Sound, Complete, and Minimal Query Rewriting for Existential Rules

Mélanie König, Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo

Sound, Complete, and Minimal Query Rewriting for Existential Rules *

Mélanie König and **Michel Leclère** and **Marie-Laure Mugnier** and **Michaël Thomazo**
University of Montpellier, France
firstname.lastname@lirmm.fr

Abstract

We address the issue of Ontology-Based Data Access which consists of exploiting the semantics expressed in ontologies while querying data. Ontologies are represented in the framework of existential rules, also known as Datalog+/-. We focus on the backward chaining paradigm, which involves rewriting the query (assumed to be a conjunctive query, CQ) into a set of CQs (seen as a union of CQs). The proposed algorithm accepts any set of existential rules as input and stops for so-called finite unification sets of rules (fus). The rewriting step relies on a graph notion, called a piece, which allows to identify subsets of atoms from the query that must be processed together. We first show that our rewriting method computes a minimal set of CQs when this set is finite, i.e., the set of rules is a fus. We then focus on optimizing the rewriting step. First experiments are reported in the associated technical report.

1 Introduction

In recent years, there has been growing interest in exploiting the semantics expressed in ontologies when querying data, an issue known as ontology-based data access (OBDA). The dominant approach to this issue is based on description logics (DLs), with the most studied DLs in this context being lightweight DLs like DL-Lite and $\mathcal{EL}$ families [Baader, 2003; Calvanese *et al.*, 2007] and their Semantic Web counterparts, so-called tractable fragments of OWL2. A newer approach, to which this paper contributes, is based on existential rules. Existential rules have the ability of generating unknown individuals, a feature that has been recognized as crucial in an open-world perspective. These rules are of the form *body* $\rightarrow$ *head*, where *body* and *head* are conjunctions of atoms (without functions), and variables that occur only in *head* are *existentially* quantified [Baget *et al.*, 2009; 2010; 2011b; Krötzsch and Rudolph, 2011]. They are also known as Datalog$\pm$ an extension of plain Datalog to database constraints called tuple-generating dependencies [Calì *et al.*, 2008; 2009].

In this paper, we consider knowledge bases (KBs) composed of a set of facts, or data, and of existential rules. We focus on the standard basic queries, namely conjunctive queries (CQs), which can be seen as existentially quantified conjunctions of atoms. The fundamental decision problem associated with query answering can be expressed in several equivalent ways, in particular as a Boolean CQ entailment problem: is a given Boolean CQ logically entailed by a knowledge base?

CQ entailment is undecidable for general existential rules. There has been an intense research effort aimed at finding decidable subsets of rules that provide good tradeoffs between expressivity and complexity of query answering (see [Mugnier, 2011] for a synthesis). Compared to lightweight DLs, these decidable rule fragments are more powerful and flexible. However, the existential rule-based OBDA framework does not come yet with practically usable algorithms, with the exception of very simple classes of rules, which can be seen as slight generalizations of lightweight DLs. In this paper, we undertake a step in this direction.

There are two classical paradigms for processing rules, namely *forward chaining* and *backward chaining*, which can both be seen as ways of integrating the rules either into the facts or into the query: forward chaining uses the rules to enrich the facts and the query is entailed by the KB if it maps by homomorphism to the enriched facts, while backward chaining uses the rules to rewrite the query in several ways and the initial query is entailed by the KB if a rewritten query maps to the initial facts. In the context of large data, the obvious advantage of backward chaining is that it does not make the data grow. When the set of rewritten queries is finite, this set can be seen as a single query, which is the union of the queries in the set. An approach initiated with DL-Lite consists of decomposing backward chaining into two steps: (1) rewrite the initial CQ as a union of CQs (2) use a database management system to answer this rewritten query. This approach aims to benefit from the optimizations developed for classical database queries. It is at the core of several systems, such as QuOnto [Calvanese *et al.*, 2007], Requiem [Pérez-Urbina *et al.*, 2009], Nyaya [Gottlob *et al.*, 2011], Rapid [Chortaras *et al.*, 2011], Iqaros [Venetis *et al.*, 2012] and Quest [Rodriguez-Muro and Calvanese, 2012]. While the above work focuses on specific rule sublanguages, in this paper we consider *gen-*

*The paper on which this extended abstract is based was the recipient of the best paper award of the 2012 Web Reasoning and Rule Systems Conference (RR 2012) [König *et al.*, 2012b].

eral existential rules, i.e., our algorithm accepts as input any set of existential rules, but of course is guaranteed to stop only for a subset of them (called "finite unification sets" in [Baget *et al.*, 2010]), which includes expressive classes of rules.

