A Sound and Complete Backward Chaining Algorithm for Existential Rules

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

A Sound and Complete Backward Chaining Algorithm for Existential Rules

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

University Montpellier 2, France

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.

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). To address this issue, several logic-based formalisms have been developed. The dominant approach is based on description logics (DLs), with the most studied DLs in this context being lightweight DLs, such as DL-Lite and $\mathcal{EL}$ families [Baa03, CGL$^+$07] and their Semantic Web counterparts, so callso-calleded tractable fragments of OWL2. A newer approach, to which this paper contributes, is based on existential rules. Existential rules have the ability of generating new unknown individuals, a feature that has been recognized as crucial in an open-world perspective, where it cannot be assumed that all individuals are known in advance. These rules are of the form $body \rightarrow head$, where the body and the head are conjunctions of atoms (without functions), and variables that occur only in the head are *existentially* quantified, hence the name $\forall\exists$-rules in [BLMS09, BLM10] or existential rules in [BMRT11, KR11]. They are also known as Datalog +/-, a recent extension of plain Datalog to tuple-generating dependencies (expressive constraints that have long been studied in databases and have the same logical form as existential rules) [CGK08, CGL09].

In this paper, we consider knowledge bases composed of a set of facts -or data- and of existential rules. The basic problem, query answering, consists of computing the set of answers to a query in the knowledge base. We consider conjunctive queries (CQs), which are the standard basic queries. CQs can be seen as existentially quantified conjunctions of atoms. The fundamental decision problem associated with query answering

M. Krötzsch and U. Straccia (Eds.): RR 2012, LNCS 7497, pp. 122–138, 2012.

can be expressed in several equivalent ways, in particular as a CQ entailment problem: is a given (Boolean) CQ logically entailed by a knowledge base?

CQ entailment is undecidable for general existential rules. There is currently an intense research effort aimed at finding decidable subsets of rules that provide good tradeoffs between expressivity and complexity of query answering (see [Mug11] for a synthesis). With respect to (lightweight) DLs, these decidable rule fragments are more powerful and flexible. However, the rule-based ODBA framework is rather new and it 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*, schematized in Figure 1. Both can be seen as ways of integrating the rules either into the facts or into the query (denoted by Q in the figure). Forward chaining uses the rules to enrich the facts and the query is entailed if it maps by homomorphism to the enriched facts. Backward chaining proceeds in the "reverse" manner: it uses the rules to rewrite the query in several ways and the initial query is entailed if a rewritten query maps to the initial facts.

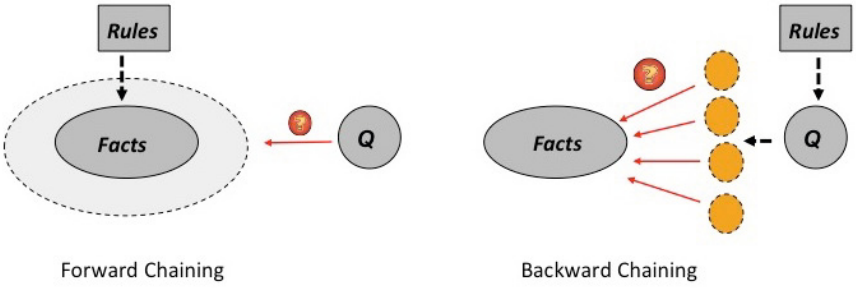

Fig. 1. Forward / Backward Chaining

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 conjunctive queries in the set. An approach initiated with DL-Lite consists of decomposing backward chaining into two steps: (1) rewrite the initial query as a union of CQs (2) use a database management system to answer this union query. This approach aims to benefit from the optimizations developed for classical database queries. Since the CQs are independent, their processing can be easily parallelized. This approach can be generalized to rewritings into first-order queries, which are the logical counterpart of SQL queries (with closed-world assumption). It is at the core of several systems, such as Nyaya [GOP11], QuOnto [CGL$^+$07] and Requiem [PUHM09]. Such rewritings are usually of exponential size with respect to the initial query (however [KKZ11] exhibits specific cases where the rewriting is of polynomial size). In [RA10] another method, also devoted to DL-Lite, is proposed: it consists of rewriting the query into a non-recursive Datalog program,

which in turn can be translated into a first-order query of smaller size than the union of CQs that would be output. [GS12] defines such a rewriting with polynomial size in both Q and $\mathcal{R}$ for some specific classes of rules. However, distributed processing of non-recursive Datalog programs is not as easy as for UCQs.

While these works focus on specific rule sublanguages, in this paper we consider backward chaining with *general 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 (so-called "finite unification sets" of rules in [BLM10], which includes expressive classes of rules, see Section 3).

The originality of our method lies in the rewriting step, which is based on a graph notion, that of a *piece*. Briefly, a piece is a subset of atoms from the query that must be erased together during a rewriting step. The backward chaining mechanisms classically used in logic programming process rules and queries atom by atom: at each step, an atom a of a query Q is unified with the head of a rule R (which is composed of 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 $u(body(R)) \cup u(Q \setminus \{a\})$. Here, existential variables in rule heads have to be taken into account, which prevents the use of atomic unification. Instead, subsets of atoms ("pieces") have to be considered at once. We present below a very simple example (in particular, the head of the rule is restricted to a single atom).

