Compact Rewriting for Existential Rules

Michaël Thomazo

Compact Rewritings for Existential Rules*

Michaël Thomazo
University of Montpellier
France
thomazo@lirmm.fr

Abstract

Querying large databases while taking ontologies into account is currently a very active domain research. In this paper, we consider ontologies described by existential rules (also known as Datalog+/-), a framework that generalizes lightweight description logics. A common approach is to rewrite a conjunctive query w.r.t an ontology into a union of conjunctive queries (UCQ) which can be directly evaluated against a database. However, the practicability of this approach is questionable due to 1) the weak expressivity of classes for which efficient rewriters have been implemented 2) the large size of optimal rewritings using UCQ. We propose to use semi-conjunctive queries (SCQ), which are a restricted form of positive existential formulas, and compute sound and complete rewritings, which are union of SCQ (USCQ). A novel algorithm for query rewriting, COMPACT, is presented. It computes sound and complete rewritings for large classes of ontologies. First experiments show that USCQ are both efficiently computable and more efficiently evaluable than their equivalent UCQ.

1 Introduction

Querying data while taking ontologies into account is a currently active research domain, often referred to as ontology-based data access (OBDA). Data is typically stored in a database and basic queries are conjunctive queries. Different means are available to represent ontologies. The mainstream approach uses description logics [Baader *et al.*, 2007]. In that case, most studies for OBDA focuses on lightweight description logics, such as DL-Lite [Calvanese *et al.*, 2007] or $\mathcal{EL}$ [Baader, 2003]. In this paper, we focus on existential rules [Baget *et al.*, 2011], similar to Datalog$^{\pm}$ [Calì *et al.*, 2012]. On the one hand, they cover lightweight description logics (and in fact, Horn Description Logics); on the other hand, these rules allow for more flexibility, since both body and head need not to be tree-shaped (as in DLs) and predicates can be of any arity (whereas they are restricted to arity 1 or 2 in DLs).

The fundamental decision problem[1] we consider is the following: given a set of facts F, a (Boolean) conjunctive query Q, and a set of existential rules $\mathcal{R}$, is it true that $F, \mathcal{R} \models Q$? Due to the presence of existential variables in the head of rules, this problem is undecidable. A variety of approaches for query answering have been investigated, based on two main mechanisms. First, some approaches include a step of *materialization*, that is, all or part of the data that can be inferred is added to the initial facts. These approaches include "pure" forward chaining (where we add all data that can be inferred) and combined approach [Kontchakov *et al.*, 2011]. Combined approach (where a step of query rewriting is added) can in particular be applied to DL-Lite and $\mathcal{EL}$ ontologies, and is thus arguably useful for practical purposes. A similar approach can be applied to a class of existential rules generalizing guarded rules [Thomazo *et al.*, 2012]. However, materialization-based approaches suffer from several drawbacks: they require read and write permits, the saturation step incurs a blow-up of data (and this is not acceptable when data is huge), and, to the best of our knowledge, no solution has been proposed when data is often changing, that is when facts are added or removed on a regular basis.

Other approaches are based on "pure" query rewriting: given Q and $\mathcal{R}$, a new query Q' is computed such that for all set of facts F, $F, \mathcal{R} \models Q$ if and only $F \models Q'$. Only read permits are necessary, no blow-up of data occurs since it is not changed, and data updates do not impact the rewritings. Mostly two kinds of rewritings occurs in the literature: Datalog programs and union of conjunctive queries. Polynomial rewriting in Datalog programs have been proposed for some existential rules [Gottlob and Schwentick, 2012; Stefanoni *et al.*, 2012; Rosati and Almatelli, 2010]. They however do not cover all classes of rules treated here, and their efficient evaluation remains an open problem.

More widely studied widely rewritings are using union of conjunctive queries (UCQs). The rationale is that most of the available data is stored in databases, and existing database systems are optimized to efficiently evaluate con-

*This work was partially supported by ANR project PAGODA (ANR 12 JS02 007 01).

[1] We consider only the decision problem for the sake of simplicity. We could consider non-boolean queries, as well as union of conjunctive queries instead of a single CQ.

junctive queries. This strand of work, initiated in [Calvanese *et al.*, 2007], has led to a number of prototypes, such as, among others, Iqaros [Venetis *et al.*, 2012], Nyaya [Gottlob *et al.*, 2011], QuOnto [Calvanese *et al.*, 2007], Rapid [Chortaras *et al.*, 2011], Requiem [Pérez-Urbina *et al.*, 2009], or the algorithm presented in [König *et al.*, 2012], relying on piece-based rewriting. More details about the characteristics of some of them and how they relate with this work will be presented in Section 6.

However, as all these tools compute rewritings which are UCQs, they share the major drawback of these rewritings: even reasonable sets of existential rules can generate huge rewritings. We propose to produce rewritings that are neither a union of conjunctive queries nor a Datalog program, but a union of so-called semi-conjunctive queries (SCQ). We advocate that such queries can be both efficiently computed and evaluated. For that purpose, we adopt a two-step approach:

1. we design a novel algorithm that produces a union of semi-conjunctive queries (USCQ), for any set of existential rules and any query that admits a finite union of most general (CQ) rewritings. It computes short rewritings, and first experiments show that they are generated faster than their equivalent UCQ by state-of-the-art tools, even though these tools are dedicated to that particular case.
2. we compare the evaluation efficiency of the USCQ with the evaluation efficiency of the optimal rewriting in conjunctive queries. First experiments show that the view-based evaluation of USCQ is more efficient than the evaluation of the corresponding UCQ.