The originality of our work lies in the rewriting step based on a notion stemming from a graph view of a set of atoms, that of a *piece*.[1] Briefly, a piece is a subset of atoms from the query that must be rewritten together. Classically, in logic programming, rules and queries are processed atom by atom: at each step, an atom a of a query Q is unified with the head of a rule R (a single atom) and a new query is generated by replacing a in Q by the body of R (precisely: let u be the unifier, the new query is $u(\mathit{body}(R)) \cup u(Q \setminus \{a\})$. Here, existential variables in rule heads have to be taken into account, which prevents the use of atomic unification. Instead, subsets of atoms (the pieces) have to be considered at once. We present below a very simple example.

Example 1 *Let the rule* $R = \forall x\ (q(x) \rightarrow \exists y\ p(x, y))$, *and the Boolean CQ* $Q = \exists u \exists v \exists w (p(u, v) \wedge p(w, v) \wedge r(u, w))$. *Assume we want to unify the atom* $p(u, v)$ *from* Q *with* $p(x, y)$ *by a substitution* $\{(u, x), (v, y)\}$. *Since* v *is unified with the existential variable* y, *all other atoms containing* v *must also be considered: indeed, simply rewriting* Q *into* $q(x) \wedge p(w, y) \wedge r(x, w)$ *would be incorrect: intuitively, the fact that the atoms* $p(u, v)$ *and* $p(w, v)$ *in* Q *share a variable would be lost in atoms* $q(x)$ *and* $p(w, y)$. *Thus,* $p(u, v)$ *and* $p(w, v)$ *have to be both unified with the head of* R *by means of the following substitution:* $\{(u, x), (v, y), (w, x)\}$. $\{p(u, v), p(w, v)\}$ *is called a piece. The corresponding rewriting of* Q *is* $q(x) \wedge r(x, x)$.

An alternative method would be to consider the Skolem form of rules, i.e., to replace existential variables in the head by Skolem functions of variables occurring in the body, however we think it is simpler and more intuitive to keep the original rule language. This framework established, we then posed ourselves the following questions:

1. Can we ensure that we produce a minimal set of rewritten conjunctive queries, in the sense that no sound and complete algorithm can produce a smaller set?
2. How to optimize the rewriting step? The problem of deciding whether there is a piece-unifier between a query and a rule head is NP-complete and the number of piece-unifiers can be exponential in the size of the query.

With respect to the first question, we first point out that any sound and complete set of CQs remains sound and complete when it is restricted to its most general elements (w.r.t. the generalization relation induced by homomorphism). We then show that all sound and complete sets of CQs restricted to their most general CQs have the same cardinality, which is minimal w.r.t. the completeness property.

With respect to the second question, we consider rules with an atomic head. This is not a restriction in terms of expressivity, since any rule can be decomposed into an equivalent set of atomic-head rules by simply introducing a new predicate for each rule [Calì *et al.*, 2008; Baget *et al.*, 2009]. Besides, many rules found in the literature have an atomic head. We exploit the fact that each atom in a CQ Q belongs to at most one piece with respect to a rule R (which is false for existential rules with non-atomic head) to efficiently compute a rewriting step, i.e., generate all CQs obtainable from R and Q. An algorithm producing a sound and complete minimal set of rewritten CQs, and benefiting from the above optimizations, has been implemented.

The paper is organized as follows. Section 2 introduces our framework. Sections 3 and 4 are respectively devoted to the first and to the second question. Finally, Section 5 reports first experiments and outlines further work. See [König *et al.*, 2012a] for the associated technical report with all proofs.

