Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Query Rewriting for Existential Rules with Compiled Preorder

Mélanie König, Michel Leclère, Marie-Laure Mugnier
University of Montpellier, Inria, CNRS
Montpellier, France

Abstract

We address the issue of Ontology-Based Query Answering (OBQA), which seeks to exploit knowledge expressed in ontologies when querying data. Ontologies are represented in the framework of existential rules (aka Datalog±). A commonly used technique consists in rewriting queries into unions of conjunctive queries (UCQs). However, the obtained queries can be prohibitively large in practice. A well-known source of combinatorial explosion are very simple rules, typically expressing taxonomies and relation signatures. We propose a rewriting technique, which consists in compiling these rules into a preorder on atoms and embedding this preorder into the rewriting process. This allows to compute compact rewritings that can be considered as "pivotal" representations, in the sense that they can be evaluated by different kinds of database systems. The provided algorithm computes a sound, complete and minimal UCQ rewriting, if one exists. Experiments show that this technique leads to substantial gains, in terms of size and runtime, and scales on very large ontologies. We also compare to other tools for OBQA with existential rules and related lightweight description logics.

1 Introduction

We address the issue of Ontology-Based Query Answering (OBQA), which seeks to exploit knowledge expressed in ontologies when querying data. We consider the novel framework of existential rules, also called Datalog± [Baget *et al.*, 2011; Calì *et al.*, 2012; Krötzsch and Rudolph, 2011]. Existential rules are an exstension of function-free positive conjunctive rules that allows for *existentially* quantified variables in rule heads. These rules are able to assert the existence of unknown entities, a fundamental feature in an open-domain perspective, where it cannot be assumed that the description of data is complete. Interestingly, existential rules generalize Horn description logics (DLs), in particular lightweight DLs used in the context of OBQA, such as DL-Lite [Calvanese *et al.*, 2005] and $\mathcal{EL}$ [Baader, 2003], which form the core of so-called tractable profiles of the Semantic Web ontological language OWL 2. They overcome some limitations of these DLs by allowing for non-tree structures as well as unbounded predicate arity. We consider here the basic database queries, i.e., conjunctive queries (CQs).

There are two main approaches to query answering in presence of existential rules and Horn DLs. The first approach is related to forward chaining: it triggers the rules to build a finite representation of inferred data such that the answers can be computed by evaluating the query against this representation, e.g., [Calì *et al.*, 2008; Thomazo *et al.*, 2012]. The second approach, initiated by DL-Lite [Calvanese *et al.*, 2005], is related to backward chaining: it rewrites the query using the rules such that the answers can be computed by evaluating the rewritten query against the data, e.g., [Baget *et al.*, 2009; Gottlob *et al.*, 2011] on existential rules. Some techniques combine both approaches [Kontchakov *et al.*, 2011; Thomazo and Rudolph, 2014]. Each technique is applicable to a specific existential rule fragment.

In this paper, we focus on the query rewriting approach, which has the advantage of being independent from the data. This technique typically outputs a union of conjunctive queries (UCQ), with the aim of benefiting from optimized relational database management systems (RDBMS). However, despite the good theoretical data complexity, first experiments exhibited a serious problem related to the size of the rewritten query, e.g., [Rosati and Almatelli, 2010]. Indeed, the output query can be exponentially larger than the initial query, even with very simple ontologies, and even if the output is extended to arbitrary first-order queries [Kikot *et al.*, 2011; 2012]. This led to a significant amount of work on rewriting into more compact queries, which may remain first-order queries (hence expressible in SQL) or go beyond them (like Datalog programs, e.g., [Gottlob and Schwentick, 2012; Stefanoni *et al.*, 2012]). Nowadays, mature systems have emerged for OWL 2 QL [Rodriguez-Muro *et al.*, 2013; Civili *et al.*, 2013], as well as prototypes for the $\mathcal{EL}$ family and more expressive DLs, e.g., [Eiter *et al.*, 2012]. Existential rules are more complex to process. Entailment with general existential rules is even undecidable (e.g., [Beeri and Vardi, 1981]). However, expressive subclasses ensuring the existence of a UCQ rewriting are known, such as *linear* rules, which generalize most DL-Lite dialects, the *sticky* family, classes satisfying conditions expressed on a graph of *rule dependencies*, and *weakly recursive* rules [Calì *et al.*, 2009; 2012; Baget *et al.*, 2011; Civili and Rosati, 2012].

A well-known source of combinatorial explosion are some very simple rules that typically express taxonomies or relation signatures. These rules are at the core of any ontology. We propose a new technique to tame this source of complexity. It relies on compiling these rules into a preorder on atoms and embedding this preorder *into* the rewriting process. Intuitively, each atom "represents" all its "specializations". Hence, the query rewriting process avoids exploring many CQs and outputs a small UCQ. This UCQ can be seen as a "pivotal" representation, in the sense that it can be evaluated by different kinds of systems: it can be passed to a Datalog engine or unfolded into a positive existential query (a UCQ or a more compact form of query) to be processed by an RDBMS; it can also be directly processed if data is stored in main memory and the appropriate homomorphism operation is implemented.

First, we prove that the new rewriting operator is sound and complete, and provide an algorithm that, given any CQ and any set of existential rules, effectively outputs a minimal, sound and complete UCQ-rewriting, if one exists (which may not be the case due to the indecidability of the problem). Since the computation of the preorder is independent from any query, we can divide this algorithm into a compilation step performed offline, and a query rewriting step, which takes the preorder as input. Second, we report experiments, which show that this optimization leads to substantial gains, in terms of size of the output and query rewriting runtime, and is able to scale on very large ontologies. In the case when the desired output is a "regular" UCQ, it remains faster for our rewriting tool to compute a pivotal UCQ and to unfold it, than to directly compute a regular UCQ. Moreover, our prototype behaves well compared to other UCQ-rewriting tools tailored for DL-Lite ontologies, even if it does not exploit the specificities of these languages.