We first recall some technical preliminaries, and in particular piece unifiers, that we generalize in this paper. Section 3 introduces semi-conjunctive queries. Section 4 introduces a generalization of piece unifiers for semi-conjunctive queries, as well as the COMPACT algorithm. We evaluate both the rewriting step and the evaluation step of COMPACT in Section 5. Last, we stress similarities and novelties with respect to existing algorithms in Section 6.

2 Preliminaries

We consider first-order logical (FOL) languages with constants but no other function symbols. A language $\mathcal{L} = (\mathcal{P}, \mathcal{C})$ is composed of two disjoint sets: a set $\mathcal{P}$ of predicates and a set $\mathcal{C}$ of constants. An atom on $\mathcal{L}$ is of form $p(t_1, \dots, t_k)$ where p is a predicate in $\mathcal{P}$ of arity k and t_i are terms, i.e., variables or constants in $\mathcal{C}$. A fact is an existentially closed conjunction of atoms[2]. A conjunctive query is an existentially quantified conjunction of atoms - it is Boolean if it is closed.

Definition 1 *An* existential rule *(or simply* rule*)* $R = (B, H)$ *on a language* $\mathcal{L}$ *is a closed formula of form* $\forall x_1 \dots \forall x_p (B \rightarrow \exists z_1 \exists z_q\ H)$ *where* B *and* H *are two (finite) conjunctions of atoms on* $\mathcal{L}$*;* $\{x_1, \dots, x_p\} = \mathsf{var}(B)$*; and* $\{z_1, \dots, z_q\} = \mathsf{var}(H) \setminus \mathsf{var}(B)$.[3] B *and* H *are respectively called the* body *and the* head *of* R*, also denoted by* $\mathsf{body}(R)$ *and* $\mathsf{head}(R)$*. The* frontier *of* R *(denoted by* $\mathsf{fr}(R)$*) is the set of variables occurring in both* B *and* H*:* $\mathsf{fr}(R) = \mathsf{var}(B) \cap \mathsf{var}(H)$.

W.l.o.g., we will consider that all existential rules have atomic head, i.e., the head of the rule is restricted to a single atom. Moreover, we will assume (w.l.o.g.) that rules and queries share no variables.[4] We will often consider logical formulas as sets: a conjunction or a disjunction of atoms will be seen as a set of atoms, and a conjunction of disjunctions will be seen as a set of disjunctions. It will be clear from the context which object is dealt with. Given two atom sets A and B, a homomorphism from A to B is a substitution h of $\mathsf{vars}(A)$ by $\mathsf{terms}(B)$ such that $h(A) \subseteq B$. Given two formulas φ and ψ, we say that φ is more general than ψ (denoted by $\varphi \geq \psi$) if $\psi \models \varphi$. In the case of (existentially closed) conjunctions of atoms, $A \geq B$ if and only if there exists a homomorphism from A to B.

We recall below the definition of piece unifiers [König *et al.*, 2012], which will be extended in this work. If rules were plain Datalog rules, this unification would be the classical one. However, in order to take existential variables of the head into account, we may unify several atoms at once. If Q is a set of atoms, for any set $Q' \subseteq Q$, we define $\mathsf{sep}(Q') = \mathsf{var}(Q') \cap \mathsf{var}(\bar{Q}')$, where $\bar{Q}' = Q \setminus Q'$.

Definition 2 (Piece unifier) *Let* Q *be a CQ and* R *be a rule. A* piece unifier *of* Q *with* R *is a pair* $\mu = (Q', u)$ *with* $Q' \subseteq Q$*,* $Q' \neq \emptyset$*, and* u *is a substitution of* $\mathsf{fr}(R) \cup \mathsf{var}(Q')$ *by* $\mathsf{terms}(\mathsf{head}(R)) \cup \mathcal{C}$ *such that:*

1. *for all* $x \in \mathsf{fr}(R)$*,* $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$ *(for technical convenience, we allow* $u(x) = x$*);*
2. *for all* $x \in \mathsf{sep}(Q')$*,* $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$*;*
3. $u(Q') \subseteq u(\mathsf{head}(R))$.

We define the notion of rewriting using piece unifiers. Given a query Q, Q_k is an $\mathcal{R}$-rewriting of Q if Q_k can be obtained from Q by a sequence of rewritings.

Definition 3 (Rewriting) *Given a CQ* Q*, a rule* R *and a piece unifier* $\mu = (Q', u)$ *of* Q *with* R*, the* rewriting *of* Q *according to* μ*, denoted* $\beta(Q, R, \mu)$*, is* $u(\mathsf{body}(R) \cup \bar{Q}')$.

