UCQ-rewritings for Disjunctive Knowledge and Queries with Negated Atoms

Enrique Matos Alfonso,[*] Alexandros Chortaras and Giorgos Stamou
Electrical and Computer Engineering, National Technical University of Athens, Greece
E-mails: gardero@image.ntua.gr, achort@cs.ntua.gr, gstam@cs.ntua.gr

Editor: Bernardo Cuenca Grau, Oxford University, UK
Solicited reviews: Michael Morak, Institut für Angewandte Informatik, Austria; Three anonymous reviewers

Abstract. In this paper, we study the problem of query rewriting for disjunctive existential rules. Query rewriting is a well-known approach for query answering on knowledge bases with incomplete data. We propose a rewriting technique that uses negative constraints and conjunctive queries to remove the disjunctive components of disjunctive existential rules. This process eventually generates new non-disjunctive rules, i.e., existential rules. The generated rules can then be used to produce new rewritings using existing rewriting approaches for existential rules. With the proposed technique we are able to provide complete UCQ-rewritings for union of conjunctive queries with universally quantified negation. We implemented the proposed algorithm in the COMPLETO system and performed experiments that evaluate the viability of the proposed solution.

Keywords: Disjunctive Rules, Queries with Negation, Backward Chaining and Query Rewriting.

1. Introduction

Rules are very important elements in knowledge-based systems and incomplete databases [21]; they allow us to perform query answering over incomplete data and come up with complete answers. There are two main approaches to perform query answering in the presence of rules, which depend on the way we use the rules. The forward chaining approach [26] applies the rules on the facts in order to produce new facts. On the other hand, the backward chaining approach [20] uses the rules to translate the input query into a set of queries (called the rewriting of the initial query) that also encode answers of the initial query. Both approaches allow us to infer answers that cannot be extracted from the initial data.

Example 1.1. *Let us consider a rule that defines the grand-parent relationship between two people based on the parent relationship*

$$r = \forall X \forall Y \, parent(X, Z) \wedge parent(Z, Y) \rightarrow grand\text{-}parent(X, Y)$$

and the information that

$$parent(ana, maria) \wedge parent(maria, julieta).$$

Traditional database systems would fail to entail that the query $Q = \exists Z \, grand\text{-}parent(ana, Z)$ holds, because it is not stated explicitly in the data. However, since the hypothesis of the rule r holds, we can infer (by forward application of the rule) that the fact $grand\text{-}parent(ana, julieta)$ also holds. Adding this new information allows traditional database systems to conclude that Q holds. Moreover, the rule r can also be applied to Q in a backward manner to derive additional queries that provide answers to the question expressed in the original query, i.e.,

$$Q' = \exists Z \exists Y \, parent(ana, Z) \wedge parent(Z, Y).$$

Forward chaining allows efficient query answering in systems where data is constant and queries change frequently. However, the size of the stored data can grow excessively, and the method is not appropriate for frequently changing data. For some ontologies, this approach does not always terminate [26], and may keep

[*]Corresponding author. E-mail: gardero@image.ntua.gr.

generating constantly new data. Backward chaining, on the other hand, is ideal for constant queries and changing data, although the size of the rewriting can be exponential with respect to the size of the initial query [19] or in some cases a finite rewriting may not exist. In both approaches, the application of the rules does not always terminate. Furthermore, having no restriction on the expressivity of the rules or having negated atoms on the query makes the query entailment problem undecidable [7].

In this paper, we focus on the entailment problem of conjunctive queries with negated atoms ($\mathsf{CQ}^{\neg}$) in the framework of disjunctive existential rules based on first-order logic (FOL) without function symbols.

Disjunctive existential rules allow the representation of very expressive knowledge in FOL, e.g.,

$$\forall X \exists Y \, \textit{is-parent}(X) \rightarrow \textit{father}(X, Y) \lor \textit{mother}(X, Y),$$

where disjunction and existentially quantified variables can appear on the right-hand side of the implication. Conjunctive query entailment is undecidable in the case of existential rules with disjunction. However, under some restrictions the problem can become decidable [7, 15]. To the best of our knowledge, existing research on existential rules with disjunction is only based on forward chaining algorithms [13, 15] or Disjunctive Datalog rewriting [3, 9, 11, 16].

Conjunctive queries with negation let us define the counterexamples of disjunctive rules, e.g.,

$$\exists X \forall Y \forall Z \, \textit{person}(X) \land \neg \textit{married}(X, Y) \land \neg \textit{parent}(X, Z)$$

describes when the following rule does not hold

$$\forall X \exists Y \exists Z \, \textit{person}(X) \rightarrow \textit{married}(X, Y) \lor \textit{parent}(X, Z).$$

Here we use the open world assumption, where the negation is associated to the "cannot" semantics, and the example query expresses the question of whether there is a person that cannot be married and cannot be a parent. We also consider *universally quantified negation* [1, 27, 28], i.e., variables that are only present in negated atoms are universally quantified.

The entailment problem for $\mathsf{CQ}^{\neg}$ is undecidable even for very simple types of rules [18, 28]. On the other hand, the use of guarded negation in queries is proven to be decidable over frontier-guarded existential rules [10]. Yet, the existing query rewriting-based approaches in the literature [14, 22, 23] that propose implementations and experiments only deal with queries that introduce negation in very limited ways.

Having existential variables, disjunction and queries with negated atoms makes the entailment problem even more difficult. However, by using these expressive resources in a smart way we can get very interesting and useful decidable fragments.

Particularly, in this paper we are interested in solving the entailment of a union of conjunctive queries with universally quantified negation by rewriting it into a union of conjunctive queries without negation (UCQ), called a UCQ-rewriting, where each element in it is a rewriting of the initial $\mathsf{UCQ}^{\neg}$.

Related Work Rosati studies query answering with respect to *description logics* (DLs) [28] and with respect to relational databases with *integrity constraints* (ICs) [27]. Extensions that allow safe negation and universally quantified negation are considered. The author provides a set of decidability, undecidability and complexity results for answering different types of queries with respect to various classes of DL knowledge bases and various combinations of ICs. In general, his results show that answering relational queries is unfeasible in many DLs and IC languages. The author considers unions of conjunctive queries with universally quantified negation, and shows that answering queries of this class is undecidable in every DL fragment and even in the absence of ICs.

In [20], König et al. define a generic rewriting procedure for conjunctive queries with respect to existential rules which takes as a parameter a rewriting operator, i.e., a function which takes as input a CQ and a set of existential rules and outputs a set of CQs. They also define the properties of rewriting operators that ensure the correctness and completeness of their algorithm. The authors prove that piece unification provides a rewriting operator with the desired properties. Finally, they provide an implementation of their algorithm together with some experiments. In this paper, we extend their definition of piece unification to deal also with disjunctive existential rules. We also extend their rewriting algorithm for existential rules into a sound and complete algorithm that also supports disjunctive existential rules and queries with negated atoms. However, the algorithm does not always terminate.

In [12], Bourhis et al. present a study of the complexity of query answering with respect to guarded disjunctive existential rules. The problem is 2EXPTIMEhard even for the simplest guarded-based class of disjunctive existential rules. Different types of UCQs are also considered by the authors in order to reduce the complexity of the problem. However, the considered query languages do not have a positive impact on the complexity of the problem. The only significant decrease in the complexity was found for atomic queries and linear disjunctive existential rules. The authors proved that for fixed atomic queries and fixed linear disjunctive existential rules the problem is in the AC_0 complexity class by establishing that the problem is first-order rewritable, i.e., it can be reduced to the problem of evaluating a first-order query over a database. The rewriting algorithm that we propose in this paper terminates for the fragment considered by Bourhis et al. The techniques and results presented in [12] can be used as a generic tool to study the complexity of query answering for several fragments of description logics.

Some other research papers [3, 9, 11, 16] focus on the transformation of the query answering problem with respect to guarded (disjunctive) existential rules into a query answering problem with respect to (disjunctive) Datalog programs by getting rid of existential variables. In [3], Ahmetaj et al. provide a transformation of the problem that yields a polynomial size (disjunctive) Datalog program when the maximal number of variables in the (disjunctive) existential rules is bounded by a constant. The translations proposed by other authors [9, 11, 16] are of exponential size.

Outline of our contributions. We introduce a restricted form of FOL resolution (constraint resolution) that is sound and refutation complete, and where the subsumption theorem holds if the consequences are clauses without positive literals. The number of choices at the moment of selecting clauses to perform constraint resolution steps is reduced with respect to the number of choices we have for the case of unrestricted resolution steps.

Based on the constraint resolution method, we propose an algorithm to compute a UCQ-rewriting for an input UCQ with respect to (disjunctive) existential rules and constraints. The algorithm can also compute UCQ-rewritings for conjunctive queries with universally quantified negation by converting queries with negation into rules. The algorithm is sound, provides a complete UCQ-rewriting, and terminates for the cases where there is a finite and complete UCQ-rewriting of the input query with respect to the (disjunctive) existential rules and the negative constraints. We also present two theorems with sufficient conditions for the termination of the algorithm. One case requires disconnected disjunctive existential rules (rules where the body and the head do not share variables) and existential rules that yield a finite and complete UCQ-rewriting for any UCQ (*finite unification set*). The other case is based on queries and knowledge bases where all the elements are linear (rules with at most one atom in the body and CQs with at most one positive atom).

Additionally, we consider unions of conjunctive queries with negated atoms and answer variables (denoted as $\mathsf{UCQa}^{\neg}$) with respect to existential rules and constraints. The proposed algorithm is modified in order to compute only the *deterministic* UCQ-rewritings, i.e. the UCQ-rewritings that lead to *certain answers* or that check the consistency of the knowledge base. We prove that the modified algorithm terminates when there is a finite and complete deterministic UCQ-rewriting of the input $\mathsf{UCQa}^{\neg}$ with respect to the existential rules and constraints of the knowledge base. We also present two theorems with sufficient conditions for the termination of the modified rewriting algorithm. One case requires the existential rules to be a finite unification set and the input queries to have the variables in both the positive and negated atoms as part of the answer variables of the query. The second case requires the queries to have only one positive atom and an arbitrary number of negated atoms; also the existential rules and constraints are required to have only one atom in the body.

Finally, we implemented the proposed algorithm for rewriting conjunctive queries with negated atoms and answer variables with respect to existential rules in the COMPLETO system, and performed experiments to evaluate the viability of the proposed solution.

Paper organization. Section 2 introduces the relevant theory needed to understand the rest of the paper. Subsection 2.1 introduces concepts related to FOL resolution and Subsection 2.2 presents the disjunctive existential rules framework. Section 3 introduces a novel type of resolution (Subsection 3.1) and a novel backward chaining rewriting algorithm for disjunctive existential rules (Subsection 3.2). Additionally, Subsection 3.3 explores some termination conditions for the proposed rewriting algorithm and Subsection 3.4 proposes a modification of the algorithm that computes only the deterministic UCQ-rewritings of $\mathsf{UCQa}^{\neg}$s. Section 4 describes an implementation of the proposed rewriting algorithm. Experiments describing the performance of our implementation are presented in Subsection 4.1. Finally, Section 5 presents an overview of the paper with some conclusions.

2. Preliminaries

In this section we introduce the basic concepts related to the FOL resolution process. Resolution is the base of all the reasoning processes we describe in this paper. All steps in high level reasoning processes can be tracked down to sequences of resolution steps that ensure the correctness. We also describe the framework of disjunctive existential rules and present the definition of conjunctive queries with negated atoms.

2.1. First-Order Logic Resolution

We assume the reader is familiar with the standard definition of first-order logic formulas. In this paper, we focus on FOL formulas without function symbols over a finite set of predicate names and a finite set of constant symbols. We also adopt the standard definitions for the entailment and equivalence of formulas, as they are rarely modified in the literature. We refer the reader to [25] in case a background reading is needed.

Because in the following we will often need to modify formulas, in order to make such modifications more compact and easier to understand we introduce the definition of conjunctive (disjunctive) set formulas (CSFs and DSFs). However, the reader should be familiar with the notation because it is often used in FOL when we write rules and clauses.

Definition 2.1 (Conjunctive (Disjunctive) Set Formula)**.** *A conjunctive (disjunctive) set formula (*CSF *and* DSF*, respectively) is a set of formulas* $\{F_1, \ldots, F_n\}$*, where the set of formulas is interpreted as a conjunction (disjunction) of the formulas in the set, i.e.,* $F_1 \wedge \ldots \wedge F_n \quad (F_1 \vee \ldots \vee F_n)$.