2 Preliminaries

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 (we do not consider other function symbols). 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.) unless otherwise specified. $\models$ denotes the classical logical consequence. Given sets of atoms A and B, a *homomorphism* h from A to B is a substitution of $\mathit{vars}(A)$ by $\mathit{terms}(B)$ such that $h(A) \subseteq B$. It is convenient to extend the domain of a substitution h to a set of terms, such that $h(t) = t$ for all unchanged terms (in particular constants). If there is a homomorphism h from A to B, we say that *A maps* to B (by h); then, A is said to be *more general* than B, and B *more specific* than A.

A *fact* is the existential closure of a conjunction of atoms (this generalization of the classical notion of a fact allows to take existential variables into account). A *conjunctive query* is an existentially quantified conjunction of atoms. W.l.o.g. we consider only *Boolean* conjunctive queries (i.e., closed formulas) in this paper and denote them by CQs. In the following, we will see facts and CQs as sets of atoms. The answer to a CQ Q in a fact F is *yes* if $F \models Q$; it is well-known that $F \models Q$ iff Q maps to F.

Definition 1 (Existential rule) *An* existential rule *(or simply* rule *hereafter) 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 is the set of variables $\mathit{vars}(B) \cap \mathit{vars}(H) = \mathbf{y}$. The* existential variables *in R is the set of variables $\mathit{vars}(H) \setminus \mathit{vars}(B) = \mathbf{z}$.*

In the following, we will omit quantifiers in rules as there is no ambiguity. E.g. $p(x, y) \rightarrow p(y, z)$ denotes the formula $\forall x \forall y (p(x, y) \rightarrow \exists z p(y, z))$.

A *knowledge base* (KB) $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts (which can be seen as a single fact) F and a finite set of existential rules $\mathcal{R}$. The CQ entailment problem is the following: given a KB $\mathcal{K} = (F, \mathcal{R})$ and a CQ Q, does $F, \mathcal{R} \models Q$ hold? This problem is known to be undecidable in the general case [Beeri and Vardi, 1981]. It can be solved by computing a sound and complete rewritten query $\mathcal{Q}$ from Q and $\mathcal{R}$, provided that such a finite $\mathcal{Q}$ exists, then asking if $F \models \mathcal{Q}$. Here, the target query $\mathcal{Q}$ is a union of CQs (UCQ), that we see as a set of CQs, called a *rewriting set* of Q.

Definition 2 *[Sound and Complete (rewriting) set of CQs] Let $\mathcal{R}$ be a set of existential rules, Q be a CQ, and $\mathcal{Q}$ be a set of CQs. $\mathcal{Q}$ is said to be* sound *w.r.t. Q and $\mathcal{R}$ if for any fact 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 any fact F, if $\mathcal{R}, F \models Q$ then there is $Q_i \in \mathcal{Q}$ such that Q_i maps to F.*

A set of rules $\mathcal{R}$ for which any query Q has a finite sound and complete set of rewritings (in other words, Q is rewritable into a UCQ), is called a *finite unification set* (*fus*) [Baget *et al.*, 2009]. Note that restricting a complete rewriting set $\mathcal{Q}$ to its most general elements preserves its completeness. A *cover* of $\mathcal{Q}$ is an inclusion-minimal subset $\mathcal{Q}'$ of $\mathcal{Q}$ such that each element Q of $\mathcal{Q}$ is more specific than an element Q' of $\mathcal{Q}'$ (i.e., in database terms: Q is contained in Q'). If a set has a finite cover, then all its covers have the same size, hence all minimal, sound and complete rewriting sets have the same cardinality [König *et al.*, 2012]. Hereafter, we, respectively, denote by $\mathcal{R}$, R and Q the considered set of rules, a rule of $\mathcal{R}$, and the initial query. We assume that rules have disjoint sets of variables, as well as Q and R.

The rewriting operation involves unifying part of Q and part of $\mathit{head}(R)$. Existential variables in rule heads induce a possibly complex structure, therefore unification has to consider subsets of atoms at once (called "pieces") instead of single atoms, hence the name *piece-unifier*. Indeed, if a variable x from Q is unified with an existential variable from $\mathit{head}(R)$, then all atoms in which x occurs must be part of the unification, otherwise the obtained rewriting is unsound. Given $Q' \subseteq Q$, we call *separating variables* of Q', the variables occurring in both Q' and $(Q \setminus Q')$; the other variables from Q' are called non-separating. Let Q' be the unified part of Q: only non-separating variables from Q' can be unified with an existential variable of the rule.

Definition 3 (Piece-Unifier [König *et al.*, 2012]) *A* piece-unifier *of Q with R is a triple $\mu = (Q', H', u)$, where $Q' \neq \emptyset$, $Q' \subseteq Q$, $H' \subseteq \mathsf{head}(R)$ and u is a substitution of $T = \mathsf{terms}(Q') \cup \mathsf{terms}(H')$ by T, such that:*

1. *$u(H') = u(Q')$;*
2. *for all existential variable $x \in \mathsf{vars}(H')$ and $t \in T$, with $t \neq x$, if $u(x) = u(t)$, then t is a non-separating variable from Q'.*