Example 1 *Let* $Q = p(x) \wedge r(x, y) \wedge r(z, y) \wedge q(z)$*, and* $R = h(x_1) \rightarrow r(x_1, y_1)$*. There is only one unifier of* R *with* Q*, which is* $\mu = (\{r(x, y), r(z, y)\}, \{x \rightarrow x_1, y \rightarrow y_1, z \rightarrow x_1\})$*, and* $\beta(Q, R, \mu) = p(x_1) \wedge h(x_1) \wedge q(x_1)$.

Definition 4 ($\mathcal{R}$-rewriting of Q) *Let* Q *be a CQ and* $\mathcal{R}$ *be a set of rules. An* $\mathcal{R}$*-rewriting of* Q *is a CQ* Q_k *obtained by a finite sequence* $(Q_0 = Q), Q_1, \dots, Q_k$ *such that for all* $0 \leq i < k$*, there is* $R_i \in \mathcal{R}$ *and a piece unifier* μ *of* Q_i *with* R_i *such that* $Q_{i+1} = \beta(Q_i, R, \mu)$.

Theorem 1 (Soundness and completeness) *(basically [Salvat and Mugnier, 1996]) Let* F *be a fact,* $\mathcal{R}$ *be a set of existential rules, and a (Boolean) CQ* Q*. Then* $F, \mathcal{R} \models Q$ *iff there is an* $\mathcal{R}$*-rewriting* Q' *of* Q *such that* $F \models Q'$.

In this paper, we will consider rewritings that are not necessarily CQs or UCQs. Given a class Φ of formulas, we define sound and complete Φ-rewritings.

[2] This generalizes the usual notion of fact.

[3] Quantifiers are usually omitted, since there is no ambiguity.

[4] One can always rename apart the variables in a rule and thus satisfy our assumption.

Definition 5 (Φ-rewriting soundness and completeness) *Let Q be a first-order formula, Φ be a class of first-order formulas and $\mathcal{R}$ be a set of existential rules.*

- *A* sound Φ-rewriting *of Q (w.r.t $\mathcal{R}$) is a formula φ belonging to Φ such that for each fact F, $F \models \varphi$ implies that $F, \mathcal{R} \models Q$.*
- *A* complete Φ-rewriting *of Q (w.r.t $\mathcal{R}$) is a formula φ belonging to Φ such that for each fact F, $F, \mathcal{R} \models Q$ implies that $F \models \varphi$.*

An interesting class of rules when performing backward chaining is the class of *finite unification sets*, which are sets of rules for which any query admits a finite UCQ-rewriting.

Definition 6 (Finite unification set) *Let $\mathcal{R}$ be a set of rules. $\mathcal{R}$ is a* finite unification set *(*fus*) if for any Q there exists a finite UCQ-rewriting of Q (w.r.t. $\mathcal{R}$).*

The *fus* property is unrecognizable. However, some recognizable subclasses are known: atomic-body rules [Calì *et al.*, 2008; Baget *et al.*, 2009], (join)-sticky-rules [Calì *et al.*, 2010], aGRD [Baget *et al.*, 2011].

3 Semi-conjunctive Queries

CQs are considered as the basic queries in databases, and UCQs can be dealt with by processing each CQ separately. Hence, most of the research effort in OBDA has focused on UCQs. The query rewriting approach usually focuses on creating UCQ-rewritings. One of the reasons is that UCQs are efficiently dealt with by existing DBMS. However, this last statement is questionable in this setting, since the size of UCQ-rewritings is generally large, in particular when ontologies contain large class/role hierarchies (which is often the case). We first introduce the notion of a semi-conjunctive query - which allows for a limited use of disjunction - and illustrate with Example 3 what it has to offer to OBDA.

Definition 7 (Semi-conjunctive query) *A* semi-conjunctive query *(SCQ) is a closed logical formula of the following form:*

$$\exists \mathbf{x}\ \mathrm{D_1} \wedge \mathrm{D_2} \wedge \ldots \wedge \mathrm{D_n}$$

where D_i is a disjunction of atoms (for any i), and $\mathbf{x}$ is the set of variables that appear in the formula.

Definition 8 (Selection) *Let $S = \bigwedge_i D_i$ be an SCQ. A CQ $Q = \bigwedge_i d_i$ is a selection of S if, for each i, we have $d_i \in D_i$.*

Example 2 (Selection) *Let $S = (r_1(x, y) \vee r_2(x, y)) \wedge (s_1(y, z) \vee s_2(y, z))$. S has four selections, which are $r_1(x, y) \wedge s_1(y, z), r_1(x, y) \wedge s_2(y, z), r_2(x, y) \wedge s_1(y, z), r_2(x, y) \wedge s_2(y, z)$.*

Example 3 *By generalizing Example 2 with k disjunctions of q atoms, the smallest UCQ equivalent to a single USCQ would contain q^k CQs.*

Property 1 *Let Q be a CQ, and S be an SCQ. S is a sound SCQ-rewriting of Q w.r.t. $\mathcal{R}$ if and only if every selection of S is a sound CQ-rewriting of Q w.r.t. $\mathcal{R}$.*

If S_1 is more general than S_2, one can discard S_2 from a USCQ-rewriting. We thus define the notion of *cover* of a set $\mathcal{S}$ of SCQs, which contains only most general elements of $\mathcal{S}$.