Example 1. Let the rule $R = q(x) \rightarrow p(x, y)$, which corresponds to the logical formula $\forall x \ (q(x) \rightarrow \exists y \ p(x, y))$, and the Boolean CQ $Q = p(u, v) \wedge p(w, v) \wedge p(w, t) \wedge r(u, w)$ (a closed existential formula), where all the terms are variables. Assume we want to unify $p(u, v)$, the first atom in 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 $q(x) \wedge p(w, y) \wedge p(w, t) \wedge r(x, w)$ as would be done in a "classical" backward chaining step would be incorrect (intuitively, the fact that the atoms $p(u, v)$ and $p(w, v)$ in Q share a variable would be lost with $q(x)$ and $p(w, y)$). Thus, $p(u, v)$ and $p(w, v)$ are both unified with the head of R by means of the following substitution: $\{(u, x), (v, y), (w, x)\}$. Since w is associated with a non-existential variable, there is no need to include $p(w, t)$ in the set, although in this example it could be added. $\{p(u, v), p(w, v)\}$ is called a piece. The corresponding rewriting of Q is $q(x) \wedge p(x, t) \wedge r(x, x)$.

Pieces come from earlier work on conceptual graph rules, whose logical translation is exactly existential rules [SM96]. This notion has then been recast in the framework of existential rules in [BLMS09][BLMS11]. In this paper, we start from the definition of a piece-unifier, which unifies part of a rule head and part of the query, while respecting pieces: when it unifies an atom in the query, it must unify the whole piece to which this atom belongs. Backward chaining based on piece-unifiers is known to be sound and complete (e.g. [BLMS11], and basically [SM96] for conceptual graphs). 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, let us say that a set $\mathcal{Q}$ of rewritten CQs from a CQ Q and a set of rules $\mathcal{R}$ is *sound and complete* if the following holds: for any set of facts F, if Q is entailed by F and $\mathcal{R}$ then there is a query Q_i in $\mathcal{Q}$ such that Q_i is entailed by F (completeness), and reciprocally (soundness). We point out that any sound and complete set of CQs (w.r.t. the same Q and $\mathcal{R}$) 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. It is easily checked that the algorithm we propose produces such a minimal set. If we moreover delete redundant atoms from the obtained CQs (which can be performed by a linear number of homomorphism tests for each query), 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).

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 (e.g. [CGK08], [BLMS09]). Besides, many rules found in the literature have an atomic head. Restricting our focus to atomic head rules allows us to obtain nice properties. We first show that it is sufficient to consider piece-unifiers that (1) are most general unifiers, and (2) process a single piece at once.[1] We then show that the number of most general single-piece unifiers of a query Q with the (atomic) head of a rule R is bounded by the size of the query. Finally, we exploit the fact that each atom in Q belongs to at most one piece with respect to R (which is false for general existential rules) to efficiently compute a rewriting step, i.e., generate all queries obtained from R and Q by most general single-piece unifiers of Q with R. A backward chaining algorithm benefiting from these results 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. A long version of this paper with all proofs is available as a technical report [KLMT12].

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 (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.

[1] Actually, this property should be extendable to rules with non-atomic head, but this would first involve defining a suitable comparison operation between piece-unifiers, operation which is simply defined with atomic-head rules.

Given atom sets A and B, a *homomorphism* h from A to B is a substitution of $vars(A)$ by $terms(B)$ such that $h(A) \subseteq B$. 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 $B \leq A$).

A *fact* is the existential closure of a 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). Note that facts and BCQs have the same logical form. In the following, we will see them as sets of atoms. It is well-known that, given a fact F and a BCQ Q, $F \models Q$ iff there is a homomorphism from Q to F.

The answer to a BCQ Q in a fact F is *yes* if there is a homomorphism from Q to F. Otherwise, let $x_1 \dots x_q$ be the free variables in Q: a tuple of constants $(a_1 \dots a_q)$ is an answer to Q in F if there is a homomorphism from Q to F that maps x_i to a_i for each i. In the following, we consider only Boolean queries for simplicity reasons. This is not a restriction, since a CQ with free variables $x_1 \dots x_q$ can be translated into a BCQ by adding the atom $ans(x_1 \dots x_q)$, where ans is a special predicate not occurring in the knowledge base. Since ans can never be erased by a rewriting step, it guarantees that the x_i can only be substituted and will not "disappear". Note that we could also consider unions of conjunctive queries, in this case each conjunctive subquery would be processed separately.

Definition 1 (Existential rule). *An* existential rule *(or simply* rule *when clear from the context) 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* $B = body(R)$ *and* $H = head(R)$ *are conjunctions of atoms, resp. called the* body *and the* head *of* R*. The* frontier *of* R*, noted* $fr(R)$*, is the set of variables* $vars(B) \cap vars(H) = \mathbf{y}$*. The* existential variables *in* R*, noted* $exist(R)$*, is the set of variables* $vars(H) \setminus fr(R) = \mathbf{z}$*.*

In the following, we will omit quantifiers in rules as there is no ambiguity.