Example 1 (Piece-unification) *Let $R = twin(x, y) \rightarrow motherOf(z, x) \wedge motherOf(z, y)$, where z is an existential variable, expressing that twins have a common mother. Let $Q_{no} = \{motherOf(v, w), painter(v)\}$, asking if there is a mother who is a painter. The only candidate atom in Q_{no} is $motherOf(v, w)$: then, v would be unified with z, hence $painter(v)$ should be unified as well, which is impossible. Let $Q_{yes} = \{motherOf(v, w), motherOf(v, t), female(w), male(t)\}$ asking if there is a mother of a female and a male. A piece-unifier of Q_{yes} with R is $\mu = (Q' = \{motherOf(v, w), motherOf(v, t)\}, H' = \{motherOf(z, x), motherOf(z, y)\}, u = \{z \mapsto v, x \mapsto w, y \mapsto t\})$ (in the general case Q' and H' may be non-isomorphic). The one-step rewriting of Q_{yes} according to μ is $\{twin(w, t), female(w), male(t)\}$, as formally defined below.*

Definition 4 (One-step Rewriting, $\mathcal{R}$-rewriting, β^*) *Given a piece-unifier $\mu = (Q', H', u)$ of Q with R, the* one-step rewriting *of Q according to μ, denoted by $\beta(Q, R, \mu)$, is the CQ $u(\mathsf{body}(R)) \cup u(Q \setminus Q')$. An* $\mathcal{R}$-rewriting *of Q is a CQ Q_k obtained by a finite sequence $(Q_0 = Q), Q_1, \ldots, Q_k$ 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 $Q_{i+1} = \beta(Q_i, R_i, \mu)$. We denote by $\beta^*(Q, \mathcal{R})$ the set of all $\mathcal{R}$-rewritings of Q.*

It is known that $\beta^*(Q, \mathcal{R})$ is a sound and complete set (within the meaning of Def. 2) [Baget *et al.*, 2011].

3 Embedding a Preorder on Atoms

We first explain the ideas underlying the rewriting technique before providing formal definitions. An essential component of any ontology is a hierarchy of concepts and, to a lesser extent, a hierarchy of relations (binary relations are also called roles or properties). In a logical framework, concepts and relations are represented by predicates. Simple rules of the form $p(x_1, \ldots, x_k) \rightarrow q(x_1, \ldots, x_k)$, where $k = 1$ for a concept, express that p is more specific than q (notation $p \leq q$). See e.g., the logical translation of atomic inclusions in DL, and the *subClassOf* and *subPropertyOf* assertions in RDFS / OWL. These rules, called *hierarchical* hereafter, are an obvious cause of combinatorial explosion in query rewriting, as illustrated by the next example.

Example 2 *Let $R_1 \ldots R_n$ be rules of the form $R_i : b_i(x) \rightarrow b_{i-1}(x)$. Let $Q = \{b_0(x_1), \ldots, b_0(x_k)\}$. Each atom $b_0(x_j)$ in Q is rewritten into $b_1(x_j)$, which in turn is rewritten into $b_2(x_j)$, and so on. Thus, there are $(n + 1)^k$ rewritings of Q.*

Hierarchical rules can be compiled into a (partial) preorder (i.e., a reflexive and transitive relation) on predicates, say $\leq$. Then, the homomorphism notion can be extended to take this preorder into account. Let us call $\leq$-homomorphism h from a set of atoms A_1 to a set of atoms A_2 a substitution of $\mathsf{terms}(A_1)$ by $\mathsf{terms}(A_2)$ such that for all $q(e_1, \ldots, e_k) \in A_1$, there is $p(h(e_1), \ldots, h(e_k)) \in A_2$ with $p \leq q$. It is easily checked that, given a set of hierarchical rules $\mathcal{R}_c$, it holds that $\mathcal{R}_c, F \models Q$ if and only if there is a $\leq$-homomorphism from Q to F. Now, let $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_e$ be a set of existential rules, where $\mathcal{R}_c$ is composed of the hierarchical rules. Given the preorder $\leq$ associated with $\mathcal{R}_c$, we would like to have $F, \mathcal{R} \models Q$ if and only if there is an $\mathcal{R}_e$-rewriting of Q that maps to F by a $\leq$-homomorphism. In Ex. 2, we would have $b_j \leq b_i$ for any $i \leq j$; then, since all rules are hierarchical, we would get a single rewriting, i.e., Q itself, instead of exponentially many. However, to achieve completeness, we cannot simply rewrite Q with $\mathcal{R}_e$ and forget about $\mathcal{R}_c$, we have to embed the preorder into the rewriting process. This idea can be further extended to compile all rules with an atomic body, as long as they do not introduce existential variables. This allows to compile other common axioms, as illustrated by Ex. 3. However, since the atoms in a rule may have predicates of different arity and arguments in different positions, we cannot rely on a simple preorder on predicates anymore. We have to embed a preorder on *atoms*.

Example 3 *Let $\mathcal{R}_{ex}$ be the following set of rules:*
$R_1 = r(x, y) \rightarrow t(x, y)$ (r is a specialization of t)
$R_2 = s(x, y) \rightarrow t(y, x)$
$R'_2 = t(x, y) \rightarrow s(y, x)$ (s and t are inverse relations)
$R_3 = t(x, y) \rightarrow q(x)$ (q is the domain of t)
$R_4 = t(x, y) \rightarrow q(y)$ (q is the range of t)
$R_5 = p(x, y, z) \rightarrow r(x, z)$ (r is a projection of p)
$R_6 = p(x, x, z) \rightarrow s(x, x)$ (introduction of a "self loop")

A rule is said to be *compilable* if it has a single body atom, no existential variable and no constant.[1] W.l.o.g. we also assume that a compilable rule has a single head.