[1] Pieces come from earlier work on conceptual graph rules [Salvat and Mugnier, 1996] recast in the framework of existential rules in [Baget *et al.*, 2009; 2011a].

2 Framework

An *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate with arity k, and the t_i are terms, i.e., variables or constants. Given an atom or a set of atoms A, $\mathit{vars}(A)$, $\mathit{consts}(A)$ and $\mathit{terms}(A)$ denote its set of variables, of constants and of terms, respectively. In the following examples, all the terms are variables (denoted by x, y, z, etc.). $\models$ denotes the classical logical consequence. A *fact* is an existentially closed conjunction of atoms.[2] A *conjunctive query* (CQ) is an existentially quantified conjunction of atoms. When it is a closed formula, it is called a *Boolean* CQ (BCQ). Hence facts and BCQs have the same logical form. In the following, we will see them as sets of atoms. Given sets of atoms A and B, a *homomorphism* h from A to B is a substitution of $\mathit{vars}(A)$ by $\mathit{terms}(B)$ s.t. $h(A) \subseteq B$. We say that A *maps* to B by h. If there is a homomorphism from A to B, we say that A is *more general* than B (or B is *more specific* than A), which is denoted $A \geq B$ (or $B \leq A$). Given a fact F and a BCQ Q, the answer to Q in F is *positive* if $F \models Q$. It is well-known that $F \models Q$ iff there is a homomorphism from Q to F.

Definition 1 (Existential rule) *An* existential rule *(or simply* rule*) is a formula* $R = \forall \mathbf{x} \forall \mathbf{y} (B[\mathbf{x}, \mathbf{y}] \rightarrow \exists \mathbf{z} H[\mathbf{y}, \mathbf{z}])$ *where* $B = \mathit{body}(R)$ *and* $H = \mathit{head}(R)$ *are conjunctions of atoms, resp. called the* body *and the* head *of* R. *The* frontier *of* R, *noted* $\mathit{fr}(R)$, *is the set* $\mathit{vars}(B) \cap \mathit{vars}(H) = \mathbf{y}$.

A *knowledge base* (KB) $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts (seen as a single fact) F and a finite set of existential rules $\mathcal{R}$. The *BCQ entailment problem* takes as input a KB $\mathcal{K} = (F, \mathcal{R})$ and a BCQ Q, and asks if $F, \mathcal{R} \models Q$. **Other notations:** Throughout the paper we note respectively R and Q the considered rule and query. We always assume that R and Q have no variables in common. Given $Q' \subseteq Q$, we note $\bar{Q}'$ the set $Q \setminus Q'$. The variables in $\mathit{vars}(Q') \cap \mathit{vars}(\bar{Q}')$ are called *separating variables* and noted $\mathit{sep}(Q')$. In the examples, we will omit quantifiers in facts and rules since there is no ambiguity.

As explained in the introduction, the rewriting step relies on a specific unification operation based on "pieces".

A piece-unifier is a pair (Q', u), where $Q' \subseteq Q$ and u is a substitution that "unifies" Q' with some $H' \subseteq \mathit{head}(R)$,

[2] We generalize the classical notion of a fact in order to take existential variables into account.

in the sense that $u(Q') = u(H')$. The substitution u can be decomposed as follows: (1) it specializes $\mathit{fr}(R)$, thus $\mathit{head}(R) \geq u(\mathit{head}(R))$, with existential variables left unchanged; (2) it maps Q' to $u(\mathit{head}(R))$, while satisfying the following constraint: the separating variables in Q' are not mapped to existential variables.