Definition 9 (Cover) *Let $\mathcal{S}$ be a set of first-order queries. A* cover *of $\mathcal{S}$ is a set $\mathcal{S}^c \subseteq \mathcal{S}$ such that:*

1. *for any $S \in \mathcal{S}$, there is $S' \in \mathcal{S}^c$ such that $S \leq S'$,*
2. *elements of $\mathcal{S}^c$ are pairwise incomparable w.r.t. $\leq$.*

Of course, a cover of a sound and complete USCQ-rewriting is also sound and complete. In the case of UCQ-rewriting, being a sound and complete cover is a sufficient condition for being of minimal size [König *et al.*, 2012]. However with SCQs, this condition does not ensure the minimality of the USCQ-rewriting, as shown by the Example 4.

Example 4 *Let $\mathcal{S}_1$ be a set of SCQs containing:*

- $(r_1(x, y) \vee r_2(x, y)) \wedge (s_1(x, y) \vee s_2(x, y))$,
- $(r_1(x, y) \vee r_3(x, y)) \wedge (s_1(x, y) \vee s_2(x, y))$,
- $(r_1(x, y) \vee r_2(x, y)) \wedge (s_1(x, y) \vee s_3(x, y))$,
- $(r_1(x, y) \vee r_3(x, y)) \wedge (s_1(x, y) \vee s_3(x, y))$.

All elements of $\mathcal{S}_1$ are incomparable by the "more general" relation, but $\mathcal{S}_2 = \{(r_1(x, y) \vee r_2(x, y) \vee r_3(x, y)) \wedge (s_1(x, y) \vee s_2(x, y) \vee s_3(x, y))\}$ is equivalent to $\mathcal{S}_1$ and contains strictly less SCQs.

Property 2 *Let S and S' be two SCQs. $S \geq S'$ iff for every selection Q' of S', there exists a selection Q of S s.t. $Q \geq Q'$.*

4 Query Rewriting Using SCQs

In this section, we present COMPACT, an algorithm that computes USCQ-rewritings. We define for that purpose piece unifiers for SCQs. This notion naturally generalizes the corresponding notion for CQs by operating a selection on SCQs. As for CQs, given an SCQ S and a set of disjunctions $S' \subseteq S$, we define $\mathit{sep}(S') = \mathit{var}(S') \cap \mathit{var}(\bar{S}')$, where $\bar{S}' = S \setminus S'$.

Definition 10 (Piece unifier) *Let S be an SCQ and R be a rule. A* piece unifier *of S with R is a triple $\mu = (S', Q', u)$ with $S' \subseteq S$, $S' \neq \emptyset$, Q' a selection of S', and u is a substitution of $\mathsf{fr}(R) \cup \mathsf{vars}(Q')$ by $\mathsf{terms}(\mathsf{head}(R)) \cup \mathcal{C}$ such that:*

1. *for all $x \in \mathsf{fr}(R)$, $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$ (for technical convenience, we allow $u(x) = x$);*
2. *for all $x \in \mathsf{sep}(S')$, $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$;*
3. *$u(Q') \subseteq u(\mathsf{head}(R))$.*

A unifier of a rule with one body atom is local *if S' contains only one disjunction, and if u restricted to $\mathsf{terms}(S')$ is injective and does not map a variable to a constant. A non-local unifier $\mu = (S', Q', u)$ is* prime *if for any $D \in S'$, there exists no u_L, substitution of $\mathsf{fr}(R) \cup \mathsf{vars}(Q'_D)$ by $\mathsf{terms}(\mathsf{head}(R)) \cup \mathcal{C}$, such that $\mu = (\{D\}, Q'_D, u_L)$ is a local unifier, where Q'_D is the selection of $\{D\}$ that selects the same elements as Q'.*

Example 5 *Let $R_1 = p(x) \rightarrow r(x, y)$, $R_2 = q(x') \wedge h(x') \rightarrow s(x', y')$, and $S = r(x_1, x_2) \wedge t(x_1, x_3) \wedge r(x_3, x_4) \wedge s(x_1, x_5) \wedge s(x_3, x_5)$. S is a CQ, thus an SCQ too.*

- *let $\mu_1 = (\{\{r(x_1, x_2)\}\}, \{r(x_1, x_2)\}, \{u_1(x_1) = x, u_1(x_2) = y\})$. μ_1 is a local unifier of R_1 with S.*

- *let* $\mu_2 = (\{\{s(x_1,x_5)\},\{s(x_3,x_5)\}\},\{s(x_1,x_5), s(x_3,x_5)\},\{u_2(x_1) = x', u_2(x_5) = y', u_2(x_3) = x'\})$ *is a prime (non-local) unifier of* R_2 *with* S*, because* R_2 *has a non-atomic body*
- *let* $\mu_3 = (\{\{r(x_1,x_2)\},\{r(x_3,x_4)\}\},\{r(x_1,x_2), r(x_3,x_4)\},\{u_3(x_1) = x, u_3(x_2) = y, u_3(x_3) = x, u_3(x_4) = y\})$. μ_3 *is a non-local (two disjunctions unified at once), non-prime unifier (because* μ_1 *is a local unifier) of* R_1 *with* S.
- *let* $\mu_4 = (\{\{s(x_1,x_5)\}\},\{s(x_1,x_5)\},\{u_4(x_1) = x', u_4(x_5) = y'\})$. μ_4 *is not a unifier of* R_2 *with* S*, because* x_5 *is mapped to an existential variable of* R_2*, and there is a disjunction of* S *containing* x_5 *but not belonging to the unified disjunctions.*

To define the rewriting operations of our algorithm, we first need the definition of X-*entailment*.