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 (Boolean) CQ entailment problem is the following: given a KB $\mathcal{K} = (F, \mathcal{R})$ and a BCQ Q, does $F, \mathcal{R} \models Q$ hold?

This question can be solved with forward chaining: $F, \mathcal{R} \models Q$ iff there exists a finite sequence $(F_0 = F), \dots, F_k$, where each F_i for $i > 0$ is obtained by applying a rule from $\mathcal{R}$ to F_{i-1}, such that $F_k \models Q$ (see e.g. [BLMS11] for details).

As explained in the introduction, backward chaining relies on a unification operation between a query and a rule head. The following definition of piece-unifier is an alternative definition of the operation defined in [BLMS11].

Other Notations: Throughout the paper we note respectively R and Q the considered rule and query. We assume that R and Q have no variables in common. When needed, a "fresh copy" of R is obtained by bijectively renaming the variables in R into "fresh" variables. We note $\mathcal{C}$ the set of constants occurring in the set of rules $\mathcal{R}$ and in Q. Given $Q' \subseteq Q$, we note $\bar{Q}'$ the set $Q \setminus Q'$. The variables in $vars(Q') \cap vars(\bar{Q}')$ are called *separating variables* and denoted $sep(Q')$.

A piece-unifier is defined as a pair (Q', u), where Q' is a non-empty subset of Q, and u is a substitution that "unifies" Q' with a subset H' of $head(R)$, in the sense

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

that $u(Q') = u(H')$; H' is the subset of $\mathit{head}(R)$ composed of atoms a such that $u(a) = u(b)$ for some $b \in Q'$. The substitution u can be decomposed as follows: (1) it specializes the frontier of R, thus $\mathit{head}(R)$, while leaving existential variables 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, i.e., they are mapped to $u(\mathit{fr}(R))$ or to constants.

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 $\mathit{fr}(R) \cup \mathit{vars}(Q')$ by $\mathit{terms}(\mathit{head}(R)) \cup \mathcal{C}$ such that:*

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

u is divided into u^R with domain $\mathit{fr}(R)$ and $u^{Q'}$ with domain $\mathit{vars}(Q')$.

Note that instead of $\mathcal{C}$, we could consider $\mathit{consts}(Q') \cup \mathit{consts}(\mathit{head}(R))$, however $\mathcal{C}$ is convenient for proof purposes.

Example 2. Let us take again $R = q(x) \rightarrow p(x, y)$ and $Q = p(u, v) \wedge p(w, v) \wedge p(w, t) \wedge r(u, w)$. Here are three piece-unifiers of Q with R:
$\mu_1 = (Q'_1, u_1)$ with $Q'_1 = \{p(u, v), p(w, v)\}$ and $u_1 = \{(u, x), (v, y), (w, x)\}$
Note that we will 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 $Q'_3 = \{p(u, v), p(w, v), p(w, t)\}$ and $u_3 = \{(u, x), (v, y), (w, x), (t, y)\}$
These piece-unifiers will be called the "most general piece-unifiers" of Q with R in Section 4.

In the previous example, R has an atomic head, thus a piece-unifier of Q' with R actually unifies the atoms from Q' and the head of R into a single atom. In the general case, a piece-unifier unifies Q' and a subset H' of $\mathit{head}(R)$ into a set of atoms, as shown by the next example.

Example 3. Let $R = q(x) \rightarrow p(x, y) \wedge p(y, z) \wedge p(z, t) \wedge r(y)$ and $Q = p(u, v) \wedge p(v, w) \wedge r(u)$. A piece-unifier of Q with R is (Q'_1, u_1) with $Q'_1 = \{p(u, v), p(v, w)\}$ and $u_1 = \{(u, x), (v, y), (w, z)\}$. $H' = \{p(x, y), p(y, z)\}$ and $u_1(Q') = u_1(H') = H'$. Another piece-unifier is (Q'_2, u_2) with $Q'_2 = Q$ and $u_2 = \{(u, y), (v, z), (w, t)\}$; in this case, $H' = \{p(y, z), p(z, t), r(y)\}$.

Finally, the next example illustrates the role of constants (in the query here, but constants may also occur in rules).

Example 4. Let $R = q(x, y) \rightarrow p(x, y, z)$ and $Q = p(u, a, v) \wedge p(a, w, v)$, where a is a constant. The variable v has to be mapped to the existential variable z. The unique piece-unifier is here $(Q, \{(x, a), (y, a), (u, a), (w, a), (v, z)\})$.

We are now able to formally define *pieces*. A piece of Q can be seen as a minimal subset Q' satisfying the above definition of a piece-unifier. Generally speaking, a set of atoms can be partitioned into subsets called pieces according to 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 do not belong to T [BLMS11]. Note that constants do not allow to connect atoms. Here, T is the set of variables from Q' that are not mapped to existential variables by u.

Definition 3 (Piece). *[BLMS11] Let A be a set of atoms and $T \subseteq (vars(A))$. A piece of A according to T is a minimal non-empty subset P of A such that, for all a and a' in A, if $a \in P$ and $(vars(a) \cap vars(a')) \not\subseteq T$, 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 $u(x) \notin exist(R)$ (equivalently: $u(x) \in fr(R) \cup \mathcal{C}$). 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 3 (contd) Q'_1 and Q'_2 are pieces. Note that an atom may belong to different pieces according to different unifiers (it is the case here for $p(u, v)$ and $p(v, w)$).