Definition 2 (Piece-unifier) *A piece-unifier of Q with R is a pair $\mu = (Q', u)$ with $Q' \subseteq Q$, $Q' \neq \emptyset$, u a substitution of $\mathit{fr}(R) \cup \mathit{vars}(Q')$ by $\mathit{terms}(\mathit{head}(R)) \cup \mathit{consts}(Q' \cup \mathit{head}(R))$ s.t.:*

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

Example 2 *Let $R = q(x) \rightarrow p(x, y)$ and $Q = p(u, v) \wedge p(z, v) \wedge p(w, t) \wedge r(u, w)$. There are three ("most general", cf. Sect. 4) piece-unifiers of Q with R:*
• *$\mu_1 = (Q'_1, u_1)$ with $Q'_1 = \{p(u, v), p(z, v)\}$ and $u_1 = \{(u, x), (v, y), (z, x)\}$; we omit identity pairs in all examples, f.i. u_1 contains (x, x)*
• *$\mu_2 = (Q'_2, u_2)$ with $Q'_2 = \{p(w, t)\}$ and $u_2 = \{(w, x), (t, y)\}$*
• *$\mu_3 = (Q'_3, u_3)$ with $Q'_3 = \{p(u, v), p(z, v), p(w, t)\}$ and $u_3 = \{(u, x), (v, y), (z, x), (w, x), (t, y)\}$*

We are now able to formally define *pieces*. Generally speaking, a set of atoms can be partitioned into subsets (called pieces) w.r.t. a set T of variables acting as "cutpoints": two atoms are in the same piece if they are connected by a path of variables that are not in T. Here, T is the set of variables from Q' that are not mapped to existential variables by u.

Definition 3 (Piece) *[Baget* et al.*, 2011a] Let A be a set of atoms and $T \subseteq \mathit{vars}(A)$. A piece of A according to T is a minimal non-empty subset P of A s.t. for all a and a' in A, if $a \in P$ and $(\mathit{vars}(a) \cap \mathit{vars}(a')) \not\subseteq T$, then $a' \in P$.*

Definition 4 (Cutpoint, Piece of Q) *Given a piece-unifier $\mu = (Q', u)$ of Q with R, a variable $x \in Q'$ is a* cutpoint *if $u(x) \in \mathit{fr}(R) \cup \mathit{consts}(Q' \cup \mathit{head}(R))$. The set of cutpoints associated with μ is denoted by $T_Q(\mu)$. We call* piece *of Q (for μ) a piece of Q according to $T_Q(\mu)$.*

Example 2 (contd) *Q'_1 and Q'_2 are pieces for μ_1 and μ_2 respectively. Both are pieces for μ_3.*

In fact, for any piece-unifier $\mu = (Q', u)$, Q' is a set of pieces of Q, which justifies the name "piece-unifier". To sum up, a piece of Q is a minimal subset of atoms that must be considered together once cutpoints in Q have been defined (indeed, an atom of Q may belong to different pieces according to different piece-unifiers). Finally, note that in rules without existential variables, such as in plain Datalog, each piece is restricted to a single atom.

Definition 5 (Immediate Rewriting) *Given a piece-unifier $\mu = (Q', u)$ of Q with R, the* immediate rewriting *of Q according to μ, denoted $\beta(Q, R, \mu)$, is $u(\mathit{body}(R)) \cup u(\bar{Q'})$.*

Definition 6 ($\mathcal{R}$-rewriting of Q) *An* $\mathcal{R}$-rewriting *of Q is a CQ Q_k obtained by a finite sequence $(Q_0 = Q), Q_1, \ldots, Q_k$ s.t. for all $0 \leq i < k$, there is $R_i \in \mathcal{R}$ and a piece-unifier μ_i of Q_i with R_i s.t. $Q_{i+1} = \beta(Q_i, R_i, \mu_i)$.*

To evaluate the correctness of different rewriting mechanisms, we introduce the notions of soundness and completeness of a set of CQs with respect to Q and $\mathcal{R}$ (such a set is called a *rewriting set* hereafter):

Definition 7 (Sound and Complete (rewriting) set of CQs) *Let $\mathcal{R}$ be a set of existential rules and Q be a (B)CQ. Let $\mathcal{Q}$ be a set of CQs. $\mathcal{Q}$ is said to be* sound *w.r.t. Q and $\mathcal{R}$ if for all facts F, for all $Q_i \in \mathcal{Q}$, if Q_i maps to F then $\mathcal{R}, F \models Q$. Reciprocally, $\mathcal{Q}$ is said to be* complete *w.r.t. Q and $\mathcal{R}$ if for all fact F, if $\mathcal{R}, F \models Q$ then there is $Q_i \in \mathcal{Q}$ s.t. $Q_i \geq F$.*