Definition 11 (X-entailment) *Let* D *be a set of atoms, and* X *a set of variables. Let* a *be an atom.* a *is* X-entailed *by* D *if there is a homomorphism* π *from* a *to* D *such that if* $x \in \mathsf{var}(a) \cap X$*, then* $\pi(x) = x$.

Example 6 *Let* $D = \{r(x,y), p(x,u)\}$. $p(x,v)$ *is* $\{x,y\}$-*entailed by D, but* $r(y,x)$ *is not.*

We distinguish two kinds of rewriting operations. Local rewritings, which are performed when rewriting w.r.t. a local unifier, and non-local rewritings otherwise. Local rewritings are the novelty of our approach: they can introduce disjunctions. Non-local rewritings are a simple recast of usual rewritings in the framework of SCQs. Disjunctions that have a unified atom are removed, the substitution is applied to each atom of the body of the rule, creating a new disjunction for each of these atoms.

Definition 12 (Local rewriting) *Let* $S = \bigwedge_{i=1}^{n} D_i$ *be an SCQ,* R *be an atomic body rule and* $\mu = (\{D_1\}, Q', u)$ *be a local piece unifier of* R *with* S*. The* local rewriting *of* S *with respect to* μ *(denoted by* $\gamma_L(S,R,\mu)$*) is* $S' = D'_1 \wedge \bigwedge_{i=2}^{n} u(D_i)$*, where* $D'_1 = u(D_1) \vee u(\mathsf{body}(R))$*, if* $u(\mathsf{body}(R))$ *is not* sep$(\{D_1\})$*-entailed by* D_1*, and* S *otherwise.*

Checking that $u(\mathsf{body}(R))$ is not *sep*$(\{D_1\})$-relatively entailed by D_1 ensures that the same unification will not add twice equivalent atoms. Thus, given an SCQ, we can saturate it by applying local rewritings until all of them have been applied. This process, called *LU-Saturation*, is presented in Algorithm 1.

Definition 13 (Non-local rewriting) *Let* $S = \bigwedge_{i=1}^{n} D_i$ *be an SCQ,* R *be a rule (with* $\mathsf{body}(R) = \bigwedge_{i=1}^{b} b_i(R)$*) and* $\mu = (\{D_1,\ldots,D_k\}, Q', u)$ *be a non local unifier of* R *with* S*. The* non-local rewriting *of* S *with respect to* μ *(denoted by* $\gamma_{NL}(S,R,\mu)$*) is* $S' = \bigwedge_{j=1}^{b} D'_j \wedge \bigwedge_{i=k+1}^{n} u(D_i)$*, where* $D'_j = u(b_i(R))$.

Example 5 ((cont.) Local and non-local rewriting) *The rewriting of S:*

- *w.r.t. to* μ_1 *is* $(r(x_1,x_2) \vee p(x_1)) \wedge t(x_1,x_3) \wedge r(x_3,x_4) \wedge s(x_1,x_5) \wedge s(x_3,x_5)$.
- *w.r.t. to* μ_2 *is* $r(x,x_2) \wedge t(x,x) \wedge r(x,x_4) \wedge q(x) \wedge h(x)$.
- *w.r.t. to* μ_3 *is* $p(x) \wedge r(x,x) \wedge s(x,x_5) \wedge s(x,x_5)$*, which is simplified to* $p(x) \wedge r(x,x) \wedge s(x,x_5)$.

Let us now focus on properties of piece-based rewriting for SCQs. Property 3 ensures that soundness is preserved while performing a rewriting.

Property 3 *Let* S *be a sound SCQ-rewriting of a CQ* Q*. If* μ *is a local (resp. non-local) unifier of* R *with* S*, then* $\gamma_L(S,R,\mu)$ *(resp.* $\gamma_{NL}(S,R,\mu)$*) is a sound* SCQ*-rewriting of* Q.

Performing an LU-saturation is relevant since applying a local unifier does not prevent any other unifier to be applied.

Property 4 *Let* S *be an SCQ,* R_1 *and* R_2 *be two rules,* μ_1 *be a local unifier of* R_1 *with* S *and* μ_2 *be a unifier (not necessarily local) of* R_2 *with* S. μ_2 *is a unifier of* R_2 *with* $\beta(S,R_1,\mu_1)$*. Moreover the (possible) locality of* μ_2 *is conserved.*

We now present the key properties that will ensure the completeness of our algorithm.