The following property is easily checked and justifies the name "piece-unifier":

Property 1. For any piece-unifier $\mu = (Q', u)$, Q' is a set of pieces of Q. In particular, $sep(Q') \subseteq T_Q(\mu)$.

To summarize, a piece of Q is a minimal subset of atoms that must be considered together once cutpoints in Q have been defined. A piece-unifier may process several pieces. In Section 4, we will focus on unifiers processing a single piece. Finally, note that in rules without existential variables, such as in plain Datalog, each piece is restricted to a single atom. Concerning the next definitions, we recall the assumption that $vars(R) \cap vars(Q) = \emptyset$:

Definition 5 (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 $u^R(body(R)) \cup u^{Q'}(\bar{Q}')$.*

Definition 6 ($\mathcal{R}$-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 $(Q_0 = Q), Q_1, \ldots, Q_k$ 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 $Q_{i+1} = \beta(Q_i, R, \mu)$.*

Theorem 1 (Soundness and completeness of piece-based backward chaining). *(basically[SM96]) Let a KB $\mathcal{K} = (F, \mathcal{R})$ and a (Boolean) CQ Q. Then $F, \mathcal{R} \models Q$ iff there is an $\mathcal{R}$-rewriting of Q that maps to F.*

The soundness and completeness of the piece-based backward chaining mechanism can be proven via the following equivalence with forward chaining: there is an $\mathcal{R}$-rewriting from Q to Q' that maps to F iff there is a sequence of rule applications leading from F to F' such that Q maps to F'.

To evaluate the quality of rewriting sets produced by different 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 (Boolean) 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 $Q_i \in \mathcal{Q}$ such that Q_i maps to F.*

As expressed by Theorem 1, the set of $\mathcal{R}$-rewritings that can be produced with piece-unifiers is sound and complete. In the next section, we will address the issue of the size of a rewriting set.

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 such that $Q_2 \leq Q_1$ 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 (and it will not undermine soundness either). 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 $\mathcal{Q}^c \subseteq \mathcal{Q}$ such that:*

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

Note that a cover is inclusion-minimal. Moreover, it can be easily checked that all covers of $\mathcal{Q}$ have the same cardinality.

Example 5. Let $\mathcal{Q} = \{Q_1, \ldots, Q_6\}$ and the following preorder over $\mathcal{Q}$: $Q_6 \leq Q_5$; $Q_5 \leq Q_1, Q_2$; $Q_4 \leq Q_1, Q_2, Q_3$; $Q_1 \leq Q_2$ and $Q_2 \leq Q_1$ (Q_1 and Q_2 are thus equivalent). There are two covers of $\mathcal{Q}$, namely $\{Q_1, Q_3\}$ and $\{Q_2, Q_3\}$.

Note that the set of rewritings of Q can have a finite cover even when it is infinite, as illustrated by Example 6.

Example 6. Let $Q = t(u), R_1 = t(x) \wedge p(x, y) \rightarrow r(y), R_2 = r(x) \wedge p(x, y) \rightarrow t(y)$. The set of $\mathcal{R}$-rewritings of Q with $\{R_1, R_2\}$ is infinite. The first generated queries are the following (note that rule variables are renamed when needed):

$Q_0 = t(u)$
$Q_1 = r(x) \wedge p(x, y)$ // from Q_0 and R_2 with $\{(u, y)\}$
$Q_2 = t(x_0) \wedge p(x_0, y_0) \wedge p(y_0, y)$ // from Q_1 and R_1 with $\{(x, y_0)\}$
$Q_3 = r(x_1) \wedge p(x_1, y_1) \wedge p(y_1, y_0) \wedge p(y_0, y)$ // from Q_2 and R_2 with $\{(x_0, y_1)\}$
$Q_4 = t(x_2) \wedge p(x_2, y_2) \wedge p(y_2, y_1) \wedge p(y_1, y_0) \wedge p(y_0, y)$ // from Q_3 and R_1
and so on ...

However, the set of the most general $\mathcal{R}$-rewritings is $\{Q_0, Q_1\}$ since any other query than can be obtained is more specific than Q_0 or Q_1.

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 [BLMS11], but several *fus* recognizable classes have been exhibited in the literature: atomic-body [BLMS09], also known as linear TGDs [CGL09], domain-restricted [BLMS09], (join-)sticky [CGP10]. 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 (while loop iteration), all queries generated at the preceding step and kept in the current cover are explored.

Algorithm 1. A BREADTH-FIRST 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$(\mathcal{Q}_F \cup \mathcal{Q}_t)$;
  $\mathcal{Q}_E \leftarrow \mathcal{Q}^c \backslash \mathcal{Q}_F$; *// select unexplored queries of the cover*
  $\mathcal{Q}_F \leftarrow \mathcal{Q}^c$;
return $\mathcal{Q}_F$

For any *fus*, CQ entailment is solvable in AC^0 for data complexity.[3] However, data complexity hides the complexity coming from the query: the size of the rewriting set can be exponential in the size of the original query. Most of the literature about rewriting techniques focuses on minimizing the *size* of the output rewritings. We will show that this size should not be a decisive criterion for comparing algorithms that output a union of CQs.