For a given set of formulas $\{F_1, \ldots, F_n\}$*, a* CSF *containing these formulas is denoted as* $F_1, \ldots, F_n$*. On the other hand, a* DSF *is denoted by* $[F_1, \ldots, F_n]$*. Finally, an empty* CSF *is denoted by and is equivalent to* $\top$*, and an empty* DSF *is denoted by and is equivalent to* $\bot$.

Note that in case a CSF is a sub-formula of another set formula we use parenthesis to avoid ambiguity, e.g., $[(A, B), D]$ is equivalent to $(A \wedge B) \vee D$.

Set operators can then be used to combine set formulas of the same type and obtain a new set formula. Moreover, elements of a set formula that are equivalent can be *collapsed* into a single element, e.g.,

$$[\neg A] \cup [A \rightarrow \bot, B] \equiv [\neg A, A \rightarrow \bot, B] \equiv [\neg A, B].$$

The following axioms can be easily proven:

1. A DSF and a CSF of one element are equivalent:
$$[F] \equiv F.$$
2. De Morgan's Laws allow changing a DSF to a CSF, and vice-versa, using negation:
$$\neg[F_1, \ldots, F_n] \equiv \neg F_1, \ldots, \neg F_n$$
$$\neg(F_1, \ldots, F_n) \equiv [\neg F_1, \ldots, \neg F_n].$$
3. Let B and F be two CSFs (DSFs) such that $B \in F$. Then, $F \equiv (F \setminus \{B\}) \cup B$. Replacing F by the equivalent formula $(F \setminus \{B\}) \cup B$ is referred to as *flattening* the formula F.

We model the entailment operator in FOL assuming that on the left-hand side of the entailment symbol we have a CSF $A_1, \ldots, A_n$ of axioms A_i:

$$A_1, \ldots, A_n \vDash F.$$

The right-hand side of the entailment operator is assumed to be a DSF. For CSFs B and A (where $B \subseteq A$) and DSFs C and F (where $C \subseteq F$), we can prove that $A \vDash F$ if and only if $A \setminus B \vDash F \cup \neg B$; likewise $A \vDash F$ if and only if $A \cup \neg C \vDash F \setminus C$.

In the following, we recall the definitions of some of the concepts that are needed for the theory presented in this paper.

A *term* is a constant, a variable or an expression $f(t_1, \ldots, t_m)$ where f is a *function* symbol and the arguments t_i are terms. However, in this paper we focus only on *simple* terms, i.e., either variables or constants. We only consider *Skolem* function symbols internally in order to get rid of existentially quantified variables.

An *atom* is a formula $a(t_1, \ldots, t_n)$ where a is a *predicate* of *arity n* (denoted by a/n). The arguments t_i of the atom are terms. A *literal* is an atom or a negated atom. The *complement* $\bar{l}$ of a literal l is $\neg a(t_1, \ldots, t_n)$ if $l = a(t_1, \ldots, t_n)$, and $a(t_1, \ldots, t_n)$ if $l = \neg a(t_1, \ldots, t_n)$. A literal is *positive* (or of *positive polarity*) if it is a non-negated atom, and *negative* (or of *negative polarity*) if it is a negated atom. Two literals are *complementary* if one is the complement of the other.

A formula is *ground* if it contains no variables. In a formula, the variables can be *universally quantified*, *existentially quantified*, or *free*. A formula without free variables is *closed*. The set of all the variables that appear in an expression F is denoted by $vars(F)$. We denote a sequence of variables $X_1, \ldots, X_n$ using a boldface character (e.g., $\mathbf{X}$).

A *substitution* $\theta = \{X_1 \leftarrow t_1, \dots X_n \leftarrow t_n\}$ is a finite mapping of variables X_i to terms t_i. The result of applying a substitution θ on an expression F is the expression $F\theta$ obtained by replacing in F every occurrence of every variable X_i by the term t_i.

Let F be an expression and θ the substitution $\{X_1 \leftarrow Y_1, \dots X_n \leftarrow Y_n\}$. We say θ is a *renaming substitution* for F, if each X_i occurs in F, and $Y_1, \dots, Y_n$ are distinct variables such that each Y_i is either equal to some X_j in θ, or Y_i does not occur in F. The *composition* of two substitutions θ and σ is a new substitution $\theta\sigma$ that when applied to any expression, has the same effect as applying those substitutions in sequence (i.e., first θ and after σ). A substitution θ is *more general than* another substitution σ if there exists another substitution γ such that $\sigma = \theta\gamma$.

A substitution θ is a *unifier* for a set of expressions $S = \{F_1, \dots, F_n\}$ iff $F_1\theta = F_2\theta, \dots, F_{n-1}\theta = F_n\theta$. The expressions in S are said to be *unifiable* if there is a unifier for S. The *most general unifier* (*mgu*) of the expressions in S is denoted by $mgu(S)$, and it is a unifier for S that is more general than any other unifier of the expressions in S. Even if it is possible for the same set S to have more than one *mgu*, they are unique up to variable renaming.

A *hypergraph* is a tuple $\langle A, E\rangle$, where A is a set of elements called nodes or vertices, and E is a set of non-empty subsets of A called *hyperedges*. We can represent a CSF of atoms as a hypergraph using the set of variables that appear in the arguments of the atoms as nodes. Each atom in the formula represents a hyperedge that connects its variables. Note that hyperedges are defined as sets of nodes and there is no notion of direction between the nodes. With this representation we can define some properties for CSFs of atoms.

The *cardinality* of a CSF of atoms F is the number of variables in the formula: $card(F) = |vars(F)|$. Two variables u and v in a CSF of atoms F are *connected* iff they both belong to the same atom ($\exists A \in F | \{v, u\} \subseteq vars(A)$), or if there is another variable z in F that is connected to both u and v.

A CSF of atoms F is *connected* if all the atoms in it contain variables and all the variables are connected to each other. An atom that has only constants in its arguments is a connected formula that is not connected to any other atom and has a cardinality of zero. It is represented by an empty hypergraph. The constants in the formula play no role in their hypergraph representation.

It follows that a CSF F can be partitioned into a set $\{U_1, \dots, U_n\}$ of connected CSFs such that if $v \in vars(U_i)$ is connected to $u \in vars(U_j)$, then $i = j$. If F is connected, this set contains only F. The *connected cardinality* of F is defined as the maximum cardinality of the connected CSFs in the partition of F and denoted by $card^*(F) = \max_i(card(U_i))$. The connected cardinality of a DSF $[F_1, \dots, F_m]$ is the maximum connected cardinality of the formulas F_i, i.e., $card^*([F_1, \dots, F_m]) = \max_i(card^*(F_i))$.

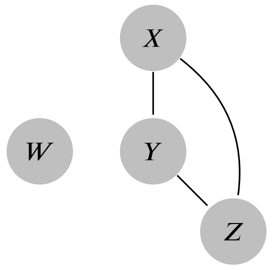

Figure 1. Hypergraph corresponding to a CSF.

Example 2.1. *The* CSF

$$F = parent(X, Y), parent(Y, Z), \\ grand\text{-}parent(X, Z), person(W)$$

is represented by the hypergraph in Figure 1. We can split F in two connected components

$$\{(person(W)), (parent(X, Y), \\ parent(Y, Z), grand\text{-}parent(X, Z))\}.$$

The connected cardinality of F is 3, the cardinality of the greatest connected component.

Lemma 2.1. *Let G be a* CSF*, and let $\{U_1, \dots, U_n\}$ be the partition of a given* CSF *F of atoms into connected* CSF*s. Then,*

$$G \models F \;\; iff \;\; G \models U_i \, for\ every \;\; U_i.$$

Proof. A detailed proof is given by Tessaris [29]. However, the reader can clearly see that since no variable is shared between the connected components U_i, the assignments for the variables that make each U_i valid in G can be combined without introducing conflicts on the values that each variable gets. □

Lemma 2.2. *Let k be a natural number. There are a finite number of equivalence classes of* CSF*s of atoms with connected cardinality of at most k that can be constructed using a finite set of predicates and constants.*

Proof. It is easy to check that two CSFs of atoms are equivalent iff they are unifiable by a renaming substitution. Since we have finitely many predicates and constant symbols, and at most k different variables, we can combine them in a finite number M of ways to form a connected CSF. In a CSF consisting of more than M connected CSFs we know that some of the connected components are renamings of others, and keeping only one of them is enough for the evaluation of F according to Lemma 2.1. Hence, there are at most 2^M different equivalence classes. □

A *clause* C is a DSF $[l_1, \dots, l_n]$ of literals l_i, where all the variables are universally quantified. A contradiction is represented by the *empty clause* $\bot$. A formula F is in conjunctive normal form (CNF) if it is a CSF of clauses, i.e,

$$F = [l^1_1, \dots, l^1_{n_1}], \dots, [l^m_1, \dots, l^m_{n_m}].$$

Every FOL formula can be transformed into an equisatisfiable CNF formula using variable standarization, Skolemization, De Morgan's laws, and the distributivity of the conjunction and disjunction logical operators.

An *instance* of a clause C is the result of applying a substitution θ to the clause, i.e., $C\theta$. If two or more literals of the same polarity in a clause C are unifiable and θ is their most general unifier, then the clause $C\theta$ is called a *factor* of C, and the process of applying θ is called *factorization*.

Definition 2.2 ((Binary) Resolution Rule). *Let C_1 and C_2 be two clauses with no variables in common, and let $l_1 \in C_1$ and $l_2 \in C_2$ be complementary literals with respect to a most general unifier $\sigma = mgu(\{l_1, \bar{l}_2\})$. The* binary resolvent *of C_1 and C_2 with respect to the literals l_1 and l_2 is the clause:*

$$C_1 \cup_r C_2 = (C_1\sigma \setminus [l_1\sigma]) \cup (C_2\sigma \setminus [l_2\sigma]).$$

C_1 and C_2 are said to be clashing clauses. *A* resolvent *$C_1 \cup_r C_2$ of C_1 and C_2 is a binary resolvent $C_1\sigma_1 \cup_r C_2\sigma_2$ of factors $C_i\sigma_i$ of the two clauses.*

It is easy to show that resolution is sound, i.e., $C_1, C_2 \models C_1 \cup_r C_2$. Consequently, the resolution rule can be used to deduce new clauses and to prove that a formula is unsatisfiable if we are able to derive the empty clause.

Definition 2.3 (Resolution Derivation (Refutation)). *Let Σ be a set of clauses and C a clause. A* (resolution) derivation *of C from Σ is a finite sequence of clauses $R_1, \dots, R_k = C$, such that each R_i is either in Σ, or a resolvent of two clauses in $\{R_1, ..., R_{i-1}\}$. If such a derivation exists, we write $\Sigma \vDash_r C$, and say that C can be derived from Σ. A derivation of the empty clause $\bot$ from Σ is called a* refutation *of Σ. The steps of a resolution derivation are the resolution operations performed to obtain the resolvents in the sequence.*

$$\begin{tikzcd}
{[a(X)]} & & {[\neg a(X), \neg b(X)]} & & \\
& {[\neg b(X)]} \arrow[ul] \arrow[ur] & & {[\neg a(X), b(X)]} & \\
& & {[\neg a(X)]} \arrow[ul] \arrow[ur] & & {[a(X)]} \\
& & & \bot \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 2. A derivation tree.

Sometimes it is useful to know which clauses were used to produce a resolvent R_i in a resolution derivation. In such cases, a graph or tree representation can be helpful.

Definition 2.4 (Derivation (Refutation) Graph). *Let Σ be a set of clauses and C a clause such that $\Sigma \vDash_r C$. A* derivation (refutation) graph *of C from Σ is a directed graph where the nodes are the clauses from the derivation $\Sigma \vDash_r C$, and where there is an edge from each resolvent to the clauses used in the resolution step by which it was derived.*

If we only include in the graph the last clause C and the clauses that are used in at least one resolution step of the derivation, we can see the derivation graph as a tree, in which the last resolvent C is the *root* of the tree and the leaves are clauses from Σ. Indeed, by cloning nodes with more than one input edges (i.e., clauses used in several resolution steps) we can transform the derivation graph into a *derivation tree*. It is more convenient to draw such derivation trees upside-down (Figure 2).