3 Minimal Rewriting Sets

We first point out that only the *most general elements* of a rewriting set need to be considered. Indeed, let Q_1 and Q_2 be two elements of a rewriting set s.t. $Q_2 \leq Q_1$ and let F be any fact: if Q_1 maps to F, then Q_2 is useless; if Q_1 does not map to F, neither does Q_2; thus removing Q_2 will not undermine completeness (nor soundness). The output of a rewriting algorithm should thus be a minimal set of incomparable queries that "covers" all rewritings of the initial query:

Definition 8 (Cover) *Let $\mathcal{Q}$ be a set of BCQs. A* cover *of $\mathcal{Q}$ is a set of BCQs $\mathcal{Q}^c \subseteq \mathcal{Q}$ s.t.:*

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

It can be easily checked that all covers of $\mathcal{Q}$ have the same cardinality. Note that the set of rewritings of Q can have a finite cover even when it is infinite (Example 3).

Example 3 *Let $Q = t(u)$, $R = t(x) \wedge p(x, y) \rightarrow t(y)$. The set of $\mathcal{R}$-rewritings of Q with $\{R\}$ is infinite. The first generated queries are the following (note that rule variables are renamed when needed):*
$Q_0 = t(u)$
$Q_1 = t(x) \wedge p(x, y)$ *// from Q_0 and R with $\{(u, y)\}$*
$Q_2 = t(x_0) \wedge p(x_0, y_0) \wedge p(y_0, y)$ *// from Q_1 and R*
$Q_3 = t(x_1) \wedge p(x_1, y_1) \wedge p(y_1, y_0) \wedge p(y_0, y)$ and so on ...
However, the set of the most general $\mathcal{R}$-rewritings is $\{Q_0\}$ since any other obtainable query is more specific than Q_0.

A set of rules $\mathcal{R}$ for which it is ensured that the set of $\mathcal{R}$-rewritings of any query has a finite cover is called a finite unification set (*fus*). The *fus* property is not recognizable [Baget *et al.*, 2011a], but several recognizable *fus* classes have been exhibited in the literature [Baget *et al.*, 2009; Calì *et al.*, 2009; 2010]. Following Algorithm 1 is a breadth-first algorithm that, given a fus $\mathcal{R}$ and a query Q, generates a cover of the set of $\mathcal{R}$-rewritings of Q. "Exploring" a query consists of computing the set of immediate rewritings of this query with all rules. Initially, Q is the only query to explore; at each step (a while loop iteration), all queries generated at the preceding step and kept in the current cover are explored. The following lemma justifies the fact that only the most general rewritings are kept at each step of the algorithm.

Algorithm 1: REWRITING ALGORITHM

Data: A fus $\mathcal{R}$, a conjunctive query Q
Result: A cover of the set of $\mathcal{R}$-rewritings of Q
$\mathcal{Q}_F \leftarrow \{Q\}$; *// resulting set*
$\mathcal{Q}_E \leftarrow \{Q\}$; *// queries to be explored*
while $\mathcal{Q}_E \neq \emptyset$ **do**
    $\mathcal{Q}_t \leftarrow \emptyset$; *// queries generated at this rewriting step*
    for $Q_i \in \mathcal{Q}_E$ **do**
        for $R \in \mathcal{R}$ **do**
            for μ *piece-unifier of* Q_i *with* R **do**
                $\mathcal{Q}_t \leftarrow \mathcal{Q}_t \cup \beta(Q_i, R, \mu)$;
    $\mathcal{Q}^c \leftarrow$ ComputeCover$(\mathcal{Q}_F \cup \mathcal{Q}_t)$; *// update cover*
    $\mathcal{Q}_E \leftarrow \mathcal{Q}^c \backslash \mathcal{Q}_F$; *// select unexplored queries*
    $\mathcal{Q}_F \leftarrow \mathcal{Q}^c$;
return $\mathcal{Q}_F$