All covers of a given set have the same (minimal) cardinality. We now prove that this property can be extended to the covers of all sound and complete rewriting sets of Q, no matter of the rewriting technique used to compute these sets.

Theorem 2. *Let $\mathcal{R}$ be a fus, Q be a BCQ, and let $\mathcal{Q}$ 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 with $\mathcal{R}$.*

Proof. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two arbitrary sound and complete rewriting sets of Q with $\mathcal{R}$, and $\mathcal{Q}_1^c$ and $\mathcal{Q}_2^c$ be one of their respective covers. $\mathcal{Q}_1^c$ and $\mathcal{Q}_2^c$ are also sound and

[3] AC^0 is a subclass of LOGSPACE itself included in PTIME. Data complexity means that Q and $\mathcal{R}$ are fixed, thus the input is restricted to F.

complete, and are of smaller cardinality. We show that they have the same cardinality. Let $Q_1 \in \mathcal{Q}_1^c$. There exists $Q_2 \in \mathcal{Q}_2^c$ such that $Q_1 \leq Q_2$. If not, Q would be entailed by $F = Q_1$ and $\mathcal{R}$ since $\mathcal{Q}_1^c$ is a sound rewriting set of Q (and Q_1 maps to itself), but no elements of $\mathcal{Q}_2^c$ would map to F: thus, Q_2^c would not be complete. Similarly, there exists $Q_1' \in \mathcal{Q}_1^c$ such that $Q_2 \leq Q_1'$. Then $Q_1 \leq Q_1'$, which implies that $Q_1' = Q_1$ by assumption on Q_1^c. For all $Q_1 \in \mathcal{Q}_1^c$, there exists $Q_2 \in \mathcal{Q}_2^c$ such that $Q_1 \leq Q_2$ and $Q_2 \leq Q_1$. Such a Q_2 is unique: indeed, two such elements would be comparable for $\geq$, which is not possible by construction of $\mathcal{Q}_2^c$. The function associating Q_2 with Q_1 is thus a bijection from $\mathcal{Q}_1^c$ to $\mathcal{Q}_2^c$, which shows that these two sets have the same cardinality. □

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. Please note that the algorithm presented in the sequel of this paper fullfils this property.

Furthermore, the proof of the preceding theorem shows that, given any two sound and complete rewriting sets of Q, there is a bijection from any cover of the first set to any cover of the second set such that two elements in relation are equivalent. However, these elements are not necessarily isomorphic (i.e., equal up to a variable renaming) because they may contain redundancies. It is well-known that the preorder induced by homomorphism on the set of all BCQs definable on some vocabulary is such that any equivalence class for this preorder possesses a unique element of minimal size (up to isomorphism), called its *core* (notion introduced for graphs, but easily transferable to queries). Every query can be transformed into its equivalent core by removing redundant atoms. From this remark and Theorem 2, we obtain:

Corollary 1. *Let $\mathcal{R}$ be a fus and Q be a BCQ. There is a unique sound and complete rewriting set of Q with $\mathcal{R}$ that has both minimal cardinality and elements of minimal size.*

4 Single-Piece Unification

We will now focus on rules with atomic head, which are often considered in the literature. Any rule can be decomposed into an equivalent set of rules with atomic head by introducing a new predicate gathering the variables of the original head, thus this restriction does not yield a loss in expressivity (e.g. [CGK08, BLMS09]).

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 that have to be considered in the backward chaining mechanism. We show it is sufficient to only keep most general single-piece unifiers. Moreover, the number of such unifiers is linear in the size of Q. Indeed, there is a unique way of associating any atom in Q with $\mathit{head}(R)$.

4.1 Correctness of Rewriting Restricted to Most General Single-Piece Unifiers

We recall that, given substitutions s_1 and s_2, s_1 is said to be more general than s_2 if s_2 can be obtained from s_1 by composition with an additional substitution (i.e., there is s s.t. $s_2 = s \circ s_1$). Piece-unifiers can be compared via their substitutions, provided that they are defined on the *same* subset of Q.

Definition 9 (Most general piece-unifier). *Let Q be a CQ, R be a rule, and $\mu_1 = (Q', u_1)$, $\mu_2 = (Q', u_2)$ be two piece-unifiers of Q with R, defined on the same set of pieces $Q' \subseteq Q$. 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. Let μ be a piece-unifier of Q with R defined on $Q' \subseteq Q$, μ is called a most general piece-unifier if for all μ' piece-unifier of Q with R defined on Q', we have $\mu \geq \mu'$.*

Property 2. Let μ_1 and μ_2 be two piece-unifiers with $\mu_1 \geq \mu_2$. μ_1 and μ_2 have the same pieces.