Theorem 2.1 (Soundness of Derivation). *Let Σ be a set of clauses, and C a clause. If $\Sigma \vDash_r C$, then $\Sigma \vDash C$.*

Proof. This is a straightforward consequence of the soundness of the resolution rule. □

A clause logically implies any instance of it, possibly extended with more literals. This follows directly from the properties of the disjunction operator and the universal quantification.

Definition 2.5 (Subsumption)**.** *Let C and D be two clauses. We say that C* subsumes *D if there exists a substitution θ such that $C\theta \subseteq D$.*

Definition 2.6 (Deduction)**.** *Let Σ be a set of clauses and C a clause. We say that there exists a* deduction *of C from Σ, and write $\Sigma \vDash_r^d C$, if C is a tautology, or if there exists a clause D such that $\Sigma \vDash_r D$ and D subsumes C. If $\Sigma \vDash_r^d C$, we say that the clause C can be* deduced *from Σ.*

Resolution steps or derivations involving ground instances of clauses (i.e., clause instances that do not contain variables) ensure that there are corresponding resolution steps or derivations involving the non-ground version of the clauses. This process is known as *lifting* a resolution step or derivation.

In the text below we recall known theorems taken from the literature [25].

Theorem 2.2 (Lifting Lemma)**.** *Let the clauses C'_1, C'_2 be ground instances of the clauses C_1, C_2, respectively. Let C' be a ground resolvent of C'_1 and C'_2. Then there is a resolvent of the clauses C of C_1 and C_2 such that C' is a ground instance of C.*

Theorem 2.3 (Derivation Lifting)**.** *Let Σ be a set of clauses, and Σ' a set of ground instances of clauses from Σ. Suppose $R'_1, ..., R'_k$ is a derivation of the clause R'_k from Σ'. Then there exists a derivation $R_1, ..., R_k$ of the clause R_k from Σ, such that R'_i is an instance of R_i, for each $i \in \{1, \dots, k\}$.*

Resolution derivations allow us to infer clauses that are logical consequences of an initial knowledge in a complete way.

Theorem 2.4 (Subsumption Theorem)**.** *Let Σ be a set of clauses, and C a clause. Then $\Sigma \vDash C$ iff $\Sigma \vDash_r^d C$.*

Theorem 2.5 (Refutation Completeness of Resolution)**.** *Let Σ be a set of clauses. Then Σ is unsatisfiable iff $\Sigma \vDash_r \perp$.*

Performing resolution steps in a *breath-first* manner ensures that we will find the empty clause for unsatisfiable formulas. However, for satisfiable formulas, we may never stop generating new clauses that are not subsumed by the already generated clauses. In general, resolution allows us to define algorithms that provide sound and complete results, but we cannot ensure termination for all FOL formulas.

The resolution operator $\cup_r$ has some useful properties that allow us to transform derivations without affecting the consequence.

Property 2.1 (Symmetry)**.** *If C_1 and C_2 are clashing clauses, then their resolvent clause can be computed in a symmetric way:*

$$C_1 \cup_r C_2 \equiv C_2 \cup_r C_1.$$

Property 2.2 (Distributivity)**.** *If C_1, C_2 and C_3 are clauses such that C_3 resolves with the resolvent of C_1 and C_2 using literals from both C_1 and C_2, then the following distributivity property holds:*

$$(C_1 \cup_r C_2) \cup_r C_3 \equiv (C_1 \cup_r C_3) \cup_r (C_2 \cup_r C_3). \quad (1)$$

Note that on right-hand side of (1) the literals used in the resolution steps with respect to C_3 ($C_1 \cup_r C_3$ and $C_2 \cup_r C_3$) need to be the same that are used in the left-hand side.

Property 2.3 (Commutativity)**.** *If C_1, C_2 and C_3 are clauses such that C_3 resolves with the resolvent of C_1 and C_2 using only literals from C_1, then the following commutativity property holds:*

$$(C_1 \cup_r C_2) \cup_r C_3 \equiv (C_1 \cup_r C_3) \cup_r C_2.$$

Proving the above properties is straightforward if we consider them over ground instances of the clauses and track the set operations on ground literals. As a consequence of Theorem 2.3, the properties also hold for general resolution over non-ground clauses.

Example 2.2. *We will illustrate the distributivity property for $C_1 = [a(X), b(X)]$, $C_2 = [a(X), \neg b(X)]$ and $C_3 = [\neg a(X), c(X)]$. On the left-hand side of (1) we have the following resolution steps:*

$$\begin{aligned}(C_1 \cup_r C_2) &= [a(X)]\\ (C_1 \cup_r C_2) \cup_r C_3 &= [a(X)] \cup_r [\neg a(X), c(X)]\\ &= [c(X)]\end{aligned}$$

and on the right-hand side

$$\begin{aligned}&(C_1 \cup_r C_3) = [b(X), c(X)]\\ &(C_2 \cup_r C_3) = [\neg b(X), c(X)]\\ &(C_1 \cup_r C_3) \cup_r (C_2 \cup_r C_3)\\ &\qquad = [b(X), c(X)] \cup_r [\neg b(X), c(X)]\\ &\qquad = [c(X)].\end{aligned}$$

Clearly, $[c(X)] \equiv [c(X)]$.

2.2. Disjunctive Existential Rules and Conjunctive Queries with Negation Framework

A *conjunctive query* (CQ) is a CSF $l_1, \dots, l_n$ of positive literals l_i where all the variables (which we denote by $\mathbf{X}$) are existentially quantified, i.e., an expression of the form $\exists \mathbf{X}\, l_1, \dots, l_n$. Queries that allow negation in the literals l_i are called *conjunctive queries with negation* (CQ$^\neg$). All the variables $\mathbf{X}$ that appear in the positive literals of a CQ$^\neg$ are assumed to be existentially quantified. In order to avoid domain dependant queries we use *universally quantified negation* [1, 27, 28], i.e., all the variables $\mathbf{Z}$ that appear only in negative literals are assumed to be universally quantified: $\exists \mathbf{X} \forall \mathbf{Z}\, l_1, \dots, l_n$. Because the variable quantification rules are straightforward we omit quantifiers, e.g., instead of $\exists X \forall Y person(X), \neg married(X, Y)$ we write $person(X), \neg married(X, Y)$. The set of variables that appear in both positive and negative literals is called the *frontier* of the query. Note that for now we do not introduce the concept of *answer variables*. Therefore, the queries we define are normally known as *Boolean conjunctive queries*. Consequently, throughout the paper by conjunctive query we mean Boolean conjunctive query. A DSF of conjunctive queries (conjunctive queries with negation) is usually referred to as a *union of conjunctive queries* (UCQ) (*union of conjunctive queries with negation* (UCQ$^\neg$)). For a UCQ$^\neg$ Q, by $Q^{\neg k}$ we denote the set of CQ$^\neg$s in Q that contain exactly k negated atoms, and by $Q^{\neg\#}$ the set of CQ$^\neg$s in Q that contain two or more negated atoms. We use the term *query* to refer to either a CQ, CQ$^\neg$, UCQ or UCQ$^\neg$.

A *fact* is a CSF $l_1, \dots, l_n$ of positive literals l_i, where all variables are existentially quantified, e.g., $parent(ana, Y), parent(maria, Y)$. Existential quantifiers are again omitted.

A closer look at the definition of facts reveals that a fact is equivalent to a Boolean conjunctive query. However, facts are used to express existing knowledge, while queries represent questions, so they have different roles in the process of reasoning.

A *rule* is a closed formula of the form

$$\forall \mathbf{X}\, \exists \mathbf{Y}\, B \rightarrow H,$$

where the *body* B is a CSF of positive literals, and the *head* H is a DSF in which all $H' \in H$ are CSFs of positive literals. The set $\mathbf{X} = vars(B)$ contains the variables that appear in the body, and they are universally quantified. On the other hand, $\mathbf{Y} = vars(H) \setminus vars(B)$ are the variables the appear only in the head. They are existentially quantified, and they are called *existential variables*. The *frontier* of a rule is the set of variables that are present in both the body and head of the rule: $vars(B) \cap vars(H)$. We omit quantifiers when writing a rule.

A *disjunctive existential rule* is a rule with more than one disjoint in the head, i.e., $\|H\| > 1$. An *existential rule* is a rule with exactly one disjoint in the head. For simplicity we write the head of the existential rule as a CSF of atoms. A rule with an empty disjoint in the head is a *negative constraint*, i.e., $B \rightarrow \bot$. If it is clear from the context that we refer to a negative constraint, we can omit the "$\rightarrow \bot$" and write only the body B of the negative constraint. Sometimes we also refer to a negative constraint as a constraint.

We say that a CSF of atoms Q *depends* on a rule r iff there is a CSF of atoms F such that $F \not\vDash Q$ and $F, r \vDash Q$. A rule r_i depends on a rule r_j iff the body of r_i depends on r_j. The concept of rule dependencies allows us to define the *graph of rule dependencies* (GRD) [4], which is a graph where nodes are rules, and a directed edge between two nodes represents the existence of a dependency between the corresponding rules.

A *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{R}, \mathcal{D} \rangle$ is composed by a CSF $\mathcal{R}$ of rules and a CSF of facts $\mathcal{D}$. For a given set of rules $\mathcal{R}$, by $\mathcal{R}^\bot$ we denote the set of constraints in $\mathcal{R}$, by $\mathcal{R}^\exists$ the set of existential rules and by $\mathcal{R}^\vee$ the set of disjunctive existential rules. A knowledge base $\langle \mathcal{R}, \mathcal{D} \rangle$ is a *disjunctive* knowledge base (DKB) if $\mathcal{R}^\vee \neq \emptyset$.

Example 2.3 (Disjunctive Knowledge Base)**.** *Let us define an example* DKB *about family relationships.*

- *Facts:*

$$(parent(Y, ana), parent(Y, jane)), \quad sibling(ana, juan) \tag{2}$$

- *Existential rules:*

$$(sibling(X, Y) \rightarrow sibling(Y, X)), \tag{3}$$

$$(sibling(X, Y) \rightarrow parent(Z, X), parent(Z, Y)) \tag{4}$$

- *Negative constraints:*

$$(sibling(X, Y), parent(X, Y)), \tag{5}$$

$$(same\text{-}age(X, Y), parent(X, Y)), \tag{6}$$

$$(parent(X, Y), parent(Y, X)), \tag{7}$$

$$(parent(X, X)) \tag{8}$$

– *Disjunctive existential rules :*

$$\begin{aligned} &\textit{first-deg-relative}(X,Y) \rightarrow \\ &\quad [\textit{parent}(X,Y), \textit{parent}(Y,X), \\ &\quad\quad \textit{sibling}(X,Y)] \end{aligned} \tag{9}$$

Note that the existential rule (4) has an existential variable Z, that refers to an anonymous entity that is a parent of both siblings, i.e., if two people are siblings, they share a parent. Rule (3) states that the predicate sibling/2 is symmetric. We could also add symmetry for same-age/2. Fact (2) states that there is an anonymous entity Y that is a parent of both jane and ana. The negative constraints state the impossibility of a person being parent of his sibling (5) and also of a person of the same age (6). Additionally, with the negative constraints we express that parent/2 is asymmetric (7) and irreflexive (8). Finally, the disjunctive existential rule (9) defines the relation that represents the first degree relative concept [24]: a parent, a child (inverse of parent), or a sibling.

In this paper, we study the *query entailment* problem for disjunctive knowledge bases, i.e., the problem of knowing whether a query $\mathcal{Q}$ can be entailed from a disjunctive knowledge base $\langle\mathcal{R},\mathcal{D}\rangle$:

$$\mathcal{R},\mathcal{D} \models_? \mathcal{Q}. \tag{10}$$

In particular, we solve the entailment problem (10) by reducing it to the entailment of a UCQ $\mathcal{Q}'$ with respect to the set of facts $\mathcal{D}$, i.e., to the problem

$$\mathcal{D} \models_? \mathcal{Q}'.$$

We say that $\mathcal{Q}'$ is a UCQ-*rewriting* of $\mathcal{Q}$ with respect to $\mathcal{R}$ if for all set of facts $\mathcal{D}$ it holds that

$$\mathcal{D} \models \mathcal{Q}' \text{ implies } \mathcal{R},\mathcal{D} \models \mathcal{Q}. \tag{11}$$

The CQs in $\mathcal{Q}'$ are called CQ-*rewritings* of $\mathcal{Q}$ with respect to $\mathcal{R}$. If the converse of (11)

$$\mathcal{R},\mathcal{D} \models \mathcal{Q} \text{ implies } \mathcal{D} \models \mathcal{Q}'$$

also holds for all set of facts $\mathcal{D}$, we say that $\mathcal{Q}'$ is a *complete* UCQ-rewriting of $\mathcal{Q}$ with respect to $\mathcal{R}$. Note that according to our definition, a UCQ-rewriting may not be complete. In this respect, our definition follows the definition of UCQ-rewriting from [20] because we extend many of the concepts and algorithms proposed by the authors.