Property 5 *Let* $\mathcal{R}$ *be a set of rules,* S *be an SCQ, and* Q *a selection of* S*. Let* Q' *be a one step* $\mathcal{R}$*-rewriting of* Q*. Either* Q' *is less general than a selection of the LU-saturation of* S*, or there exists a one step rewriting* S' *of* S *s.t.* Q' *is less general than a selection of* S'.

The following property ensures that computing the cover at each step does present completeness.

Property 6 *Let* S *and* S' *be two SCQs such that* $S \geq S'$*. For any selection* Q'_r *of a one step-rewriting* S'_r *of* S_r*, either there exists a selection* Q *of LU-*saturation(S) *such that* $Q \geq Q'_r$*, or there exists a selection* Q_r *of a one step rewriting of LU-*saturation(S) *such that* $Q_r \geq Q'_r$.

COMPACT (Algorithm 2) performs a breadth-first exploration of the SCQ-rewritings of Q. For each SCQ S to be explored, S is first LU-saturated. Then, any prime unifier is used to generate new SCQs - which are in turn explored. To ensure that COMPACT halts, one should check that newly created SCQs are not less general than previously explored SCQs. This check is not trivial: in particular, we use the very specific structure of the rewriting generated by COMPACT in order to avoid to blow-up each SCQ, as would be suggested by Property 2.

Algorithm 1: LU-SATURATION

```
Data: A SCQ S, a set of existential rules R
Result: S saturated with respect to local unifications
S_o = null;
S_n = S;
while S_o ≠ S_n do
    S_o = S_n;
    for every rule R ∈ R do
        for every local unifier μ of R with S_n do
            S_n = γ_L(S_n, R, μ);
return S_n
```

Algorithm 2: COMPACT

Data: A CQ S (thus as SCQ), a *fus* $\mathcal{R}$
Result: A sound and complete USCQ-rewriting of S w.r.t. $\mathcal{R}$

```
S_F = {S};
S_E = {S};
while S_E ≠ ∅ do
    S_t = ∅;
    for every S' ∈ S_E do
        S' = LU-SATURATION(S');
        for every rule R ∈ R do
            for every prime unifier μ of R with S' do
                S_t = S_t ∪ {γ_NL(S', R, μ)}
    S_t = cover(S_F ∪ S_t);
    S_E = S_t \ S_F;
    S_F = S_t;
return S_F
```

Property 7 *Algorithm 2 outputs a sound and complete rewriting of S w.r.t. $\mathcal{R}$.*

Sketch of proof: Property 3 ensures that all generated queries are sound. An induction on the length of the smallest derivation generating a CQ-rewriting of Q based on Property 5 and Property 6 shows that for any CQ-rewriting of Q, there exists an SCQ S_Q generated by Algorithm 2 such that Q is less general than a selection of S_Q. □

Example 5 ((Cont.)Execution of COMPACT **on** $S, \mathcal{R}$**)** *We start from S and LU-saturate it. We obtain $S' =$ LU-Saturate$(S) = (r(x_1, x_2) \vee p(x_1)) \wedge t(x_1, x_3) \wedge (r(x_3, x_4) \vee p(x_3) \wedge s(x_1, x_5) \wedge s(x_3, x_5)$. The only prime unifier applicable to S' unifies R_2 with S', which is rewritten into $S'' = (r(x, x_2) \vee p(x)) \wedge t(x, x) \wedge (r(x, x_4) \vee p(x)) \wedge q(x)$. No new unifications are possible, and thus $\{S', S''\}$ is a sound and complete rewriting of S w.r.t. $\mathcal{R}$.*

The efficiency of the USCQ representation of sound and complete rewritings is striking when dealing with class or role hierarchies, which are covered by the following property.

Property 8 *Let $\mathcal{R}$ be a set of rules with no constants, no existential in the head, and such that no variable appear twice in the same atom. Let Q be a conjunctive CQ. The sound and complete SCQ-rewriting of Q w.r.t. $\mathcal{R}$ is a single SCQ.*

5 Experimental Evaluation

We now evaluate both steps of our query rewriting approach for OBDA. One the one hand, we want a rewriting algorithm that computes quickly sound and complete rewritings. On the other hand, we want these rewritings to be efficiently evaluable by current RDMS. For both steps, we use the ontologies introduced for benchmarking in [Pérez-Urbina *et al.*, 2009], which have been also used in [Gottlob *et al.*, 2011; Chortaras *et al.*, 2011; Imprialou *et al.*, 2012]. Moreover, since these ontologies are rather flat, we slightly modify the LUBM ontology[5], creating LUBM$_n$, by adding n subpredicates for each original predicate (e.g., Course has as sublcasses Course$_1$,. . ., Course$_n$). This process is very similar to what has been done in [Rodriguez-Muro and Calvanese, 2012] and [Lutz *et al.*, 2012]. As for the queries, we use two sets of queries for each ontology. First, the original handcrafted queries, which are only five. Then, we use the query generator Sygenia[6] [Imprialou *et al.*, 2012] in order to have a larger number of queries. For space reasons, we only present the results on the LUBM$_n$ ontologies and its modified version, which are representative of other experiments.