Definition 10 (Single-piece unifier). *A piece-unifier $\mu = (Q', u)$ of a CQ Q with a rule R is a single-piece unifier if Q' is a piece of Q according to $T_Q(\mu)$.*

From Property 2, it follows that a single-piece unifier can be compared only with other single-piece unifiers. The next results show that it is sufficient to consider (1) most general piece-unifiers (Theorem 3) (2) single-piece unifiers, (Theorem 4) and finally most general single-piece unifiers (Theorem 5).

Property 3. Let $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ be two piece-unifiers such that $\mu_1 \geq \mu_2$. Then $\beta(Q, R, \mu_1) \geq \beta(Q, R, \mu_2)$.

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 $\beta(Q_1, R, \mu_1) \geq \beta(Q_2, R, \mu_2)$.*

The following theorem follows from Property 3 and Lemma 1:

Theorem 3. *Given a BCQ Q and a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively most general piece-unifiers is sound and complete.*

Let $\mu = (Q', u)$ be a piece-unifier of Q with R. μ can be decomposed into several single-piece unifiers: for each piece P of Q according to $T_Q(\mu)$, there is a single-piece unifier (P, u_P) of Q with R where $u_P = u^R \cup u^{Q'}|_{vars(P)}$. However, applying successively each of these underlying single-piece unifiers may not lead to a CQ equivalent to $\beta(Q, R, \mu)$: the resulting query may be strictly more general than $\beta(Q, R, \mu)$, as the following example illustrates it.

Example 7. Let $R = p(x, y) \rightarrow q(x, y)$ and $Q = q(u, v) \wedge r(v, w) \wedge q(t, w)$. $\mu = (Q', u)$ with $Q' = \{q(u, v), q(t, w)\}$ and $u = \{(u, x), (v, y), (t, x), (w, y)\}$ is a piece-unifier of Q with R, which contains two pieces: $P_1 = \{q(u, v)\}$ and $P_2 = \{q(t, w)\}$. The rewriting of Q according to μ is $\beta(Q, R, \mu) = p(x, y) \wedge r(y, y)$. If we successively apply the two underlying single-piece unifiers, noted μ_{P_1} and μ_{P_2} (we note R' the fresh copy of R used for the second computation), we obtain $\beta(\beta(Q, R, \mu_{P_1}), R', \mu_{P_2}) = \beta(p(x, y) \wedge r(y, w) \wedge q(t, w), R', \mu_{P_2}) = p(x, y) \wedge r(y, y') \wedge p(x', y')$, which is strictly more general than $\beta(Q, R, \mu)$.

Property 4. For any piece-unifier μ of Q with R, there is a sequence of rewritings of Q with R using only single-piece unifiers and leading to a CQ Q^s such that $Q^s \geq \beta(Q, R, \mu)$.

From Lemma 1 and Property 4, it follows that:

Theorem 4. *Given a BCQ Q and a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively single-piece unifiers is sound and complete.*

Property 5. For any piece-unifier μ of Q with R, there is a sequence of rewritings of Q with R using only most general single-piece unifiers and leading to a CQ Q^s such that $Q^s \geq \beta(Q, R, \mu)$.

From Lemma 1 and Property 5, we obtain:

Theorem 5. *Given a BCQ Q and 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.*

4.2 Computing all the Most General Single-Piece Unifiers

We first check that properties of most general unifiers in the classical logical meaning also hold for piece-unifiers (that operate on the same subset of Q): unicity of a most general piece-unifier up to a bijective variable renaming and existence of a most general piece-unifier.

Lemma 2. *If two piece-unifiers $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ are equivalent (i.e., $\mu_1 \geq \mu_2$ and $\mu_2 \geq \mu_1$), then μ_1 and μ_2 can be obtained from each other by a bijective variable renaming.*

Lemma 3. *If two piece-unifiers $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ are incomparable (i.e., $\mu_1 \not\geq \mu_2$ and $\mu_2 \not\geq \mu_1$), then there exists a piece-unifier $\mu = (Q', u)$ with $\mu \geq \mu_1$ and $\mu \geq \mu_2$.*

The next property follows from the two previous lemmas:

Property 6. Let Q be a CQ and R be a rule. For any $Q' \subseteq Q$, if Q' is a piece for a piece-unifier of Q with R, then Q' is part of a *unique* most general (single-piece) piece-unifier of Q with R (up to a bijective variable renaming).

Lemma 4. *Let Q be a CQ and R be a rule. For all atoms $a \in Q$, there is at most one $Q' \subseteq Q$ such that $a \in Q'$ and Q' is a piece for a piece-unifier of Q with R.*

Property 6 and the above lemma entail the following result:

Theorem 6. *Every atom in Q participates in at most one most general single-piece unifier of Q with R (up to a bijective variable renaming).*

It follows that the number of most general single-piece unifiers of Q with R is less or equal to the cardinality of Q.

To compute most general single-piece unifiers, we first introduce the notion of pre-(piece)-unifier of a set of atoms with the head of a rule. A pre-unifier is an adaptation of a classical logical unifier, that takes existential variables into account, and chooses to keep variables from the head of the rule in the resulting atom. To become a piece-unifier, a pre-unifier has to satisfy an additional constraint on $sep(Q')$ (Condition 2 in piece-unifier definition).