Finally, we may be interested in the values that some of the variables in $\mathcal{Q}$ take. However, this does not change the semantic definition of conjunctive queries and therefore, we discuss it later in Subsection 3.4.

3. Backward Chaining with Disjunctive Knowledge

In this section we first present *constraint resolution*, a novel type of resolution that is sound and refutation complete. Constraint resolution reduces the number of available choices in the resolution process by focusing on producing resolvents with a smaller number of positive literals. Constraint resolution is then translated into backward rewriting steps, allowing the definition of a rewriting algorithm for the framework of disjunctive existential rules that is able to solve the query entailment problem (10).

3.1. Constraint Resolution

The entailment problem (10) can be transformed into a consistency check problem

$$\mathcal{R},\mathcal{D},\neg\mathcal{Q} \models_? \bot, \tag{12}$$

which can then be solved using resolution refutation. Depending on the expressivity of the queries in $\mathcal{Q}$ their negation can yield new facts, negative constraints, existential rules or even disjunctive existential rules.

To apply resolution, we need to convert first the facts $\mathcal{D}$ and rules $\mathcal{R}$ of the DKB, as well as the negated query $\neg\mathcal{Q}$ to a CNF. In what follows, our purpose is to define a restricted resolution strategy in order to control the process of resolution among these clauses, so as to decrease the number of available choices every time we perform a resolution step. This might result in longer derivations, but the algorithm to generate them is simpler. The main goal of the restrictions we introduce is to make the resolution process focus on eventually generating resolvents without positive literals and any number of negative literals. The process can then continue by eliminating those remaining negative literals without introducing again positive literals.

Definition 3.1 (Positive/Negative Charge)**.** *The* positive (negative) charge $|C|^+$ ($|C|^-$) *of a clause C is the number of positive (negative) literals in the clause.*

Table 1
Properties of different types of clauses in the disjunctive existential rules framework.

Name	Properties
Rule clause (RC)	$\lvert C\rvert^{+} = 1 \land \lvert C\rvert^{-} \geq 1$
Fact clause (FC)	$\lvert C\rvert^{+} = 1 \land \lvert C\rvert^{-} = 0$
Constraint clause (CC)	$\lvert C\rvert^{+} = 0 \land \lvert C\rvert^{-} \geq 1$
Disjunctive RC (DRC)	$\lvert C\rvert^{+} \geq 2 \land \lvert C\rvert^{-} \geq 0$

According to the above definition, a *Horn clause C* is a clause with positive charge smaller than or equal to one, i.e., $|C|^{+} \leq 1$. The clause we obtain by converting to CNF a negative constraint or the negation of a CQ has zero positive charge. We call a clause with no positive literals a *constraint clause (*CC*)*. The Skolemized version of an existential rule can produce several clauses with one positive literal, while a disjunctive existential rule can give rise to several clauses with more than one positive literal. We call a clause with only one positive literal a *rule clause* (RC), and a clause with more than one positive literal a *disjunctive rule clause* (DRC). Facts generate ground clauses containing only one literal, which has positive polarity, and we call such clauses *fact clauses* (FCs). Note that existential variables in the facts are replaced by Skolem terms. Table 1 summarizes the properties that define the different types of clauses we may encounter when doing resolution on a DKB. As we can see, RCs, FCs and CCs are Horn clauses. Finally, a CQ$^{\neg}$ may produce a FC, RC or DRC, depending on its positive and negative charge.

Example 3.1. *From the* DKB *of Example 2.3 we obtain the following clauses:*

– *Fact clauses:*

$$[parent(f_0, ana)]$$
$$[parent(f_0, jane)]$$
$$[sibling(ana, juan)].$$

– *Rule clauses:*

$$[\neg sibling(X,Y), sibling(Y,X)]$$
$$[\neg sibling(X,Y), parent(f_2(X,Y),X)]$$
$$[\neg sibling(X,Y), parent(f_2(X,Y),Y)].$$

– *Constraint clauses:*

$$[\neg sibling(X,Y), \neg parent(X,Y))],$$
$$[\neg same\text{-}age(X,Y), \neg parent(X,Y)]$$
$$[\neg parent(X,Y), \neg parent(Y,X)]$$
$$[\neg parent(X,X)].$$

Table 2
Properties of the resolvent C_3 for different types of clauses C_1 and C_2.

C_1	C_2		
	CC	DRC	RC
FC	$\lvert C_3\rvert^{-} < \lvert C_1\rvert^{-}$	$\lvert C_3\rvert^{-} < \lvert C_1\rvert^{-}$	$\lvert C_3\rvert^{-} < \lvert C_1\rvert^{-}$
RC	$\lvert C_3\rvert^{+} = 0$	$1 \leq \lvert C_3\rvert^{+} \leq \lvert C_2\rvert^{+}$	$\lvert C_3\rvert^{+} = 1$
DRC	$\lvert C_3\rvert^{+} < \lvert C_1\rvert^{+}$	$\lvert C_3\rvert^{+} > \max(\lvert C_1\rvert^{+}, \lvert C_2\rvert^{+})$	
CC	does not exist		

– *Disjunctive rule clauses :*

$$[\neg first\text{-}deg\text{-}relative(X,Y), parent(X,Y), parent(Y,X), sibling(X,Y)].$$

Table 2 shows the properties of the resolvent for different types of clauses when performing a resolution step. From these properties follows that a resolution refutation should involve resolution steps with respect to fact clauses, because they always produce clauses with a smaller negative charge. Such resolution steps can be arranged so that they are performed in the last part of the resolution derivation. Additionally, this type of resolution steps is generally linked to *data retrieval* with respect to databases. For this reason, we mainly focus on the initial part of a rearranged resolution derivation, until it reaches a clause that has only negated atoms. Such a process is linked to producing CQ-rewritings, since the resulting clause corresponds to the negation of a conjunctive query.

As we can see from Table 2, in order to get derivations that produce clauses with a non-increasing positive charge, we need to avoid resolutions steps involving two DRCs, i.e., we need to use in every resolution step at least one Horn clause. However, if we focus on resolution steps where one of the clauses used is a CC, the resolvent will always have a smaller positive charge (See CC column on Table 2).

Definition 3.2 (Constraint Derivation)**.** *Let Σ be a set of clauses and C a clause. A resolution derivation (refutation) $\Sigma \vDash_r C$ of a clause $C(\bot)$ from Σ is a* constraint derivation *(*refutation*) iff all its resolution steps involve resolution with a constraint clause. A constraint derivation of a clause C from Σ is written as $\Sigma \vDash_c C$. Similarly, there is a* constraint deduction *of C from Σ, written as $\Sigma \vDash_c^d C$, if C is a tautology or if there is a clause D such that $\Sigma \vDash_c D$ and D subsumes C.*

The subsumption theorem can be formulated using constraint deductions and consequences with no positive literals.

Theorem 3.1 (Constraint Subsumption for Constraint Clause Consequences)**.** *Let Σ be a set of clauses and C a constraint clause. Then $\Sigma \models C$ iff $\Sigma \vDash_c^d C$.*

Proof. Based on the subsumption theorem for resolution derivations (Theorem 2.4), the fact that $\Sigma \models C$ implies that we have a deduction of C, i.e., there is a derivation $\Sigma \vDash_r D$ of a clause D that subsumes C. Note that D needs to be a CC in order to subsume another CC. This proof is based on being able to transform every resolution derivation of a CC D into a constraint derivation of D.

The resolution derivation $\Sigma \vDash_r D$ for sure involves resolution steps with respect to some constraint clauses. Let C_{i_1} be the closest resolvent to the root (D) of the corresponding derivation tree that is obtained by applying binary resolution without using a constraint clause: $C_{i_1} = \left(C_{i_0} \cup_r C_{j_0}\right)$. We can assume that on the path to the root of the tree the resolution steps involve always constraint clauses $C_{j_1}, \dots, C_{j_k}$:

$$\left(C_{i_0} \cup_r C_{j_0}\right) \cup_r C_{j_1} \cup_r \dots \cup_r C_{j_k} = D.$$

Therefore, we can apply the distributivity property if C_{j_1} clashes on literals coming from both clauses C_{i_0} and C_{j_0}, obtaining

$$\left((C_{i_0} \cup_r C_{j_1}) \cup_r (C_{j_0} \cup_r C_{j_1})\right) \dots \cup_r C_{j_k} = D.$$

On the other hand, if C_{j_1} clashes on literals coming from only one of the clauses (we assume it is C_{i_0} without loss of generality) we apply the commutativity property to obtain

$$\left((C_{i_0} \cup_r C_{j_1}) \cup_r C_{j_0}\right) \dots \cup_r C_{j_k} = D.$$

If we continue the same process for all the clauses $C_{j_1}, \dots, C_{j_k}$ in the same order, we obtain

$$\begin{gathered}\left(C_{i_0} \cup_r C_{j'_1} \cup_r \dots \cup_r C_{j'_{k'}}\right) \cup_r \\ \left(C_{j_0} \cup_r C_{j''_1} \cup_r \dots \cup_r C_{j''_{k''}}\right) = D.\end{gathered}$$

Because D is a constraint clause at least one of the clauses used to obtain it is also a constraint clause:

$$\left|\left(C_{i_0} \cup_r C_{j'_1} \cup_r \dots \cup_r C_{j'_{k'}}\right)\right|^+ = 0$$
or
$$\left|\left(C_{j_0} \cup_r C_{j''_1} \cup_r \dots \cup_r C_{j''_{k''}}\right)\right|^+ = 0.$$

This eliminates the resolution step between the two non-constraint clauses. In the same way, we can eliminate the rest of the resolution steps that involve two non-constraint clauses. Thus, transforming the existing deduction of C into a constraint deduction of C. □

Theorem 3.2 (Completeness of Constraint Resolution Derivations)**.** *A set of clauses Σ is unsatisfiable iff there exists a constraint refutation of Σ, i.e., $\Sigma \vDash_c \bot$.*

Proof. This follows from Theorem 3.1 by taking as consequence the empty clause that has no positive literals and is only subsumed by itself. □

3.2. Rewriting Operations and Resolution

Conjunctive query rewriting is a process that mimics the constraint derivations introduced in the previous section. However, resolution steps involving Skolem functions are performed together in order to avoid introducing literals with Skolem functions that will not be able to be removed. For existential rules, the process of query rewriting is well known [20]. However, in most of the existing literature disjunctive rules are mainly used in a forward chaining manner [13, 15] or to perform Disjunctive Datalog rewritings [3, 9, 11, 16].

In Example 2.3, one could infer that two first-degree relatives that have the same age have to be siblings:

$$\begin{aligned}&\textit{first-deg-relative}(X,Y),\\ &\qquad \textit{same-age}(X,Y) \rightarrow \textit{sibling}(X,Y).\end{aligned} \tag{13}$$

This rule can be obtained by a constraint derivation using (9) and the clauses corresponding to:

$$\begin{gathered}(\textit{same-age}(X,Y), \textit{parent}(X,Y) \rightarrow \bot)\\ (\textit{same-age}(X,Y) \rightarrow \textit{same-age}(Y,X))\end{gathered}$$

expressing respectively that children and their parents cannot have the same age and that the *same-age*/2 predicate is symmetric.

The new existential rule (13) can then be used in rewriting steps defined for existential rules.

A rewriting step as defined in the existential rules framework [20] corresponds to the resolution steps between RCs corresponding to an existential rule and a CC corresponding to the negation of a CQ.