Rewriting. To test the rewriting step of COMPACT, we compare the rewritings obtained by COMPACT and by Iqaros, which has been shown to be faster than other tools on the considered benchmarks [Imprialou *et al.*, 2012]. For COMPACT, we present the number of output SCQs, the number of selections (i.e., the number of CQs that would be obtained by exploding each output SCQ), as well as the time needed to generate the USCQ. For Iqaros, we present the number of output CQs and the time required for computing them.

Querying. Since generating queries is only half of the story, we also test how efficiently the output queries can be evaluated against a database. The method is the following: given a CQ Q, let $\mathcal{Q}$ be the optimal UCQ-rewriting, and $\mathcal{S}$ be the USCQ output by COMPACT. We separately evaluate each CQ of $\mathcal{Q}$ by translating them into an SQL query, and compute the time required for evaluating all queries. We do the same thing for each SCQ of $\mathcal{S}$, where the translation involves the creation of views. The data we evaluate the query on is generated from the LUBM generator, with 20 universities (for a total of 556k unary atoms, and 2,2M binary atoms). A timeout has been set: all queries of the benchmark should have been answered within 30 minutes. COMPACT is implemented in Java. All tests have been performed on a 2.4GHz processor, with 4GB of RAM. The RDMS used is Sqlite.

5.1 Rewriting Step

Rewriting results are presented in Tables 1 and 2. For most queries, the USCQ rewriting output by COMPACT contains only one SCQ. On the benchmarks, the time needed for computing USCQs is better than the time needed for computing UCQs. The difference increases dramatically as the size of the UCQ rewritings increases, as witnessed by handcrafted queries with the LUBM$_n$ ontologies.

5.2 Querying Step

Figures 1 and 2 present the time, in seconds, needed to evaluate the optimal UCQ rewriting (black bars), and USCQ rewriting (white bars), for Sygenia-generated queries and for handcrafted queries, respectively. The ontologies are LUBM$_n$, for n from 0 to 8. Missing bars are timeouts.

USCQs are evaluated faster than their equivalent (optimal) UCQs. The difference grows as the size of the UCQ grows (with a fixed size of USCQ), which typically happens when class or role hierarchies are present.

[5] http://swat.cse.lehigh.edu/projects/lubm/

[6] http://code.google.com/p/sygenia/

Table 1: Rewriting time and output for Sygenia queries

	COMPACT			Iqaros	
	# SCQs	# Selections	Time (ms)	# CQ	Time (ms)
0	102	486	44	486	152
1	102	1203	56	1203	171
2	102	1910	75	1910	182
3	102	2691	68	2691	205
4	102	3546	86	3546	257
5	102	4475	108	4475	342
6	102	5478	144	5478	440
7	102	6555	173	6555	556
8	102	7706	217	7706	692

Table 2: Rewriting time and output for handcrafted queries

	COMPACT			Iqaros	
	# SCQs	# Selections	Time (ms)	# CQ	Time (ms)
0	5	19	68	19	200
1	5	102	73	102	247
2	5	360	143	360	460
3	5	972	213	972	1454
4	5	2190	303	2190	5242
5	5	4338	406	4338	17382
6	5	7812	530	7812	56095
7	5	13080	667	13080	155566
8	5	20682	823	20682	403229

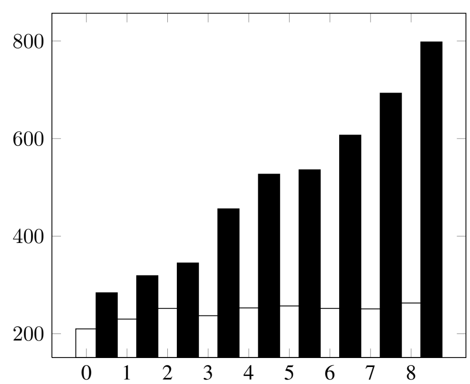

Figure 1: Querying time for Sygenia generated queries

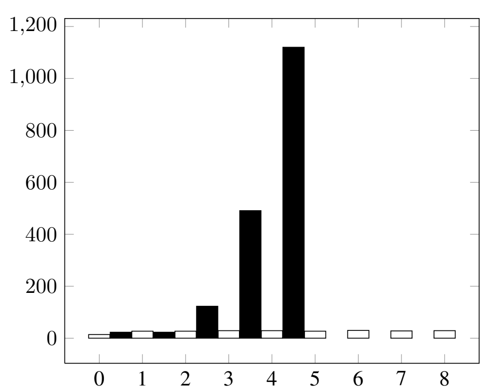

Figure 2: Querying time for handcrafted queries

6 Related Work

COMPACT has some similarities with two algorithms already published: Rapid and the piece-based rewriting algorithm. Incremental rewriting performed by Iqaros being further from our methods, we focus on these two.

First, the piece-based rewriting algorithm presented in [König *et al.*, 2012]: the distinguishing feature of this algorithm is that it computes optimal CQ-rewritings for any finite unification set. This generality - any finite unification set - is due to the use of piece unifiers. COMPACT is based upon the same mechanisms, modulo some generalizations in order to take disjunction into account.