Definition 5 (Inferred Rule, Saturation) *Let R_1 and R_2 be compilable rules such that $\mathsf{head}(R_1)$ and $\mathsf{body}(R_2)$ are unifiable by a (classical) most general unifier u. The rule* inferred *from (R_1, R_2) is $R_1 \bullet R_2 = u(\mathsf{body}(R_1)) \rightarrow u(\mathsf{head}(R_2))$. Given a set $\mathcal{R}_c$ of compilable rules, the* saturation *of $\mathcal{R}_c$, denoted by $\mathcal{R}_c^*$, is the closure of $\mathcal{R}_c$ by the $\bullet$ operation.*

Example 4 *The rules inferred from $\mathcal{R}_{ex}$ (Ex. 3) are the following; we recall that rules have disjoint sets of variables, even if we use the same variables for simplicity:*
$R_1 \bullet R'_2 = r(x, y) \rightarrow s(y, x)$; $R_1 \bullet R_3 = r(x, y) \rightarrow q(x)$
$R_1 \bullet R_4 = r(x, y) \rightarrow q(y)$; $R_2 \bullet R'_2 = s(x, y) \rightarrow s(x, y)$
$R_2 \bullet R_3 = s(x, y) \rightarrow q(y)$; $R_2 \bullet R_4 = s(x, y) \rightarrow q(x)$
$R'_2 \bullet R_2 = t(x, y) \rightarrow t(x, y)$; $R_5 \bullet R_1 = p(x, y, z) \rightarrow t(x, z)$
$R_6 \bullet R_2 = p(x, x, z) \rightarrow t(x, x)$
$R_5 \bullet R_1 \bullet R'_2 = p(x, y, z) \rightarrow s(z, x)$
$R_5 \bullet R_1 \bullet R_3 = p(x, y, z) \rightarrow q(x)$
$R_5 \bullet R_1 \bullet R_4 = p(x, y, z) \rightarrow q(z)$
$R_6 \bullet R_2 \bullet R_3 = R_6 \bullet R_2 \bullet R_4 = p(x, x, z) \rightarrow q(x)$

The size of $\mathcal{R}_c^*$ is polynomial in the size of $\mathcal{R}_c$ when the predicate arity is bounded (more specifically, when the number of occurrences of a variable in a predicate is bounded):

[1]The condition on constants is to simplify definitions.

Proposition 1 *$|\mathcal{R}_c^*|$ is bounded by $|\mathcal{R}_c|^2 \times B_k$, where k is the maximum predicate arity and B_k is the k-th Bell number.*

Proof: The number of possible specializations of an atom with arity k is bounded by the number of partitions of $\{1, \ldots, k\}$, known as the k-th Bell number B_k. Thus, the number of distinct rules (up to variable renaming) that can be obtained is bounded by $|\mathcal{R}_c|^2 \times B_k$. Note that $B_k < 2^{k^2}$. $\square$

The saturation of $\mathcal{R}_c$ can easily be made minimal by removing tautological rules (necessarily of the form $H \rightarrow H$) and redundant rules, i.e., rules entailed by other rules. Given non-tautological compilable rules R_i and R_j, $R_j \models R_i$ iff R_i *subsumes* R_j, as defined below.

Definition 6 (Rule Subsumption) *Let R_i and R_j be compilable rules. We say that R_i* subsumes *R_j if there is a homomorphism h from $\mathsf{body}(R_i)$ to $\mathsf{body}(R_j)$ such that $h(\mathsf{head}(R_i)) = \mathsf{head}(R_j)$.*

Example 3 ***(cont'd)*** $R_2 \bullet R_2'$ and $R_2' \bullet R_2$ are tautological rules. $R_6 \bullet R_2 \bullet R_3 = p(x,x,z) \rightarrow q(x)$ is subsumed by $R_5 \bullet R_1 \bullet R_3 = p(x,y,z) \rightarrow q(x)$.

We first define a preorder $\preccurlyeq$ on atoms (and sets of atoms) that will be used to avoid query rewriting with $\mathcal{R}_c$.

Definition 7 ($\preccurlyeq$) *Let* a *and* b *be atoms. We note* a $\preccurlyeq$ b *if (i)* a = b *or (ii) there is a rule $R \in \mathcal{R}_c^*$ that subsumes the rule (*a $\rightarrow$ b*) (equivalently: the application of R to* a *yields exactly* b*). Let A and B be sets of atoms. We note $A \preccurlyeq B$ if there is a surjective mapping f from B to A such that for all* b $\in B$, f(b) $\preccurlyeq$ b.

In the above definition, note that $a \preccurlyeq b$ implies $\mathsf{terms}(b) \subseteq \mathsf{terms}(a)$, and the same holds for A and B; moreover, if $a \preccurlyeq b$ and $a \neq b$, then the one-step rewriting of b with R may be strictly more general than a, as shown in Ex. 5.

Example 5 *Let $R = r(x,y) \rightarrow q(x)$,* a $= r(c_1,c_2)$ *and* b $= q(c_1)$. *The application of R to* a *yields* b, *hence* a $\preccurlyeq$ b. *The one-step rewriting of* b *with R is not* a *but $r(c_1, y)$.*

More generally, considering sets of atoms:

Property 1 *Let A and B be sets of atoms. It holds that $A \preccurlyeq B$ iff there is an $\mathcal{R}_c$-rewriting B' of B that maps to A by a substitution s of $\mathsf{vars}(B') \setminus \mathsf{vars}(B)$ by $\mathsf{terms}(A)$ such that $s(B') = A$.*

Example 6 *Let $\mathcal{R}_{ex}^*$ from Ex. 3 and 4. Let $A = \{p(u,u,c_1), r(c_2,c_1)\}$ and $B = \{s(u,u), s(c_1,u), t(c_2,c_1), q(c_2)\}$, where c_1 and c_2 are constants. One has $A \preccurlyeq B$ since $p(u,u,c_1) \rightarrow s(u,u)$ is subsumed by R_6, $p(u,u,c_1) \rightarrow s(c_1,u)$ is subsumed by $R_5 \bullet R_1 \bullet R_2'$, $r(c_2,c_1) \rightarrow t(c_2,c_1)$ is subsumed by R_1 and $r(c_2,c_1) \rightarrow q(c_2)$ is subsumed by $R_1 \bullet R_3$. According to Prop. 1, one can also check that $B' = \{p(u,u,z), p(u,y',c_1), r(c_2,c_1), r(c_2,y)\}$ is an $\mathcal{R}_{ex}$-rewriting of B (by successively using rules R_6, R_5, R_1, R_2', R_1, R_1, R_3) and that B' maps to A.*

Thanks to the preorder $\preccurlyeq$, we are now able to embed compiled rules in piece-unifiers.