Definition 3.3 (Rewriting Step)**.** *Let $r = B \rightarrow H$ be an existential rule, and Q a conjunctive query. If there is a subset $H' \subseteq H$ that unifies with some $Q' \subseteq Q$ through a mgu θ (i.e., $H'\theta = Q'\theta$) such that*

1. *if $v \in vars(Q \setminus Q')$ and $v \neq v\theta$, then $v\theta$ is a frontier variable of r or a constant, and*
2. *if v is an existential variable of the rule r, then $v\theta \notin vars(Q \setminus Q')$,*

then the query $(B \cup (Q \setminus Q'))\theta$ is a rewriting *of Q using the existential rule r.*

If an existential rule r has more than one atoms in its head, it gives rise to more than one RCs. Nevertheless, the resolution steps with such RCs are always performed together in order to avoid unnecessary propagation of Skolemized existential variables. Thus, the resulting clause cannot contain a Skolem term representing an existential variable of r. Hence, existential variables cannot be assigned to a variable that will be part of the result (condition 1 in Definition 3.3) nor should be replaced by a variable that belongs to the result (condition 2 in Definition 3.3).

Definition 3.3 is an adaptation to our framework of the *piece-based* rewriting step proposed in [20].

For the resolution steps between some DRCs obtained from a disjunctive existential rule and a CC obtained from the negation of a CQ, we define a corresponding rewriting step generalizing Definition 3.3 with the goal to support both existential rules and disjunctive existential rules.

Definition 3.4 (General (Disjunctive) Rewriting Step). *Let $r = B \rightarrow H$ be a rule, and Q a conjunctive query. If there is a subset $H' \subseteq H$, and for each $h_i \in H'$ there is a subset $h'_i \subseteq h_i$ that unifies with some $Q' \subseteq Q$ through a mgu θ (i.e., $h'_1\theta = \dots h'_n\theta = Q'\theta$) such that*

1. *if $v \in vars(Q \setminus Q')$, then $v\theta$ is a frontier variable of r or a constant, and*
2. *if v is an existential variable of the rule r, then $v\theta \notin vars(Q \setminus Q')$,*

then $(B \cup (Q \setminus Q') \rightarrow H \setminus H')\theta$ is a rewriting *of Q using the rule r. A rewriting step is a* disjunctive rewriting step *if the rule used is a disjunctive existential rule.*

A disjunctive rewriting step can yield a disjunctive rule with fewer disjunctive components, an existential rule in case $|(H \setminus H')\theta| = 1$ or a negative constraint (the negation of a conjunctive query) in case $H = H'$.

Example 3.2. *Consider a disjunctive existential rule:*

$$r_1 = diabetesRisk(X) \rightarrow [(diabetic(Y), sibling(Y, X)), (diabetic(Z), parent(Z, X))].$$

If we want to rewrite the query $Q = diabetic(X_1)$, to learn if there are any diabetic people, we can obtain the UCQ*-rewriting $[diabetic(X_1), diabetesRisk(X)]$, using r_1 with the unifier $\theta = \{Y \leftarrow X_1, Z \leftarrow X_1\}$.*

On the other hand, if we have the negative constraint $singleChild(X_1), sibling(Y_1, X_1)$ and the query $Q' = diabetic(Y_2), parent(Y_2, X_2)$ asking if there is a diabetic parent, we can derive the existential rule

$$diabetesRisk(X), singleChild(X) \rightarrow diabetic(Z), parent(Z, X),$$

by rewriting the constraint using the rule r_1 and the unifier $\theta_2 = \{X_1 \leftarrow X, Y_1 \leftarrow Y\}$. Using the new existential rule we obtain the following UCQ*-rewriting:*

$$[(singleChild(X), sibling(Y, X)), (diabetic(Y), parent(Y, X)), (diabetesRisk(X), singleChild(X))].$$

Note that the final UCQ*-rewriting contains also negated constraints which are possible reasons for which a query can be entailed, i.e., inconsistent knowledge bases. However, sometimes we might want to filter out the negated constraints if we are sure that the knowledge base is consistent.*

Using the above-defined rewriting steps, we can now define rewriting for DKBs.

Definition 3.5 (Rewriting). *Let $\langle \mathcal{R}, \mathcal{Q} \rangle$ be a tuple consisting of a set $\mathcal{R}$ of rules and a* UCQ *$\mathcal{Q}$. A* one-step rewriting *$\langle \mathcal{R}', \mathcal{Q}' \rangle$ of $\langle \mathcal{R}, \mathcal{Q} \rangle$ can be obtained by adding to $\mathcal{R}$ or to $\mathcal{Q}$, as appropriate, the result f' of a general rewriting step that uses one of the conjunctive queries in $\mathcal{Q}$ and a rule in $\mathcal{R}$, i.e., $\mathcal{Q}' = \mathcal{Q} \cup (\neg f')$ if f' is a negative constraint, otherwise $\mathcal{R}' = \mathcal{R} \cup (f')$.*

A k-step rewriting *of $\langle \mathcal{R}, \mathcal{Q} \rangle$ is obtained by applying a one-step rewriting to a $(k-1)$-step rewriting of $\langle \mathcal{R}, \mathcal{Q} \rangle$. For any k, a k-step rewriting of $\langle \mathcal{R}, \mathcal{Q} \rangle$ is a* rewriting *of $\langle \mathcal{R}, \mathcal{Q} \rangle$.*

So far we have dealt with rewritings of conjunctive queries with respect to existential rules and disjunctive existential rules. However, we have not considered negative constraints and conjunctive queries with negated atoms. Negative constraints are transformed into queries in the rewriting process, i.e.,

$$\mathcal{R}^{\exists}, \mathcal{R}^{\vee}, \mathcal{R}^{\perp}, \mathcal{D} \vDash \mathcal{Q} \text{ iff } \mathcal{R}, \mathcal{R}^{\vee}, \mathcal{D} \vDash \neg\mathcal{R}^{\perp}, \mathcal{Q}.$$

In a similar way, if Q is a UCQ$^{\neg}$, the entailment problem can be reduced to the entailment of a UCQ:

$$\begin{aligned}&\mathcal{R}^{\exists}, \mathcal{R}^{\vee}, \mathcal{R}^{\perp}, \mathcal{D} \models Q \text{ iff}\\ &(\mathcal{R}^{\exists}, \neg Q^{\neg 1}), (\mathcal{R}^{\vee}, \neg Q^{\neg \#}), \mathcal{D} \models \neg\mathcal{R}^{\perp}, Q^{\neg 0},\end{aligned} \tag{14}$$

where $\neg Q^{\neg 1}$ contains existential rules (negations of CQ$^{\neg}$s with one negated atom) and $\neg Q^{\neg \#}$ disjunctive existential rules (negations of CQ$^{\neg}$s with more than one negated atom).

Theorem 3.3 (Soundness and Completeness of Rewritings)**.** *Let $\langle \mathcal{R}, \mathcal{D}\rangle$ be a* DKB *and Q a* UCQ*. Then $\mathcal{R}, \mathcal{D} \models Q$ iff there is a rewriting $\langle \mathcal{R}', Q'\rangle$ of $\big\langle(\mathcal{R}, \mathcal{R}^{\vee}), (Q, \neg\mathcal{R}^{\perp})\big\rangle$ such that $\mathcal{D} \models Q_i$ for some conjunctive query Q_i in Q'.*

Proof. The k-step rewriting of $\big\langle(\mathcal{R}, \mathcal{R}^{\vee}), (Q, \neg\mathcal{R}^{\perp})\big\rangle$ is based on a constraint derivation. Moreover, such a rewriting can be mapped to a constraint derivation. Since constraint derivations are sound and complete (Theorem 3.2), this theorem also holds. □

Algorithm 1 Function to rewrite UCQs with respect to existential rules and disjunctive existential rules.

```
function rewrite_k($\mathcal{R}$, $Q$)
  $Q$ := $Q \cup \neg\mathcal{R}^{\perp}$
  $\mathcal{R}$ := $\mathcal{R} \setminus \mathcal{R}^{\perp}$
  do
    $\mathcal{R}_{old}$ := $\mathcal{R}$
    $Q_{old}$ := $Q$
    $Q$ := rewrite$^{\exists}_k$($\mathcal{R}^{\exists}$, $Q$)
    $\mathcal{R}$ := rewrite$^{\vee}$($\mathcal{R}$, $Q$)
  while ($Q \neq Q_{old}$ or $\mathcal{R} \neq \mathcal{R}_{old}$)
  return $Q$
end function
```

Given a set of rules $\mathcal{R}$ and a UCQ Q, function rewrite$_k$/2 presented in Algorithm 1 computes all the rewritings of $\big\langle(\mathcal{R}, \mathcal{R}^{\vee}), (Q, \neg\mathcal{R}^{\perp})\big\rangle$. The algorithm alternates between computing the rewritings of CQs using existential rules (rewrite$^{\exists}_k$/2 presented in Algorithm 2) and computing the rewritings using disjunctive existential rules (rewrite$^{\vee}$/2 presented in Algorithm 3). New CQs are used to generate more rules, and new existential rules are used to generate more CQs until a fixed point is reached, i.e., until no new rule or query is produced.

Algorithm 2 Function to rewrite UCQs using existential rules.

```
function rewrite$^{\exists}_k$($\mathcal{R}$, $Q$)
  $Q_{old}$ := $Q$
  $Q_{exp}$ := $Q$
  level := 0
  do
    $Q$ :=cover($Q \cup$ rew($Q_{exp}$, $\mathcal{R}$))
    $Q_{exp}$ := $Q \setminus Q_{old}$
    $Q_{old}$ := $Q$
    level := level + 1
  while $Q_{exp} \neq \emptyset$ and level < k
  return $Q$
end function
```

Algorithm 3 Function to rewrite UCQs using disjunctive existential rules.

```
function rewrite$^{\vee}$($\mathcal{R}$, $Q$)
  $\mathcal{R}_{old}$ := $\mathcal{R}$
  $\mathcal{R}_{exp}$ := $\mathcal{R}^{\vee}$
  do
    $\mathcal{R}$ := $\mathcal{R} \cup$ rew$^{\vee}$($Q$, $\mathcal{R}_{exp}$)
    $\mathcal{R}_{exp}$ := $\mathcal{R}^{\vee} \setminus \mathcal{R}_{old}$
    $\mathcal{R}_{old}$ := $\mathcal{R}$
  while $\mathcal{R}_{exp} \neq \emptyset$
  return $\mathcal{R}$
end function
```

Function rew/2 (in Algorithm 2) computes the set of the results of all possible rewriting steps for all the combinations of existential rules and CQs in its arguments. This step is known as the *expansion* of a query, and it generates more conjunctive queries. On the other hand, function rew$^{\vee}$/2 (in Algorithm 3) computes the expansion of disjunctive existential rules by computing the set of the results of all disjunctive rewriting steps for all the combinations of disjunctive existential rules and CQs in its arguments that do not yield a conjunctive query. This restriction does not affect completeness because a CQ Q' that can be generated from a CQ Q in a disjunctive rewriting step using rule r can also be generated in two steps. In particular, a disjunctive rewriting step generates first an existential rule r'

($r' \in \texttt{rew}^{\vee}(\{Q\}, \{r\})$), and then a rewriting step using r' generates the query Q' ($Q' \in \texttt{rew}(\{Q\}, \{r'\})$).

The cover/1 function (in Algorithm 2) computes, for a given UCQ Q, the minimal subset $Q' \subseteq Q$ such that for all $q \in Q$ there is a $q' \in Q'$ such that q' subsumes q, i.e., the corresponding clause to q' subsumes the corresponding clause to q.

The cover/1 function allows us to keep always the minimal set of CQs that can yield the same results. In [20], the authors perform a deeper analysis showing that using the cover computation on the rewriting algorithm they propose ensures that the resulting UCQ-rewriting will be of minimal size (cardinality).

In both $\texttt{rewrite}^{\exists}_{k}/2$ and $\texttt{rewrite}^{\vee}/2$, all newly generated CQs and rules are also expanded, unless some CQs are removed when computing the cover. The process stops when a fixed point is reached.

All CQs generated by Algorithm 1 are computed according to our definition of a rewriting; this ensures the correctness, i.e., every CQ that is generated is a CQ-rewriting of the input query with respect to the input sets of rules and constraints.

The rewriting function for disjunctive existential rules ($\texttt{rewrite}^{\vee}/2$) generates all the possible rules using an input UCQ rewriting. The fact that new rules have less disjunctive components in the head ensures that the output is always finite. Therefore, the completeness of the result of Algorithm 1 relies totally on the completeness of the result provided by Algorithm 2.