Definition 11 (pre-unifier). *Let $Q' \subseteq Q$ and R be a rule. A pre-unifier u of Q' with R is a substitution of $\mathit{fr}(R) \cup \mathit{vars}(Q')$ by $\mathit{terms}(\mathit{head}(R)) \cup \mathcal{C}$ such that:*

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

Algorithm 2 computes a most general pre-unifier of a set of atoms, in a way similar to Robinson's algorithm.

Algorithm 2. MostGeneralPreUnifier

Data: A: a set of atoms with the same predicate p, $A \subseteq \mathit{head}(R) \cup Q$
Result: a most general pre-unifier of A if it exists, otherwise *Fail*
$u \leftarrow \emptyset$;
foreach $i \in$ *positions of* p **do**
  $E \leftarrow$ set of terms in position i in A;
  if *E contains two constants or two existential variables or (a constant and an existential variable) or (a frontier variable and an existential variable)* **then**
    return *Fail*
  if *E contains a constant or an existential variable* **then**
    $t \leftarrow$ this term
  else
    `// E contains at least one frontier variable`
    $t \leftarrow$ one of these frontier variables
  $u' \leftarrow \{(v, t) \mid v \textit{ is a variable in } E \textit{ and } v \neq t\}$
  $u \leftarrow u' \circ u$;
  $A \leftarrow u'(A)$;
return u

The fact that an atom from Q participates in at most one most general single-piece unifier suggests an incremental method to compute these unifiers. Assume the head of R has predicate p. We start from each atom $a \in Q$ with predicate p and compute the subset of atoms from Q that would necessarily belong to the same piece as a; more precisely, we build Q' such that Q' and $\mathit{head}(R)$ can be pre-unified, then check if $\mathit{sep}(Q')$ satisfies the additional condition of a piece-unifier. If there is a piece-unifier of Q' built in this way with $\mathit{head}(R)$, all atoms in Q' can be removed from Q for the search of other single-piece unifiers; otherwise, a is removed from Q for the search of other single-piece unifiers but the other atoms in Q' still have to be taken into account.

Example 8. Let $R = q(x) \rightarrow p(x, y)$ and $Q = p(u, v) \wedge p(v, t)$. Let us start from $p(u, v)$: this atom is unifiable with $\mathit{head}(R)$ and $p(v, t)$ necessarily belongs to the same pre-unifier (if any) because v is mapped to the existential variable y; however, $\{p(u, v), p(v, t)\}$ is not unifiable with $\mathit{head}(R)$ because, since v occurs at the first and at the second position of a p atom, x and y should be unified, which is not possible since y is an existential variable; thus $p(u, v)$ does not belong to any pre-unifier with R. However, $p(v, t)$ still needs to be considered. Let us start from it: $p(v, t)$ is unifiable with $\mathit{head}(R)$ and forms its own piece because its single variable t mapped to an existential variable is not shared with another atom. There is thus one (most general) piece-unifier of Q with R, namely $(\{p(v, t)\}, \{(v, x), (t, y)\})$.

More precisely, Algorithm 3 first builds the subset A of atoms in Q with the same predicate as $\mathit{head}(R)$. While A has not been emptied, it initializes a set Q' by picking an atom a in A, then repeats the following steps:

1. compute the most general pre-unifier of the current Q' with $\mathit{head}(R)$ if it exists; if there is no pre-unifier, the attempt with a fails;
2. if the found pre-unifier satisfies the condition on $\mathit{sep}(Q')$, then it is a single-piece unifier, and all the atoms in Q' are removed from A;
3. otherwise, the algorithm tries to extend Q' with all atoms from Q containing a variable from $\mathit{sep}(Q')$ that is mapped to an existential variable by the pre-unifier; if these atoms are in A, Q' can grow, otherwise the attempt with a fails.

Algorithm 3. Compute all most general single-piece unifiers

Data: a CQ Q and an atomic-head rule R
Result: the set of most general single-piece unifiers of Q with R
begin
  $U \leftarrow \emptyset$; `// resulting set`
  $A \leftarrow \{a \in Q \mid predicate(a) = predicate(\mathit{head}(R))\}$;
  while $A \neq \emptyset$ **do**
    $a \leftarrow$ choose an atom in A ;
    $Q' \leftarrow \{a\}$;
    $u \leftarrow$MostGeneralPreUnifier($Q' \cup \mathit{head}(R)$) ;
    while $u \neq \mathit{Fail}$ and $\mathit{sep}(Q') \setminus T_Q(u) \neq \emptyset$ **do**
      $Q'' \leftarrow \{a' \in Q \mid a'$ contains a variable in $\mathit{sep}(Q') \setminus T_Q(u)\}$;
      if $Q'' \subseteq A$ **then**
        $Q' \leftarrow Q' \cup Q''$;
        $u \leftarrow$ MostGeneralPreUnifier($Q' \cup \mathit{head}(R)$)
      else
        $u \leftarrow Fail$
    if $u \neq \mathit{Fail}$ **then**
      $U \leftarrow U \cup \{u\}$;
      $A \leftarrow A \setminus Q'$
    else
      $A \leftarrow A \setminus \{a\}$
  return U

