104	.001	33	**AX**	3	.375	.469	29.109	4,466	.001	57
	4	.015	.031	.156	224	.001	60		4	.265	.641	23.516	3,159	.001	85
	5	.062	.078	.610	624	.001	38		5	3.375	49.984	1:56:21.585	32,921	.001	72

Table 3. Evaluation results. The times T are in hh.mm.ss.msec format The results for Requiem are for its greedy modality, which applies forward query subsumption, dependency graph pruning and greedy unfolding.

The results show clearly the efficiency of Rapid$_f$. It is always faster than Rapid, and much faster than Requiem; in several cases the improvement is significant. The most striking case is ontology AX and query 5, in which Rapid$_f$ completes the computation of the 32,921 core rewritings in less than 4 seconds, while Rapid needs 50 seconds and Requiem about 2 hours. The more detailed study of this particular case showed that Rapid$_f$ computes directly the final core rewriting set and performs no subsumption checks at all. On the other hand, Rapid spends about 45 seconds checking for subsumptions and Requiem about 1.5 hours.

Table 3 also shows, as expected, that Rapid$_d$ is always much faster than any of the other algorithms, since it does not include the unfolding step, which is the main source of complexity, even for the optimized Rapid$_f$ algorithm. For the same ontologies and query pairs tested in [6], similar times and numbers of rewritings are reported. Note, however that the rewriting sizes do not coincide, because Rapid$_d$ and Presto do not produce the same datalog programs. This is due to the fact that the *Split* and *EliminateEJVars* steps of Presto are performed in a different way by the shrinking step of Rapid$_d$. The expansion of the datalog program to a UCQ is of course the same, for both algorithms.

5 Conclusions

We presented Rapid$_f$, an efficient algorithm for the computation of the core rewriting set of queries posed over DL-Lite$_R$ ontologies. Rapid$_f$ 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. We presented also Rapid$_d$ a modification of Rapid$_f$, which does not unfold the rewritings, but encodes the unfoldings into a datalog program similarly to [6]. The experimental evaluation of Rapid$_f$ showed a significant performance benefit if compared to RQR and Rapid, which in several practical cases can alleviate the exponential behavior. The performance of Rapid$_d$ is similar to Presto, but Rapid$_d$ supports the full syntactic expressivity of DL-Lite$_R$.

References

1. A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev. The DL-Lite family and relations. J. of Artificial Intelligence Research, 36:1–69, (2009).
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite Family. J. of Automated Reasoning, 39:385–429, (2007).
3. A. Chortaras, D. Trivela, G. Stamou. Optimized query rewriting for OWL 2 QL, In Procs of CADE 2011 (*accepted*), (2011).
4. H. Perez-Urbina, I. Horrocks, 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, R. Rosati. Linking data to ontologies. J. on Data Semantics, 10:133–173, (2008).
6. R. Rosati, A. Almatelli. Improving query answering over DL-Lite ontologies. In Procs of KR 2010, pp. 290–300, (2010)
7. M. Stocker, M. Smith. Owlgres: A scalable OWL reasoner. In Procs of OWLED 2008, CEUR-WS.org Vol-432, (2008).