Algorithm 2 describes the rewriting function for existential rules ($\texttt{rewrite}^{\exists}_{k}/2$), and is based on the general rewriting algorithm proposed on [20]. It implements a breath-first expansion process where each iteration of the loop expands a new level of conjunctive queries. We have introduced the parameter k that allows us to control how many levels of CQs will be expanded and ensures termination of each individual call to Algorithm 2 for $k \neq \infty$. However, the loop in Algorithm 1 will keep on calling Algorithm 2 as long as new CQs are generated, without affecting the completeness of the whole rewriting process. The parameter k defines a *pause* on the existential rules rewriting process in order to generate more rules from the disjunctive existential rules.

König et al. study the completeness of the UCQ-rewriting computed by the rewriting algorithm for existential rules based on different definitions of the query expansion function [20]. Our expansion function (rew/2) ensures that the computed UCQ-rewriting is complete because it corresponds to their piece-based rewriting operator that ensures the completeness. Moreover, if there is a finite and complete UCQ-rewriting of the input UCQ, the function $\texttt{rewrite}^{\exists}_{k}/2$ will find it after a finite number of calls to it.

3.3. Rewritable Queries and Disjunctive Knowledge Bases

The termination of Algorithm 1 depends on the termination of Algorithms 2 and 3. Algorithm 3 always terminates because the produced rules contain less disjunctive components in the head. On the other hand, setting $k = \infty$ or executing a possibly infinite number of calls to Algorithm 2 is denoted by $\texttt{rewrite}^{\exists}/2$ and corresponds to the classical rewriting algorithm for existential rules proposed in [20], whose termination is studied in [7]. In general, the problem of knowing if there exists a finite UCQ-rewriting for any UCQ with respect to an arbitrary set of existential rules is undecidable [7]. A set of existential rules that ensures the existence of a finite UCQ-rewriting for any UCQ is called a finite unification set (*fus*) [6]. There are some classes of existential rules that have the *fus* property:

1. *Linear* existential rules [6]: existential rules with one atom in the body.
2. *Disconnected* existential rules [8]: existential rules that do not share variables between the body and the head.
3. *Acyclic graph of rule dependencies* (*aGRD*) [4]: existential rules that do not contain cycles in the *graph of rule dependencies*.

If a set $\mathcal{R}$ of existential rules is a *fus* and the new existential rules generated by Algorithm 3 are also a *fus*, combining them could yield a new set of existential rules that is not a *fus* [7]. Therefore, we need stronger conditions to ensure that we always call Algorithm 2 with a set of existential rules that is a *fus*.

For a set of existential rules $\mathcal{R}$, a *cut* is a partition $\{\mathcal{R}_1, \mathcal{R}_2\}$ of $\mathcal{R}$, and it is a *directed cut* ($\mathcal{R}_1 \rhd \mathcal{R}_2$) if none of rules in $\mathcal{R}_1$ depends on a rule of $\mathcal{R}_2$.

Property 3.1. *Let $\mathcal{R}$ be a set of existential rules with a directed cut $\mathcal{R}_1 \rhd \mathcal{R}_2$. For any* CQ *$Q$ and any set of facts $\mathcal{D}$ we have that*

$$\mathcal{D}, \mathcal{R} \models Q \text{ if there is a CQ } Q' \text{ such that } \mathcal{D}, \mathcal{R}_1 \models Q' \text{ and } Q', \mathcal{R}_2 \models Q.$$

Proof. The proof is based on being able to organize the application of the rules of $\mathcal{R}$. The existing dependencies ensure that the rules of $\mathcal{R}_1$ are never depending on the rules of $\mathcal{R}_2$. For a detailed proof check [7]. □

Property 3.1 [7] allows us to study the decidability of entailment when we combine two sets of rules for which the entailment problem is decidable.

In Algorithm 3, even if the resulting set of existential rules $\mathcal{R}$ is a *fus*, the process of generating new rules could potentially continue forever after we obtain new CQs from Algorithm 2. Therefore, we need ways to ensure that the total number of existential rules generated by Algorithm 3 is bounded, i.e., at some point the algorithm will not produce new rules.

Existential rules with one atom in the body ensure that the CQ-rewritings will never grow in size. Indeed, a rewriting operation will replace one or more atoms for the atomic body.

A rule $B \rightarrow H$ is *linear* if it has only one atom in the body, i.e., $|B| = 1$.

Theorem 3.4. *Let $\mathcal{R}$ be a set of rules and Q a* UCQ. *If Q contains only atomic queries and $\mathcal{R}$ only linear rules, then Algorithm 1 stops for any value of k.*

Proof. Linear existential rules are a *fus* and rewriting queries with them stops even when $k = \infty$. The new rules generated by the linear disjunctive existential rules and the atomic queries will also be linear rules, and combining them with $\mathcal{R}^{\exists}$ will also produce a *fus*. Additionally, the number of single atoms that can be built using a finite number of predicates, variables and constants is bounded. Therefore, the number of rules that we can derive from the linear disjunctive existential rules is finite. Consequently, Algorithm 1 stops because at some point no new rules and no new queries can be generated. □

Theorem 3.4 is closely related to Theorem 7.13 and Lemma 7.12 proposed by Bourhis et al. in [12]. However, it is still interesting to prove it considering the algorithm we have proposed so that the technique can be extended to the study of other fragments.

Rules that do not share variables between the head and the body produce rewritings where the introduced body of the rule is not connected to the remaining part of the the query.

A rule $B \rightarrow H$ is *disconnected* if no variable from the body is present in the head of the rule, i.e., $vars(B) \cap vars(H) = \emptyset$. Disconnected rules can still share constants between the body and the head of the rule and this allows us to express knowledge about specific individuals.

Theorem 3.5. *Let $\mathcal{R}_1$ be a* fus *and $\mathcal{R}_2$ a set of disconnected existential rules. The union of both sets $\mathcal{R}_1 \cup \mathcal{R}_2$ is also a* fus.

Proof. Disconnected rules add atoms to the rewritings that do not share variables with the remaining part of the query. It follows that the connected cardinality of CQ-rewritings produced by rules $B \rightarrow H \in \mathcal{R}_2$ is bounded as follows:

$$card^*(\texttt{rew}(B \rightarrow H, Q)) \leq \max(card^*(B), card^*(Q)).$$

The rules in $\mathcal{R}_1$ may produce CQ-rewritings with a larger connected cardinality, but they only produce a finite number of CQ-rewritings because $\mathcal{R}_1$ is a *fus*. It follows that the connected cardinality of the CQ-rewritings of an initial query Q is bounded, i.e.,

$$card^*(\texttt{rewrite}^{\exists}(\mathcal{R}_1 \cup \mathcal{R}_2, Q)) \leq \max(card^*(\texttt{rewrite}^{\exists}(\mathcal{R}_1, B)), card^*(\texttt{rewrite}^{\exists}(\mathcal{R}_1, Q))).$$

Using Lemma 2.2 we conclude that the number of rewritings produced by $\mathcal{R}_1 \cup \mathcal{R}_2$ cannot be infinite. Thus, $\mathcal{R}_1 \cup \mathcal{R}_2$ is also a *fus*. □

Theorem 3.6. *Let $\mathcal{R}$ be a set of rules and Q a* UCQ. *If $\mathcal{R}^{\exists}$ is a fus, and $\mathcal{R}^{\vee}$ a set of disconnected disjunctive existential rules, then Algorithm 1 stops for any set of constraints $\mathcal{R}^{\perp}$, any* UCQ *Q and for any value of k.*

Proof. The new existential rules produced by the function $\texttt{rewrite}^{\vee}$ are disconnected rules and they can be combined with the rules in $\mathcal{R}^{\exists}$ and yield a *fus* (follows from Theorem 3.5).

Rewritings of disjunctive rules $B_i \rightarrow H_i$ will have the following form:

$$\bigcup_j B'_j \cup \bigcup_j Q'_j \rightarrow H',$$

where $H' \subseteq H_i\sigma$ is a subset of an instance $H_i\sigma$ of the original head of the rule H_i, $B'_j \subseteq B''\sigma'$ is a subset of an instance $B''\sigma'$ of a rewriting of the body B_j of a disjunctive existential rule, i.e., $B'' \in \texttt{rewrite}^{\exists}(B_j, \mathcal{R}^{\exists})$, and $Q'_j \subseteq Q''\sigma''$ is a subset of an instance $Q''\sigma''$ of a rewriting of an input CQ or a negated constraint Q_j, i.e., $Q'' \in \texttt{rewrite}^{\exists}(Q_j, \mathcal{R}^{\exists})$. The substitutions σ' and σ'' are compositions of the *mgus* applied in the rewriting steps. None of the B'_j or Q'_j share variables between them because they are introduced using an atom in the head of the disjunctive rule that does not share variables with the body.

Because $\mathcal{R}^{\exists}$ is a *fus* we have a finite number of rewritings B'_j and Q'_j. This ensures that there is only a finite number of different bodies B' for the generated existential rules. The number of different heads, H' is obviously finite too. Therefore, there is only a finite number of different existential rules that will eventually be generated by `rewrite`$^{\vee}$. Thus, Algorithm 1 stops for any value of k. □

For other types of queries and knowledge bases there is no certainty that the algorithm will stop. However, we can still try to compute the rewritings up to a certain depth. Nevertheless, we should point that our algorithm stops if there is a finite and complete UCQ-rewriting of the input query with respect to the rules and the constraints in the knowledge base.

Theorem 3.7. *Let $\mathcal{R}$ be a set of rules and Q a* UCQ*. If a* UCQ *Q has a finite and complete* UCQ*-rewriting with respect to $\mathcal{R}$, then Algorithm 1 stops for any finite value of k.*

Proof. The completeness of the definition of rewritings and the fact that we produce rewritings using all possible one-step rewritings ensures that if there is a finite UCQ rewriting Q_f, then after a finite number of steps Algorithm 1 (with a finite value of k) will produce a rewriting equivalent to Q_f. Because Q_f is complete, any further rewriting of Q_f using existential rules will not produce new conjunctive queries, i.e., the condition $Q = Q_{old}$ holds for the rest of the iterations in the loop.

The algorithm can still produce new (disjunctive) existential rules, but since no new CQs are generated, the number of new rules that can be produced is finite due to the fact that new rules are produced with strictly less disjoint components in the head. Consequently, we will reach an iteration of the loop in Algorithm 1 where $\mathcal{R} = \mathcal{R}_{old}$. Therefore, Algorithm 1 terminates because after a finite number of iterations the condition to continue iterating on the loop will no hold. □

Note that a subset of the existential rules needed to generate the finite and complete UCQ-rewriting of the initial query in Theorem 3.7 could potentially produce an infinite number of rewritings.

Example 3.3. *To illustrate this we can consider the following* DKB*:*

– *Existential rules:*

$$(r(X,W), r(W,Y) \rightarrow r(X,Y)),$$
$$(b(X) \rightarrow a(X)),$$
$$(c(X) \rightarrow a(X)).$$

– *Disjunctive existential rules:*

$$s(X) \rightarrow [b(X), c(X)].$$

If we try to rewrite the UCQ $[a(X), (s(X), r(X, f))]$ *with respect to $\mathcal{R}^{\exists}$ using $k = \infty$ it would produce an infinite set of* CQ*-rewritings of the form:*

$$s(X), r(X, W_1), r(W_1, W_2), \ldots, r(W_n, f).$$

However, for $k = 1$ (or any other finite value) the new rules produced by the disjunctive existential rule would eventually generate the conjunctive query $s(X)$. The application of the `cover/1` *function would then remove the queries that can produce the infinite set of rewritings and yield the finite and complete* UCQ*-rewriting $[a(X), s(X), b(X), c(X)]$ of the initial query.*

Therefore, the expansion process in Algorithm 2 needs to have a finite depth (i.e., $k \neq \infty$) in order to avoid infinite loops.

Theorem 3.7 ensures that Algorithm 1 stops only if there is a finite and complete UCQ-rewriting for the input query otherwise the algorithm may never stop. However, it does not require the *fus* property for the set of existential rules in the knowledge base. On the other hand, Theorems 3.4 and 3.6 ensure that Algorithm 1 will always terminate if the required conditions are met.