Lemma 1 *If $Q_1 \geq Q_2$ then for all piece-unifiers μ_2 of Q_2 with R: either (i) $Q_1 \geq \beta(Q_2, R, \mu_2)$ or (ii) there is a piece-unifier μ_1 of Q_1 with R such that $\beta(Q_1, R, \mu_1) \geq \beta(Q_2, R, \mu_2)$.*

Theorem 1 *Let $\mathcal{R}$ be a fus and $\mathcal{Q}$ be a sound and complete rewriting set of Q (with $\mathcal{R}$). Any cover of $\mathcal{Q}$ is of minimal cardinality among sound and complete rewriting sets of Q.*

From the previous observation, we conclude that any sound and complete rewriting algorithm can be "optimized" so that it outputs a set of rewritings of minimal cardinality. If we moreover delete redundant atoms from the obtained CQs, we obtain a *unique* sound and complete set of CQs that has both minimal cardinality and elements of minimal size (unicity is of course up to a bijective variable renaming).

4 Most General Single-Piece Unifiers

W.l.o.g. we now focus on rules with atomic head. What is simpler with these rules? The definition of a piece-unifier in itself does not change. The difference lies in the number of piece-unifiers to be considered at a rewriting step. We first notice that we can restrict our focus to *most general single-piece* unifiers: the number of such unifiers of Q with R is bounded by $|Q|$, since there is a unique way of associating any atom in Q with $head(R)$.

Let $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ be two piece-unifiers of Q with R, defined on the *same* subset Q' of Q. μ_1 is said to be *more general* than μ_2, noted $\mu_1 \geq \mu_2$, if u_1 is more general than u_2 (i.e., there is a substitution s s.t. $u_2 = s \circ u_1$).

Property 1 *Let $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ be piece-unifiers s.t. $\mu_1 \geq \mu_2$. Then $\beta(Q, R, \mu_1) \geq \beta(Q, R, \mu_2)$.*

A piece-unifier $\mu = (Q', u)$ of Q with R is said to be *single-piece* if Q' is a piece of Q. Any piece-unifier can be decomposed into single-piece unifiers. Note however that applying successively each of these underlying single-piece unifiers may lead to a CQ strictly more general than $\beta(Q, R, \mu)$, as illustrated in the next example:

Example 4 *Let $R = p(x, y) \rightarrow q(x, y)$ and $Q = q(u, v) \wedge r(v, w) \wedge q(t, w)$. Let $\mu = (Q', u)$ be a piece-unifier of Q with R with $Q' = \{q(u, v), q(t, w)\}$ and $u = \{(u, x), (v, y), (t, x), (w, y)\}$. $\beta(Q, R, \mu) = p(x, y) \wedge r(y, y)$. Q' has two pieces w.r.t. μ: $P_1 = \{q(u, v)\}$ and $P_2 = \{q(t, w)\}$. If we successively apply the underlying single-piece unifiers μ_{P_1} and μ_{P_2}, we obtain $\beta(\beta(Q, R, \mu_{P_1}), R, \mu_{P_2}) = \beta(p(x, y) \wedge r(y, w) \wedge q(t, w), R, \mu_{P_2}) = p(x, y) \wedge r(y, y') \wedge p(x', y') \lneq \beta(Q, R, \mu)$.*

Property 2 *For any piece-unifier μ of Q with R, there is Q^s an $\{R\}$-rewriting of Q obtained by considering exclusively most general single-piece unifiers s.t. $Q^s \geq \beta(Q, R, \mu)$.*

From Lemma 1 and Property 2, we obtain:

Theorem 2 *Given a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively most general single-piece unifiers is sound and complete.*

However, single-piece unifiers cannot be used as such in Algorithm 1. The next example shows that, despite the completeness result of Theorem 2, the restriction to single-piece unifiers is not compatible with selecting most general rewritings at each step, as done in Algorithm 1.