Rapid performs sound and complete rewritings for OWL-QL ontologies. Rapid first performs a shrinking step, where all possible "structures" of CQ rewriting are generated at once. An important property of OWL-QL on which rapids is based on (in order to be time-efficient) is that such a shrinking step can be done only once. Then, every atom is separately rewritten using the ontology. Last, a distributivity step, including some compatibility checks, is performed. The separate rewriting for each atom can find a counter-part in COMPACT with the use of local unifiers, and creation of disjunctions. The distributivity step is not done - which allows for a dramatic time improvement when there are several large disjunctions in a single semi-conjunctive query.

7 Conclusion and Further Work

In this paper, we proposed a novel method to compute sound and complete rewritings of conjunctive queries with respect to finite unification sets of existential rules. Designing efficient tools for that task is important because it allows one to answer conjunctive queries against databases while taking ontologies into account without changing data. The distinguishing feature of our method is that it outputs a set of semi-conjunctive queries, which are a more general form of positive existential formulas than conjunctive queries. We advocated that such queries allow for a more compact representation of sound and complete rewritings that can be efficiently computed and evaluated. In particular, we presented a novel algorithm, COMPACT, that outperforms state-of-the-art algorithms on OWL-QL, while producing sound and complete rewritings for any finite unification set. First experiments showed that semi-conjunctive queries can be more efficiently evaluated than the equivalent union of conjunctive queries, especially when the size of the disjunctions involved is big enough - which happens even with queries and ontologies of moderate size.

Further work includes both practical and theoretical aspects. First, since existing benchmarks are limited, properly evaluating rewriting algorithms is hard. In particular, we evaluated COMPACT only on OWL-QL ontologies, whereas it is designed as a rewriting tool for any finite unification set. Then, an interesting improvement of COMPACT would be to take into account the possible completeness of a database with respect to some predicates. This method has been proven useful with some very light description logics [Rodriguez-Muro and Calvanese, 2012]. We believe such an approach could be successfully adapted to any finite unification set.

References

[Baader *et al.*, 2007] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[Baader, 2003] F. Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI*, pages 325–330, 2003.

[Baget *et al.*, 2009] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending Decidable Cases for Rules with Existential Variables. In *IJCAI*, pages 677–682, 2009.

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On Rules with Existential Variables: Walking the Decidability Line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[Calì *et al.*, 2008] A. Calì, G. Gottlob, and M. Kifer. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In *KR*, pages 70–80, 2008.

[Calì *et al.*, 2010] Andrea Calì, Georg Gottlob, and Andreas Pieris. Query rewriting under non-guarded rules. In *AMW*, 2010.

[Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[Chortaras *et al.*, 2011] Alexandros Chortaras, Despoina Trivela, and Giorgos B. Stamou. Optimized query rewriting for OWL 2 QL. In *CADE*, pages 192–206, 2011.

[Gottlob and Schwentick, 2012] Georg Gottlob and Thomas Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *KR*, 2012.

[Gottlob *et al.*, 2011] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: Rewriting and optimization. In *ICDE*, pages 2–13, 2011.

[Imprialou *et al.*, 2012] Martha Imprialou, Giorgos Stoilos, and Bernardo Cuenca Grau. Benchmarking ontology-based query rewriting systems. In *AAAI*, 2012.

[König *et al.*, 2012] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. A sound and complete backward chaining algorithm for existential rules. In *RR*, pages 122–138, 2012.

[Kontchakov *et al.*, 2011] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The Combined Approach to Ontology-Based Data Access. In *IJCAI*, pages 2656–2661, 2011.

[Lutz *et al.*, 2012] Carten Lutz, Inanç Seylan, David Toman, and Franck Wolter. The combined approach to OBDA: Taming role hierarchies using filters. In *SSWS+HPCSW*, 2012.

[Pérez-Urbina *et al.*, 2009] Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik. Efficient query answering for OWL 2. In *International Semantic Web Conference*, pages 489–504, 2009.

[Rodriguez-Muro and Calvanese, 2012] Mariano Rodriguez-Muro and Diego Calvanese. High performance query answering over DL-lite ontologies. In *KR*, 2012.

[Rosati and Almatelli, 2010] R. Rosati and A. Almatelli. Improving query answering over dl-lite ontologies. In *KR*, 2010.

[Salvat and Mugnier, 1996] Eric Salvat and Marie-Laure Mugnier. Sound and complete forward and backward chainingd of graph rules. In *ICCS*, pages 248–262, 1996.

[Stefanoni *et al.*, 2012] Giorgio Stefanoni, Boris Motik, and Ian Horrocks. Small datalog query rewritings for EL. In *Description Logics*, 2012.

[Thomazo *et al.*, 2012] Michaël Thomazo, Jean-François Baget, Marie-Laure Mugnier, and Sebastian Rudolph. A generic querying algorithm for greedy sets of existential rules. In *KR*, 2012.

[Venetis *et al.*, 2012] Tassos Venetis, Giorgos Stoilos, and Giorgos B. Stamou. Incremental query rewriting for OWL 2 QL. In *Description Logics*, 2012.