3.4. On Queries with Answer Variables and Linear Queries

While Theorems 3.4 and 3.6 impose rather strong restrictions on the disjunctive framework, they also suggest the existence of finite UCQ-rewritings for very expressive types of queries with negated atoms and answer variables.

A CQ Q *with answer variables* (CQa) is a CQ of the form $ans(\mathbf{X}), B$, where B is a CSF of atoms, called the *body* of the query, and $ans(\mathbf{X})$ is the *answer atom* of the query. The fresh predicate ans/n is called the *answer predicate* and $\mathbf{X}$ a tuple of variables or constants such that $var(\mathbf{X}) \subseteq var(B)$, is called the *answer tuple* of the query. A CQa Q is often written as $ans(\mathbf{X}) :- B$. Conjunctive queries without answer variables are called *Boolean* CQs and for them the answer tuple is the empty tuple $\mathbf{X} = ()$. A CQ$^{\neg}$ *with answer variables* (CQa$^{\neg}$) is defined in the same way, but variables that appear only in negated atoms are not allowed to be part of the answer tuple $\mathbf{X}$. A UCQ (UCQ$^{\neg}$) *with answer variables* (UCQa or UCQa$^{\neg}$ respectively) is a DSF of CQas (CQa$^\neg$s) with the same answer predicate. In general, a *query with answer variables* refers to a CQa, CQa$^\neg$, UCQa or to a CQa$^\neg$.

Given a knowledge base $\mathcal{K}$, a tuple $\mathbf{t}$ of constants in $\mathcal{K}$ is a certain answer of a query with answer variables Q with respect to $\mathcal{K}$ iff $\mathcal{K}, ans(\mathbf{t}) \vDash Q$. The set of certain answers of a query Q with respect to a knowledge base $\mathcal{K}$ is denoted by $cert(Q, \mathcal{K})$. Computing the set $cert(Q, \mathcal{K})$ is known as the *query answering problem.*

Example 3.4. *Consider the following three CQas, which have different sets of answer tuples:*

$$Q_1 = ans_1() \;:-\; sibling(X, Y),$$
$$Q_2 = ans_2(X) \;:-\; sibling(X, Y)$$
$$Q_3 = ans_3(X, Y) \;:-\; sibling(X, Y).$$

We expect the certain answers of Q_1 to contain the empty tuple if someone has a sibling or otherwise be the empty set, the certain answers of Q_2 to be the set of people that have siblings, and the certain answers of Q_3 to be set of pairs of people that are siblings.

Consider also a knowledge base $\mathcal{K}$ based on Example 2.3, with a different set of facts:

$$sibling(pedro, ana)$$
$$sibling(juan, Y).$$

The first fact states that pedro and ana are siblings, while the second fact that juan has a sibling.

The certain answers of the queries with respect to $\mathcal{K}$ are the following:

$$cert(Q_1, \mathcal{K}) = \{()\}$$
$$cert(Q_2, \mathcal{K}) = \{pedro, ana, juan\}$$
$$cert(Q_3, \mathcal{K}) = \{\langle pedro, ana\rangle, \langle ana, pedro\rangle\}.$$

Thus, there are some siblings in $\mathcal{K}$ (by Q_1), pedro, ana, and juan have siblings (by Q_2) and ana is a sibling of pedro and pedro a sibling of ana since sibling/2 is symmetric (by Q_3). Note that the sibling of juan has no identity so it is not included in the certain answers of Q_3. The answers can be easily verified by solving the corresponding entailment problems, e.g., pedro is a certain answer of Q_2 because

$$\mathcal{K}, ans_2(pedro) \vDash ans_2(X), sibling(X, Y).$$

For queries with answer variables we focus on the query answering problem instead of the entailment problem. In theory we could try all possible assignments of constants in $\mathcal{K}$ to variables in the answer tuple $\mathbf{X}$ and check whether the resulting query is entailed. However, computing a UCQ-rewriting for each possible assignment of constants would not be very efficient. We can compute the UCQ-rewritings Q' of a query with answer variables Q with respect to the rules in $\mathcal{R}$ and then transform the entailment problem, i.e.,

$$\mathcal{R}, \mathcal{D}, ans(\mathbf{t}) \vDash Q \;\text{ iff }\; \mathcal{D}, ans(\mathbf{t}) \vDash Q'. \tag{15}$$

The answer atoms in the elements of Q' will be affected by the *mgus* of the rewriting process but the answer variables will never be replaced by an existential variable because of condition 1 in the definition of general (disjunctive) rewriting step.

The entailment of a UCQa$^\neg$ can be transformed into the entailment of a UCQ (14). However, the presence of answer atoms in CQa$^\neg$s will create rules with answer atoms in their body that may produce rewritings with more than one occurrence of answer atoms. Because the set of facts can only contain one answer atom, a rewriting with more than one occurrence of answer atoms $ans(\mathbf{X_1}), \ldots, ans(\mathbf{X_n}), B'$ can only be entailed in case there is an *mgu* for $\{ans(\mathbf{X_1}), \ldots, ans(\mathbf{X_n})\}$. A UCQ-*rewriting* of a query with answer variables is *deterministic* if the CQs in it do not contain more than one occurrence of answer atoms. Rewritings without answer atoms correspond to rewritings of the negated constraints. They allow us to check the consistency of the data $\mathcal{D}$ with respect to the rules in our knowledge base. However, the query answering problem does not make much sense when one of these rewritings is entailed by the data.

Algorithm 1 needs to be modified in order to avoid unnecessary rewritings of queries with answer variables. In Algorithms 2 and 3 we need to modify the functions *rew*/2 and *rew*$^\vee$/2 that compute one-step rewritings so that they only give rewritings with no more than one answer atom. More specifically, the resulting CQs B (rules $B \rightarrow H$) with more than one answer atom (i.e, $\{ans(\mathbf{X_1}), \ldots, ans(\mathbf{X_n})\} \subseteq B$) are replaced by the CQ $B\theta$ (rule $B\theta \rightarrow H\theta$) where $\theta = mgu(\{ans(\mathbf{X_1}), \ldots, ans(\mathbf{X_n})\})$. We call these modified functions *deterministic one-step rewriting* functions. Algorithm 1 with deterministic one-step rewriting functions computes a deterministic UCQ-rewriting that is complete based on the fact that the answer predicate is fresh, i.e., it is not used in the knowledge base. Using deterministic one-step rewriting functions helps with the termination of the rewriting process.

Example 3.5. *Consider a knowledge base without rules and the following* UCQa$^\neg$

$$Q = [(ans(X,Z), r(X,Y), r(Y,Z), \neg r(X,Z)),\\ (ans(X,a), r(X,a))].$$

The query Q has infinitely many CQ*-rewritings of the following form:*

$$ans(X,a), ans(X,X_1), \dots, ans(X,X_{n-1}),\\ r(X,X_n), r(X_n,X_{n-1}), \dots, r(X_1,a).$$

However, there is a finite and complete deterministic UCQ*-rewriting of Q:*

$$[(ans(X,a), r(X,X_2), r(X_2,a), r(a,a)),\\ (ans(X,a), r(X,X_1), r(X_1,a)),\\ (ans(X,a), r(X,a))].$$

Answer variables play a different role when splitting CSFs on connected components. The *super cardinality* of a CSF of atoms F with answer variables $\mathbf{X}$ is the number of non-answer variables in it: $card_+(F) = |vars(F) \setminus vars(\mathbf{X})|$. We say that two non-answer variables u and v in a CSF of atoms F are *super connected* if they belong to the same atom, or if there is another non-answer variable z in F that is super connected to both u and v.

A CSF of atoms is *super connected* if all the atoms in it contain non-answer variables super connected to each other. Atoms containing only constants or answer variables in their arguments are super connected formulas with a super cardinality of zero.

A CSF F can be partitioned into a set $\{U_1, \dots, U_n\}$ of super connected components such that if $v \in vars(U_i)$ is super connected to $u \in vars(U_j)$, then $i = j$. The *super connected cardinality* of F is defined as the maximum super cardinality of the super connected components in the partition of F and denoted as $card_+^*(F) = \max_i(card_+(U_i))$. The super connected cardinality of a DSF $[F_1, \dots, F_m]$ is the maximum super connected cardinality of the formulas F_i, i.e., $card_+^*([F_1, \dots, F_m]) = \max_i (card_+^*(F_i))$.

Lemma 3.1. *Let $\mathcal{K}$ be a knowledge base and F a* CSF *of atoms with answer atom $ans(\mathbf{X})$ partitioned into the super connected components $\{U_1, \dots, U_n\}$. Then,*

$$\mathcal{K}, ans(\mathbf{t}) \vDash F \quad \textit{iff} \quad \mathcal{K}, ans(\mathbf{t}) \vDash ans(\mathbf{X}), U_i \\ \textit{for every } U_i. \qquad (16)$$

Proof. It follows directly from Lemma 2.1 after transforming (16) into:

$$\mathcal{K} \vDash F\theta \quad \text{iff} \quad \mathcal{K} \vDash U_i\theta \\ \text{for every } U_i \neq ans(\mathbf{X}), \qquad (17)$$

where $\theta = mgu(ans(\mathbf{t}), ans(\mathbf{X}))$. Note that θ replaces the answer variables by constants and the resulting connected components will be the same as the super connected components. □

Lemma 3.2. *Let k be a natural number. There are a finite number of equivalence classes of* CSF*s of atoms with super connected cardinality of at most k that can be constructed using a finite set of predicates and a finite set of constants.*

Proof. Straightforward using Lemma 3.1 and similar arguments to the ones used in the proof of Lemma 2.2. □

Given a set of rules $\mathcal{R}$ without disjunctive existential rules (i.e., $\mathcal{R}^\vee = \emptyset$), using reduction (14) we can focus on the disjunctive rules that are obtained from the negation of the CQa$^\neg$s:

$$\mathcal{R} \vDash Q \quad \text{iff} \quad (\mathcal{R}^\exists, \neg Q^{\neg 1}), \neg Q^{\neg \#} \vDash \neg \mathcal{R}^\perp, Q^{\neg 0}, \quad (18)$$

and study when Algorithm 1 with deterministic one-step rewriting functions terminates.

If all the variables in the frontier of the CQa$^\neg$s are also answer variables, then the corresponding disjunctive existential rules will act similarly to disconnected rules if we use deterministic one-step rewriting functions on Algorithm 1.

Theorem 3.8. *Let $\mathcal{R}$ be a set of rules without disjunctive existential rules and Q a* UCQa$^\neg$*. If all the variables in the frontier of the* CQa$^\neg$*s in Q are also answer variables and $\mathcal{R}^\exists$ is a fus, then Algorithm 1 with deterministic one-step rewriting functions, applied on the rules $\mathcal{R} \cup Q^{\neg 1} \cup \neg Q^{\neg \#}$, and* UCQ *$Q^{\neg 0}$, terminates for any value of k.*

Proof. The variables that appear in negated atoms of the CQ$^\neg$s will end up being the variables in the head of the corresponding rules in $(\neg Q^{\neg i})_{i>0}$. However, the variables that only appear in the negated atoms will be translated to existential variables and only answer variables are going to be frontier variables of the corresponding rules. Therefore, every new existential rule will have the frontier variables included in the set of answer variables.

In the presence of deterministic one-step rewriting functions, existential rules with all the frontier included in the set of answer variables ($\mathcal{R}_a$) add atoms to the rewritings that do not share non-answer variables with the remaining part of the query. This ensures that when they are combined with a *fus*, the super connected cardinality of the UCQ-rewritings will be bounded. More specifically for any UCQ Q',

$$\begin{aligned} card^*_+(\texttt{rewrite}^\exists\,(\mathcal{R}^\exists \cup \mathcal{R}_a, Q')) \leq \\ \max\,(\, card^*_+(\texttt{rewrite}^\exists(\mathcal{R}^\exists, B)), \\ card^*_+(\texttt{rewrite}^\exists(\mathcal{R}^\exists, Q'))), \end{aligned}$$

where B is the body of rules in $\mathcal{R}_a$.

Using Lemma 3.2 we can also affirm that there are finitely many rewritings that can be obtained. Therefore, every existential rule that is generated from the negated CQ$^\neg$s in $(Q^{\neg i})_{i>0}$ together with the rules in $\mathcal{R}^\exists$ will yield a finite and complete deterministic UCQ-rewriting. Thus, we can ensure the termination of every iteration of the loop in Algorithm 1 for any value of k.