Definition 8 ($\preccurlyeq$-Piece-Unifier, $\preccurlyeq$-rewriting, $\beta_\preccurlyeq$) *Given a preorder $\preccurlyeq$ on atoms, a $\preccurlyeq$-piece-unifier of Q with R is a triple $\mu = (Q', H', u)$ defined similarly to a piece-unifier (Def. 3), with Condition 1 replaced by $u(H') \preccurlyeq u(Q')$. The one-step $\preccurlyeq$-rewriting of Q according μ, denoted by $\beta_\preccurlyeq$, is the CQ $u(\mathsf{body}(R)) \cup u(Q \setminus Q')$.*

Example 7 *Consider the rules from Ex. 3, the rule $R = b(x) \rightarrow t(x,y)$ (where y is existential) and the query $Q_1 = \{t(u,v), q(v)\}$. There is no piece-unifier of Q_1 with R but there is a $\preccurlyeq$-piece-unifier $\mu = (Q_1, \mathsf{head}(R), \{x \mapsto u, y \mapsto v\})$, since R_4 subsumes $t(u,v) \rightarrow q(v)$. We obtain $\beta_\preccurlyeq(Q_1, R, \mu) = \{b(u)\}$. Consider now $Q_2 = \{q(w), s(z,w), c(w)\}$. With the $\preccurlyeq$-piece-unifier $\mu' = (\{q(w), s(z,w)\}, \mathsf{head}(R), \{x \mapsto w, y \mapsto z\})$, using R_3 and R_2', we obtain $\beta_\preccurlyeq(Q_2, R, \mu') = \{b(w), c(w)\}$.*

Logical entailment between queries and facts can be computed by a homomorphism check, which we now extend to embed $\preccurlyeq$.

Definition 9 ($\preccurlyeq$-homomorphism) *Let A and B be sets of atoms. A $\preccurlyeq$-homomorphism from B to A is a substitution h from $\mathsf{vars}(B)$ to $\mathsf{terms}(A)$ such that for all $b \in B$, there is $a \in A$ with $a \preccurlyeq h(b)$.*

Example 8 *Consider Q_1 and Q_2 from Ex. 7. The substitution $h = \{(u,w), (v,z)\}$ is a $\preccurlyeq$-homomorphism from Q_1 to Q_2. Indeed, $t(u,v)$ and $q(v)$ are both mapped to $s(z,w)$, by using R_2 and $R_2 \bullet R_4$ respectively.*

Property 2 *Let A and B be sets of atoms, and $\mathcal{R}_c$ be a set of compilable rules with associated preorder $\preccurlyeq$. There is a $\preccurlyeq$-homomorphism from B to A iff $\Phi(A), \mathcal{R}_c \models \Phi(B)$, where Φ assigns an existentially closed formula to a set of atoms.*

The next theorem states that the new rewriting operator $\beta_\preccurlyeq$, associated with $\preccurlyeq$-homomorphism, is logically sound and complete.

Theorem 1 (Soundness and Completeness of $\beta_\preccurlyeq$) *Let $\mathcal{K} = (F, \mathcal{R})$ be a KB, where $\mathcal{R}$ is partitioned into $\mathcal{R}_e$ and $\mathcal{R}_c$, a set of compilable rules with associated preorder $\preccurlyeq$. Let Q be a CQ. Then $F, \mathcal{R} \models Q$ iff there is $Q' \in \beta_\preccurlyeq^*(Q, \mathcal{R}_e)$ with a $\preccurlyeq$-homomorphism from Q' to F.*

Proof:(Sketch) We rely on the soundness and completeness of classical piece-based rewriting and prove that: (1) if $Q' \in \beta_\preccurlyeq^*(Q, \mathcal{R}_e)$ then $Q' \in \beta^*(Q, \mathcal{R})$ and (2) if $Q' \in \beta^*(Q, \mathcal{R})$ then there is $Q'' \in \beta_\preccurlyeq^*(Q, \mathcal{R}_e)$ such that $Q' \preccurlyeq Q''$. $\square$

4 A Correct $\preccurlyeq$-Rewriting Algorithm

The compilation step consists in partitioning $\mathcal{R}$ into $\mathcal{R}_c$ and $\mathcal{R}_e$, and compiling $\mathcal{R}_c$ into the preorder $\preccurlyeq$. Then, the rewriting step (Algorithm 1) follows the general schema of [König *et al.*, 2012]. Given $\mathcal{R}_e$, $\preccurlyeq$ and Q, the algorithm starts from the rewriting set $\mathcal{Q}_F = \{Q\}$ and proceeds in a breadth-first manner. At each step, queries from $\mathcal{Q}_F$ that have been generated at the preceding step (i.e., set $\mathcal{Q}_E$) are explored. "Exploring" a query consists of computing the set of one-step rewritings of this query with all rules in $\mathcal{R}_e$ (set $\mathcal{Q}_t$). At the end of the step, only a cover of $\mathcal{Q}_F \cup \mathcal{Q}_t$ is kept (in case of equivalent queries, priority is given to $\mathcal{Q}_F$ for termination reasons).