5 First Experiments and Perspectives

The global backward chaining algorithm (cf. Algorithm 1), based on most general single-piece unifiers (cf. Algorithm 3), has been implemented in Java. First experiments have been made with the same rules and queries as in [GOP11]. The considered sets of rules are translations from ontologies expressed in DL-Lite$_\mathcal{R}$ developed in several research projects, namely ADOLENA (A), STOCKEXCHANGE (S), UNIVERSITY (U) and VICODI (V). See [GOP11] for more details. The obtained rules have atomic head and body, which corresponds to the linear Datalog+/- fragment. Queries are canonical

examples coming from projects in which the ontologies have been developed. For these first experiments, we compared our prototype to the NY* prototype, dedicated to linear Datalog+/- and part of the Nyaya system [GOP11]. The running time of both implementations are comparable. Concerning the sizes of the rewritings of the sample queries (*i.e.* the cardinalities of the output sets), they are equal for ontologies S, U and V, but not for ontology A (cf. Table 1, columns "final size"). Note that in [GOP11] the size of the rewritings output by NY* was already shown to be smaller than the one obtained with Requiem and QuOnto with substantial differences in some cases. Surprisingly, none of these systems computes a rewriting set of minimal size.

Table 1. Results with Nyaya and Piece-Based Rewriting

		NY*	Piece-Based Rewriting		
		final size	final size	# explorated	# generated
A	Q_1	249	27	457	1307
	Q_2	94	50	1598	4658
	Q_3	104	104	4477	13871
	Q_4	456	224	4611	15889
	Q_5	624	624	50508	231899
S	Q_1	6	6	6	9
	Q_2	2	2	48	256
	Q_3	4	4	64	536
	Q_4	4	4	240	1760
	Q_5	8	8	320	3320
U	Q_1	2	2	5	4
	Q_2	1	1	42	148
	Q_3	4	4	48	260
	Q_4	2	2	2196	9332
	Q_5	10	10	100	1280
V	Q_1	15	15	15	14
	Q_2	10	10	10	9
	Q_3	72	72	72	117
	Q_4	185	185	185	328
	Q_5	30	30	30	59

These first experimental results need to be extended by considering larger and more complex queries and rule bases, as well as comparing to other systems based on query rewriting. The size of the rewriting set should not be a decisive criterion (indeed, assuming that the systems are sound and complete, a minimal rewriting set is obtained by selecting most general elements, cf. Theorem 2). Therefore, other criteria have to be taken into account, such as the running time or the total number of CQs built during the rewriting process. As a first step in this direction, we indicate in Table 1 the number of explorated CQs (# explorated) and of generated CQs (# generated) with our system. The generated rewritings are all the rewritings built during the rewriting process (excluding the initial Q and possibly including some multi-occurrences of the same rewritings). Since we eliminate the subsumed rewritings at each step of the breadth-first algorithm, only some of the generated rewritings at a given step are explored at the next step.

Finally, our backward chaining mechanism is yet far from being optimized. Indeed, we have greatly simplified the unification operation —conceptually and algorithmically, which is important in itself— but in a way we have pushed the complexity into the composition of several rewritings. The question of whether it is worthwhile, when rules do not have atomic heads, to deal directly with them, still needs to be addressed.

Acknowledgements. We thank Giorgio Orsi for providing us with rule versions of the ontologies.

References

[Baa03] Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI 2003, pp. 325–330 (2003)

[BLM10] Baget, J.-F., Leclère, M., Mugnier, M.-L.: Walking the decidability line for rules with existential variables. In: KR 2010, pp. 466–476. AAAI Press (2010)

[BLMS09] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: Extending decidable cases for rules with existential variables. In: IJCAI 2009, pp. 677–682 (2009)

[BLMS11] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)

[BMRT11] Baget, J.-F., Mugnier, M.-L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: IJCAI 2011, pp. 712–717 (2011)

[CGK08] Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: KR 2008, pp. 70–80 (2008)

[CGL$^+$07] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)

[CGL09] Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: PODS 2009, pp. 77–86 (2009)

[CGP10] Calì, A., Gottlob, G., Pieris, A.: Query Answering under Non-guarded Rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)

[GOP11] Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE 2011, pp. 2–13 (2011)

[GS12] Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: KR 2012 (2012)

[KKZ11] Kikot, S., Kontchakov, R., Zakharyaschev, M.: Polynomial Conjunctive Query Rewriting under Unary Inclusion Dependencies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 124–138. Springer, Heidelberg (2011)

[KLMT12] König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: A Sound and Complete Backward Chaining Algorithm for Existential Rules. Technical Report RR-12016, LIRMM, GraphIK - INRIA Sophia Antipolis (2012)

[KR11] Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI 2011, pp. 963–968 (2011)

[Mug11] Mugnier, M.-L.: Ontological Query Answering with Existential Rules. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 2–23. Springer, Heidelberg (2011)

[PUHM09] Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient Query Answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)

[RA10] Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR 2010 (2010)

[SM96] Salvat, E., Mugnier, M.-L.: Sound and Complete Forward and Backward Chainings of Graph Rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS (LNAI), vol. 1115, pp. 248–262. Springer, Heidelberg (1996)