Example 5 *Let $Q = p(y, z) \wedge p(z, y)$ and $R = r(x, x) \rightarrow p(x, x)$. There are two single-piece unifiers of Q with R, $\mu_1 = (p(y, z), u)$ and $\mu_2 = (p(z, y), u)$ with $u = \{(y, x), (z, x)\}$, which yield the same rewriting $Q_1 = r(x, x) \wedge p(x, x)$. There is also a two-piece unifier $\mu = (Q, u)$, which yields $Q' = r(x, x)$. A query equivalent to Q' can be obtained from Q_1 by a further single-piece unification. Now, assume that we restrict unifiers to single-piece unifiers* and *keep most general rewritings at each step. Since $Q \geq Q_1$, Q_1 is not kept, so Q' will never be generated, whereas it is incomparable with Q.*

To keep the correctness of Algorithm 1, we have to combine single-piece unifiers when they are *compatible*: two piece-unifiers of Q with R, $\mu_1 = (Q'_1, u_1)$ and $\mu_2 = (Q'_2, u_2)$, are said to be compatible if (1) $Q'_1 \cap Q'_2 = \emptyset$ (2) for all $x \in vars(Q'_1) \cap vars(Q'_2)$, when $u_1(x)$ and $u_2(x)$ are both constants, it holds that $u_1(x) = u_2(x)$. To sum up, we keep the schema of Algorithm 1 but, instead of computing all the piece-unifiers at a given step, we compute the single-piece unifiers, then apply together the compatible ones.

5 Perspectives

First experiments were led with the same benchmark as [Gottlob *et al.*, 2011], then extended using the query generator from [Imprialou *et al.*, 2012]. We compared our algorithm to Nyaya/NY* rewriting engine [Gottlob *et al.*, 2011]. Both running times were comparable, however we found that NY* did not output minimal rewriting sets (although its output was already shown smaller than the output of other existing systems). Since the benchmark we used consists of very simple ontologies we need to consider larger and more complex rule bases. We also have to compare to other recent rewriting systems, though these systems deal with more restricted classes of rules. Finally, our rewriting mechanism is yet far from being optimized. For instance, the number of explored queries is still very large w.r.t. the size of the final cover. The question of whether it is worthwhile, when rules do not have atomic heads, to deal directly with them, still needs to be addressed.

References

[Baader, 2003] F. Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI'03*, 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'09*, pages 677–682, 2009.

[Baget *et al.*, 2010] J.-F. Baget, M. Leclère, and M.-L. Mugnier. Walking the decidability line for rules with existential variables. In *KR'10*, pages 466–476. AAAI Press, 2010.

[Baget *et al.*, 2011a] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.

[Baget *et al.*, 2011b] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI'11*, pages 712–717, 2011.

[Calì *et al.*, 2008] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *KR'08*, pages 70–80, 2008.

[Calì *et al.*, 2009] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS'09*, pages 77–86, 2009.

[Calì *et al.*, 2010] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in datalog+/-. In *RR'10*, pages 1–17, 2010.

[Calvanese *et al.*, 2007] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

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

[Gottlob *et al.*, 2011] G. Gottlob, G. Orsi, and A. Pieris. Ontological queries: Rewriting and optimization. In *ICDE'11*, pages 2–13, 2011.

[Imprialou *et al.*, 2012] M. Imprialou, G. Stoilos, and B. Cuenca Grau. Benchmarking ontology-based query rewriting systems. In *AAAI*, 2012.

[König *et al.*, 2012a] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. A Sound and Complete Backward Chaining Algorithm for Existential Rules. Technical Report RR-12016, LIRMM, GraphIK - INRIA Sophia Antipolis, 2012.

[König *et al.*, 2012b] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. A sound and complete backward chaining algorithm for existential rules. In M. Krötzsch and U. Straccia, editors, *RR*, volume 7497 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2012.

[Krötzsch and Rudolph, 2011] M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI'11*, pages 963–968, 2011.

[Mugnier, 2011] M.-L. Mugnier. Ontological Query Answering with Existential Rules. In *RR'11*, pages 2–23, 2011.

[Pérez-Urbina *et al.*, 2009] H. Pérez-Urbina, I. Horrocks, and B. Motik. Efficient query answering for owl 2. In *ISWC'09*, pages 489–504, 2009.

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

[Salvat and Mugnier, 1996] E. Salvat and M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *ICCS'96*, volume 1115 of *LNAI*, pages 248–262. Springer, 1996.

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