Algorithm 1: $\preccurlyeq_c$-REWRITING ALGORITHM

```
Data: A set of existential rules $\mathcal{R}_e$, a preorder on atoms
      $\preccurlyeq$ and a conjunctive query $Q$
Result: A cover of the sound $\mathcal{R}_e$-$\preccurlyeq$-rewritings of $Q$
$\mathcal{Q}_F \leftarrow \{Q\}$; // resulting set
$\mathcal{Q}_E \leftarrow \{Q\}$; // queries to explore
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 $R \in \mathcal{R}_e$ do
            for $\mu$ $\preccurlyeq$-piece-unifier of $Q_i$ with $R$ do
                $\mathcal{Q}_t \leftarrow \mathcal{Q}_t \cup \{\beta_{\preccurlyeq}(Q_i, R, \mu)\}$;
    $\mathcal{Q}^c \leftarrow$ ComputeCover($\mathcal{Q}_F \cup \mathcal{Q}_t$); // update cover
    $\mathcal{Q}_E \leftarrow \mathcal{Q}^c \setminus \mathcal{Q}_F$; // select unexplored queries
    $\mathcal{Q}_F \leftarrow \mathcal{Q}^c$;
return $\mathcal{Q}_F$
```

Pairwise comparing all queries at *each* step may seem expensive since the comparison relies on a $\preccurlyeq$-homomorphism check. The point is to ensure the termination of the algorithm whenever a finite rewriting set exists: since a set of rewritings may be infinite and still have a finite cover, a cover has to be maintained at each step (or computed after a finite number of steps). Note, however, that for linear and sticky rules, this problem does not occur, and the cover could be computed only once at the end of the algorithm.

Theorem 1 is not sufficient to ensure the completeness of the produced set. Indeed, one has to ensure that by pruning more specific queries at each step, the algorithm does not "miss" rewritings. A sufficient property is the so-called *prunability* of the rewriting operator [König *et al.*, 2013]. Intuitively, this property ensures that for any Q_2 more specific than Q_1, the following holds: each one-step rewriting of Q_2 is either more specific than Q_1 itself, or than a one-step-rewriting of Q_1; hence no rewriting is missed if Q_2 is removed from the rewriting set without being explored. This property can be formally expressed as follows for $\beta_{\preccurlyeq}$:

Property 3 (Prunability) *Let Q_1 and Q_2 be CQs, R be a rule, and $\preccurlyeq$ be a preorder on atoms. If Q_2 is more specific than Q_1, i.e., Q_1 maps to Q_2 by a $\preccurlyeq$-homomorphism, then for all $\preccurlyeq$-piece-unifier μ_2 of Q_2 with R, either $\beta_{\preccurlyeq}(Q_2, R, \mu_2)$ is more specific than Q_1, or there is a $\preccurlyeq$-piece-unifier μ_1 of Q_1 with R such that $\beta_{\preccurlyeq}(Q_2, R, \mu_2)$ is more specific than $\beta_{\preccurlyeq}(Q_1, R, \mu_1)$.*

With the previous property and Theorem 1, we can prove the correctness of the algorithm:

Theorem 2 *Algorithm 1 outputs a (minimal) sound and complete finite rewriting set (with respect to $\preccurlyeq$-homomorphism), if such exists, and it does not terminate otherwise.*

Algorithm 1 stops exactly when a UCQ-rewriting exists for the input set of rules and query (a sufficient condition being that the set of rules is *fus*).

5 Query Evaluation

The rewriting set $\mathcal{Q}$ produced by Algorithm 1 can be seen as a "pivotal" representation, in the sense that it can be transformed into different kinds of queries, depending on the type of data storage and the applicative context. Obviously, $\mathcal{Q}$ can be directly evaluated with an adequate implementation of $\preccurlyeq$-homomorphism in the case the data can be loaded in main memory. Otherwise, the set $\mathcal{Q} \cup \mathcal{R}_c$ can be straightforwardly translated into a Datalog query. A mixed approach can be adopted with $\mathcal{R}_c$ being used to saturate the data, and $\mathcal{Q}$ being evaluated over the saturated data. One may even assume that all information that could be inferred by compilable rules is already present in the data, and delegate the encoding of this information to the database manager. This notion is called H-completeness in [Rodriguez-Muro *et al.*, 2013] in the specific context of DL ABoxes. In particular, if $\mathcal{R}_c$ is composed solely of hierarchical rules and the data are stored in a RDBMS, semantic index techniques allow to avoid the effective computation of saturation [Rodriguez-Muro and Calvanese, 2012].

When partial saturation of the data is not feasible, $\mathcal{Q}$ may also be *unfolded* into a set of CQs (i.e., a UCQ) $\mathcal{Q}'$: $\mathcal{Q}'$ is obtained from $\mathcal{Q}$ by adding, for each $Q \in \mathcal{Q}$, all Q' such that $Q' \preccurlyeq Q$ (then eliminating redundant queries). Our experiments (see the last section) show that it is more efficient to unfold $\mathcal{Q}$ than to directly compute $\mathcal{Q}'$. More compact forms of positive existential queries can be computed, for instance unions of *semi-conjunctive* queries (SCQs), which are conjunctions of disjunctions [Thomazo, 2013]: each CQ $Q \in \mathcal{Q}$ is transformed into an SCQ by replacing each atom $a \in Q$ by the disjunction of all atoms a' such that $a' \preccurlyeq a$.

6 Related Work

Since the seminal paper on DL-Lite [Calvanese *et al.*, 2005], a significant amount of work has been carried out on query rewriting algorithms, mainly for DL-Lite, but also for other Horn-DLs. The work closest to ours is certainly the tree-witness (tw) rewriting algorithm for DL-Lite [Kikot *et al.*, 2012] because of the similarities between tw-rewritings and pieces. Another similarity is that tw-rewriting can make the assumption that the database is already saturated with inferrable knowledge that does not involve existential variables (H-completeness assumption). In the context of DL-Lite, this kind of knowledge corresponds exactly to compilable rules (up to the usual DL restrictions: predicate arity bounded by two, no multiple occurrences of variables in atoms). Hence, one could see our technique as an extension of tw-rewriting with the H-completeness assumption to existential rule KBs. However, the underlying techniques are quite different. Moreover, tw-rewriting heavily exploits the specificities of DL-Lite. First, "DL-Lite rules" without existential variables are necessarily compilable rules. Second, the "anonymous part" of the possibly infinite canonical model of a DL-Lite knowledge base is a set of *trees* (instead of a hypergraph with any structure for existential rules). This allows for a smart technique that rewrites the query in a *single* pass (instead of possibly exponentially many passes with *fus* rules).

Regarding general existential rules, two rewriting methods were proposed [Baget *et al.*, 2009; Gottlob *et al.*, 2011] and respectively implemented in Alaska/PURE [König *et al.*, 2012] and Nyaya [Virgilio *et al.*, 2012].

7 Experiments

Our algorithm[2] was implemented in Java, as an extension of the query rewriting prototype PURE. All tests were performed on a DELL machine with a processor at 3.60 GHz and 16 GB of RAM. As benchmarks dedicated to existential rules are not available yet, and in order to compare with other tools producing UCQs, which are mostly restricted to DL-Lite, we considered rule bases obtained by translation of DL-Lite$_{\mathcal{R}}$ ontologies: first, the widely used benchmark introduced in [Pérez-Urbina *et al.*, 2009] (i.e., ADOLENA (A), STOCKEXCHANGE (S), UNIVERSITY (U) and VICODI (V)); second, very large ontologies built from OpenGalen2 (G) and OBOProtein (O), and used in [Trivela *et al.*, 2013], which respectively contain more than 53k and 34k rules, with 54% and 64% of compilable rules. Each ontology is provided with 5 handcrafted queries. Timeout was set to 10 minutes. Due to space limitation, we list only parts of the experiments.

We first evaluated the impact of rule compilation on the rewriting process, w.r.t. rewriting sizes and runtime respectively. We denote by PURE$_C$ the extension of PURE and call *pivotal* UCQ its output. Table 1 shows the size of the UCQ (we recall that it is the same for all systems outputting a minimal, sound and complete UCQ), the size of the pivotal UCQ produced by PURE$_C$, and the number of generated queries during the rewriting process for PURE and PURE$_C$. Missing values are due to timeouts. We find a huge gap between the sizes of the output; the pivotal UCQ is often restricted to a single CQ even when the UCQ has thousands of CQs (which also explains the gap between the numbers of generated queries). Unsurprisingly, the results on the runtimes lead to similar observations (Table 2, Columns 1 and 2). We see that PURE$_C$ remains faster than PURE when we include the time required to unfold the pivotal UCQ into a UCQ (Table 2, Column 3), except for Q_2 on O, which comes from the fact that the pivotal UCQ is almost as large as the UCQ. Note that we implemented a brute-force unfolding method, which removes redundant CQs only at the end by a pairwise comparison of queries. Almost all the unfolding time is actually spent in checking redundancies. Nervertheless, the size of the UCQ obtained for some queries (up to more than 30000 CQs on O) clearly advocates for more compact forms of output.

We also compared to other query rewriting tools, namely: PURE and Nyaya, which are the only tools processing existential rules, as well as some well-known DL tools: Requiem [Pérez-Urbina *et al.*, 2009] (optimized "full modality"), Iqaros [Imprialou *et al.*, 2012], Rapid [Chortaras *et al.*, 2011] and tw-rewriting (part of the Ontop OBDA system [Rodriguez-Muro *et al.*, 2013]). We emphasize again that these DL tools exploit the specificities of DL-Lite, specially the most recent ones, Rapid and tw-rewriting, whereas the algorithms of Nyaya and PURE are designed for general existential rules. Despite this fact, PURE$_C$ (without or with unfolding) scales well on DL-Lite large ontologies (except for extreme cases which are difficult for all tools, see Ontology O). Globally, PURE$_C$ behaves similarly to tw-rewriting and Rapid. If we restrict the comparaison to classical UCQ output, the fastest tools are undeniably tw-rewriting and Rapid, followed by PURE$_C$ with unfolding. The difficulties of Nyaya on A can be explained by the fact that A contains some rules with two atoms in the head, whereas Nyaya only processes rules with a single head; hence, it had to take as input the ontology obtained by decomposing these rules into single-head rules, which introduces new predicates, whereas PURE processes rules with any head size. Nyaya could not process the very large ontologies G and O, which also needed to be decomposed.

We checked that all systems return exactly the same size of UCQ, hence the choice of a given query rewriting tool between those outputting a UCQ is irrelevant for the query evaluation step. We carried out additional experiments to compare

		UCQ	p-UCQ	gen (PURE)	gen (PURE$_C$)
A	Q1	27	2	460	14
	Q2	50	2	172	2
	Q3	104	1	317	1
	Q4	224	2	827	6
	Q5	624	1	1417	1
V	Q1	15	1	15	1
	Q2	10	1	10	1
	Q3	72	1	118	1
	Q4	185	1	329	1
	Q5	30	1	60	1
G	Q1	2	1	3	1
	Q2	1152	1	1276	1
	Q3	488	5	1515	5
	Q4	147	1	155	1
	Q5	324	19	909	19
O	Q1	27	20	29	22
	Q2	1356	1264	1356	1264
	Q3	33887	1	-	1
	Q4	34733	682	-	794
	Q5	36612	-	-	-

Table 1: Impact of rule compilation on sizes

		Pure	Pure$_C$	Pure$_C$ to UCQ	Nyaya	Requiem	Iqaros	tw	Rapid
A	Q1	180	10	130	1120	260	50	10	10
	Q2	90	0	40	860	100	50	10	20
	Q3	170	10	30	2360	130	190	0	30
	Q4	280	0	130	5550	250	130	10	40
	Q5	1500	0	440	33200	460	570	10	90
V	Q1	10	0	0	10	10	10	0	0
	Q2	10	0	10	50	10	10	0	0
	Q3	110	0	70	20	60	20	0	20
	Q4	120	0	60	20	130	30	0	30
	Q5	10	0	10	20	70	40	10	20
G	Q1	0	0	10	-	40	40	0	0
	Q2	1060	50	620	-	209040	5860	10	70
	Q3	1020	70	260	-	259100	9180	20	50
	Q4	20	10	10	-	190250	770	0	10
	Q5	890	30	90	-	238450	7400	20	40
O	Q1	440	130	140	-	3440	6670	10	20
	Q2	1160	1110	1870	-	21780	27810	570	950
	Q3	TO	90	557990	-	TO	TO	70	610
	Q4	TO	430	TO	-	TO	139980	1230	14690
	Q5	TO	TO	TO	-	TO	TO	TO	562220

Times are in ms, with increments of 10 *ms* (0 *means* < 10)*;*
TO *stands for timeout.*

Table 2: Impact of rule compilation on rewriting time

[2]Available at https://github.com/graphik-team/graal

the evaluation of the pivotal UCQ over the data saturated by the compilable rules and the evaluation of the corresponding classical UCQ over the initial data (data generated with the modified LUBM generator [Lutz *et al.*, 2013] and stored in a MySQL database). As expected, in a number of cases the system did not accept the classical UCQ because of its size, and in the other cases the pivotal UCQ was evaluated much more efficiently than the classical UCQ.

Further work includes extending query rewriting techniques outside the *fus* fragment, by exploiting datalog rewritability or combining with other paradigms for rules.

Acknowledgments. This work was partially funded by the ANR project PAGODA (ANR-12-JS02-007-01).

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.*, 2011] 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.

[Beeri and Vardi, 1981] C. Beeri and M. Vardi. The implication problem for data dependencies. In *ICALP'81*, volume 115 of *LNCS*, pages 73–85, 1981.

[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.*, 2012] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[Calvanese *et al.*, 2005] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *AAAI*, pages 602–607, 2005.

[Chortaras *et al.*, 2011] Alexandros Chortaras, Despoina Trivela, and Giorgos B. Stamou. Optimized Query Rewriting for OWL 2 QL. In *CADE'11*, pages 192–206, 2011.

[Civili and Rosati, 2012] C. Civili and R. Rosati. A Broad Class of First-Order Rewritable Tuple-Generating Dependencies. In *Datalog 2.0 Worksop*, pages 68–80, 2012.

[Civili *et al.*, 2013] C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli, and D. F. Savo. MASTRO STUDIO: Managing Ontology-Based Data Access applications. *PVLDB*, 6(12):1314–1317, 2013.

[Eiter *et al.*, 2012] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query Rewriting for Horn-SHIQ Plus Rules. In *AAAI.*, 2012.

[Gottlob and Schwentick, 2012] G. Gottlob and T. Schwentick. Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. In *KR*, 2012.

[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.

[Kikot *et al.*, 2011] S. Kikot, R. Kontchakov, and M. Zakharyaschev. Polynomial Conjunctive Query Rewriting under Unary Inclusion Dependencies. In *RR*, 2011.

[Kikot *et al.*, 2012] S. Kikot, R. Kontchakov, and M. Zakharyaschev. Conjunctive Query Answering with OWL 2 QL. In *KR*, 2012.

[König *et al.*, 2012] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. A Sound and Complete Backward Chaining Algorithm for Existential Rules. In *RR'12*, pages 122–138, 2012.

[König *et al.*, 2013] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. On the Exploration of the Query Rewriting Space with Existential Rules. In *RR*, pages 123–137, 2013.

[Kontchakov *et al.*, 2011] R. Kontchakov, C. Lutz, D. Toman, F.Wolter, and M. Zakharyaschev. The Combined Approach to Ontology-Based Data Access. In *IJCAI*, 2011.

[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.

[Lutz *et al.*, 2013] C. Lutz, I. Seylan, D. Toman, and F. Wolter. The Combined Approach to OBDA: Taming Role Hierarchies Using Filters. In *ISWC'2013*, pages 314–330, 2013.

[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.

[Rodriguez-Muro *et al.*, 2013] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-Based Data Access: Ontop of Databases. In *ISWC, Proceedings, Part I*, pages 558–573, 2013.

[Rosati and Almatelli, 2010] R. Rosati and A. Almatelli. Improving Query Answering over DL-Lite Ontologies. In *KR'10*, 2010.

[Stefanoni *et al.*, 2012] G. Stefanoni, B. Motik, and I. Horrocks. Small Datalog Query Rewritings for EL. In *DL*, 2012.

[Thomazo and Rudolph, 2014] M. Thomazo and S. Rudolph. Mixing Materialization and Query Rewriting for Existential Rules. In *ECAI*, pages 897–902, 2014.

[Thomazo *et al.*, 2012] M. Thomazo, J.-F. Baget, M.-L. Mugnier, and S. Rudolph. A Generic Querying Algorithm for Greedy Sets of Existential Rules. In *KR*, 2012.

[Thomazo, 2013] M. Thomazo. Compact Rewritings for Existential Rules. In *IJCAI*, 2013.

[Trivela *et al.*, 2013] Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising Resolution-Based Rewriting Algorithms for DL Ontologies. In *DL'13*, pages 464–476, 2013.

[Virgilio *et al.*, 2012] Roberto De Virgilio, G. Orsi, L. Tanca, and R. Torlone. NYAYA: A System Supporting the Uniform Management of Large Sets of Semantic Data. In *ICDE*, pages 1309–1312, 2012.