Likewise, every existential rule that is generated from the negated CQ$^\neg$s in $(Q^{\neg i})_{i>0}$ will have a body with a bounded super connected cardinality. Thus, we cannot generate infinitely many existential rules from the CQ$^\neg$s in $(Q^{\neg i})_{i>0}$ (Lemma 3.2), which ensures the termination of Algorithm 1 with deterministic one-step rewriting functions for any value of k. □

Based on Theorem 3.4, we can also define other restrictions on UCQa$^\neg$s that ensure that the rewriting algorithm stops. As in the case of existential rules, we say that a CQ (CQ$^\neg$) is *linear* if it contains only one positive literal. Similarly, a UCQ (UCQ$^\neg$) is linear if all the CQs (UCQ$^\neg$s) in it are also linear. A CQa (CQa$^\neg$) is linear if it contains only one positive literal in the body. Likewise, a UCQa (UCQa$^\neg$) is linear if all the CQas (UCQa$^\neg$s) in it are also linear.

Theorem 3.9. *Let $\mathcal{R}$ be a set of rules without disjunctive existential rules and Q a* UCQa$^\neg$. *If $\mathcal{R}$ is a set of linear rules and Q a linear* UCQa$^\neg$, *then Algorithm 1 with deterministic one-step rewriting functions, applied on the rules $\mathcal{R} \cup Q^{\neg 1} \cup \neg Q^{\neg \#}$, and* UCQ $Q^{\neg 0}$, *stops for any value of k.*

Proof. Because Q is linear the corresponding disjunctive rules $(\neg Q^{\neg i})_{i>1}$ will have two atoms in the body and one of them will be an answer atom. The CQas will have one atom in the body and the negated constraints will only have one atom. Therefore, the deterministic one-step rewriting functions ensure that the new rules generated from disjunctive rules will contain only two atoms in the body and one of them will be an answer atom. There is a finite number of rules than can be generated with two atoms in the body and a decreasing number of disjoints in the head.

The deterministic CQ-rewritings produced by linear existential rules in $\mathcal{R}$ and the existential rules generated from $(\neg Q^{\neg i})_{i>0}$ will have a maximum of two atoms. Thus, there are finitely many deterministic CQ-rewritings that can be generated.

We conclude that Algorithm 2 terminates every time it is called in the main loop and also that the condition to continue executing the loop at some point will not hold because there are a finite number of rules and CQs rewritings that can be generated. Consequently, Algorithm 1 with deterministic one-step rewriting functions stops for any value of k. □

Finally, if there is a finite and complete deterministic UCQ-rewriting of a UCQa$^\neg$ Q with respect to a set of rules $\mathcal{R}$ that does not contain disjunctive existential rules, we can ensure that Algorithm 1 with deterministic one-step rewriting functions stops.

Theorem 3.10. *Let $\mathcal{R}$ be a set rules without disjunctive existential rules and Q a* UCQa$^\neg$. *If there is a finite and complete deterministic* UCQ*-rewriting of Q with respect to $\mathcal{R}$, then Algorithm 1 with deterministic one-step rewriting functions, applied on the rules $\mathcal{R} \cup Q^{\neg 1} \cup \neg Q^{\neg \#}$, and* UCQ $Q^{\neg 0}$, *terminates for any finite value of k.*

Proof. Deterministic one-step rewriting functions will only discard rewritings that are not deterministic. The algorithm produces rewritings using all possible deterministic one-step rewritings and this ensures that if there is a finite deterministic UCQ rewriting Q_f, then after a finite number of iterations (with a finite value of k) a rewriting equivalent to Q_f will be generated. The completeness of Q_f ensures us that any further rewriting of Q_f using existential rules and deterministic one-step rewriting functions will not produce new deterministic CQs, i.e., the condition $Q = Q_{old}$ holds for the rest of the iterations in the loop. Consequently, after finitely many iterations we also reach the condition $\mathcal{R} = \mathcal{R}_{old}$. Therefore, Algorithm 1 terminates because after a finite number of iterations the condition to continue iterating on the loop will not hold. □

4. Implementation and Experiments

COMPLETO[1] is a query rewriting system that focuses on answering UCQ$^\neg$s in the framework of existential rules. The first version of COMPLETO [22] answers CQ$^\neg$ using a resolution-based approach to eliminate negated atoms. The algorithm proposed is complete only for a very restricted type of queries.

In the second version of the system [23], queries with only one negated atom are answered by being transformed into rules. The approach is complete but termination is guaranteed only when the resulting set of rules is a *fus*.

The current version of COMPLETO implements Algorithm 1 with deterministic one-step rewriting functions and answers queries with answer variables that have an arbitrary number of negated atoms. Algorithm 1 can be seen as a generalization of both algorithms proposed in [22, 23]. Indeed, queries with one negated atom are transformed into rules, while the rewriting defined for disjunctive rules is similar to what was presented in [22] as constraint resolution. Furthermore, we take advantage of the termination results for knowledge bases consisting of a *fus* and UCQ$^\neg$s whose frontier is part of the answer variables of the query (Theorem 3.8), as well as for knowledge bases consisting only of linear elements (Theorem 3.9). Choosing $k = \infty$ allows the rewriting with respect to existential rules to be performed by an external rewriter if the are no answer variables in the queries.

4.1. Experiments

To the best of our knowledge there is no other system that produces UCQ-rewritings for UCQ$^\neg$s with universally quantified negation. Therefore, the experiments were preformed in order to get a general idea of the performance of COMPLETO producing UCQ-rewritings. We used an Intel(R) Core(TM) i5-7300HQ CPU at 2.50 GHz with 8 GB of RAM running 64-bit Windows 10.

For the experiments we used two ontologies that contain negative constraints and have been used in previous research papers based on queries with negation [22, 23]. The first is the Lehigh University Benchmark (LUBM) ontology [17], enriched with 70 additional disjoint classes axioms added for the atomic *sibling* classes, i.e., for classes asserted to share the same super-class. Secondly, we used the TRAVEL ontology[2] that has 10 disjoint class axioms. The OWL 2 ER [5] fragment of both ontologies was translated into existential rules. We were not able to prove the *fus* property for the set of existential rules obtained from neither of the two ontologies we used.

[1]http://image.ntua.gr/~gardero/completo3.0/

[2]https://protege.stanford.edu/ontologies/travel.owl

Table 3
Rewriting experiments results for the CQa$^\neg$s from LUBM and TRAVEL ontologies.

Ontology	Info	rew	time	mem
LUBM	UCQa$^\neg$	77	6193.12	2138
	min	0	104	1129
	mean	4	205.58	2069
	max	55	466	2237
TRAVEL	UCQa$^\neg$	18	264.96	2043
	min	0	1	123
	mean	2	2.12	143
	max	76	8	920

We also prepared a query file with 500 CQa$^\neg$s for each ontology which we used to let COMPLETO produce finite UCQ-rewritings of the UCQa$^\neg$ that contains all the queries in the file and also for each separated CQa$^\neg$. The queries contain 3 atoms and 2 of them are negated. The queries have one variable in the frontier which is also the answer variable of the query. We generated the queries by performing Association Rule Mining [2] on a dataset obtained from the assertions of the ontologies. The queries and the ontologies we used are publicly available [3].

Table 3 shows the size of the UCQ-rewriting (rew) for the UCQa$^\neg$ containing all the CQa$^\neg$s in the file and the minimum (min), mean and maximum (max) statistics for the rewriting of each individual CQa$^\neg$ in the file. The table also shows the time (time) in seconds and the RAM memory (mem) in Mb used by the rewriting process in each of the cases. The results give an idea of the performance of the system with respect to each UCQa$^\neg$ or individual CQa$^\neg$.

For the TRAVEL ontology, the size of the UCQ-rewriting of the UCQa$^\neg$ is smaller than the biggest UCQ-rewriting for an individual CQa$^\neg$. The time that it took to compute the rewriting of the UCQa$^\neg$ is the time that it takes on average to rewrite 125 individual CQa$^\neg$s (5 min). The RAM memory used to rewrite the UCQa$^\neg$ is approximately the double of the RAM used for rewriting the individual CQa$^\neg$ that consumed the most RAM memory.

For the LUBM ontology, the size of the rewriting of the UCQa$^\neg$ has 11 more queries than the biggest re-

[3]http://image.ntua.gr/~gardero/completo3.0/ontologies/

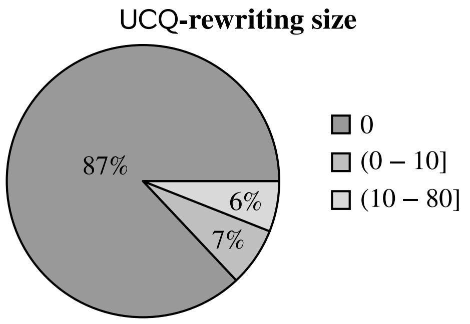

Figure 3. Size of the UCQ-rewritings for the TRAVEL ontology.

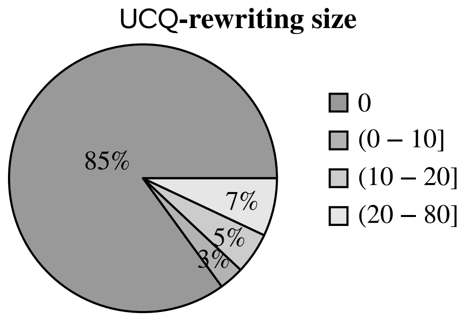

Figure 4. Size of the UCQ-rewritings for the LUBM ontology.

writing for an individual CQa¬. The time that it took to compute the rewriting of the UCQa¬ is the time that it takes on average to rewrite 30 individual CQa¬s (less than 2 hours). The RAM memory used to write the UCQa¬ is less than the RAM memory that was used for rewriting the individual CQa¬ that consumed the most RAM memory.

For both ontologies the RAM memory consumed to compute the rewritings was approximately 2 GB.

Figures 3 and 4 show information about the UCQ-rewriting size. In both ontologies at least 85 % of the queries have zero rewritings. In this cases, the CQ-rewritings of the CQas are subsumed by the CQ-rewritings of the negative constraints of the DKB.

Figures 5 and 6 show the cumulative distribution of the rewriting runtime. Dashed horizontal lines represent the mean runtime. Each bar represents the number of queries that were rewritten in or faster than the corresponding time. Note that in both cases the runtime for more than 60 % of the queries was smaller than the mean runtime.

Figures 7 and 8 show the correlation matrix with different performance parameters for the TRAVEL and LUBM ontologies. In order to get an idea of the rewriting process we computed the RAM memory used by

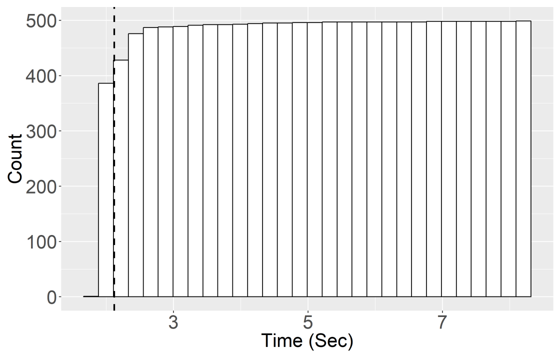

Figure 5. Cumulative distribution of the time needed to compute the UCQ-rewriting for the TRAVEL ontology.

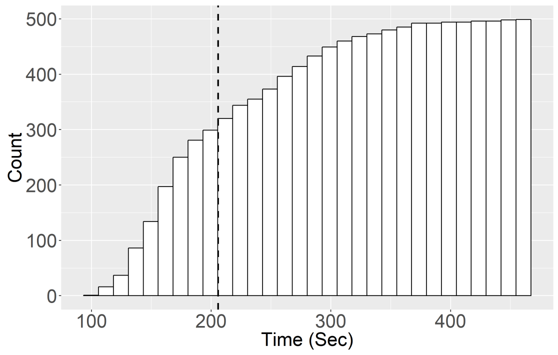

Figure 6. Cumulative distribution of the time needed to compute the UCQ-rewriting for the LUBM ontology.

the system (mem), the time that it takes to compute the UCQ-rewriting (time), the size of the UCQ-rewriting (rew), the number of generated existential rules in the rewriting process (ger), the number of rewritten (expanded) disjunctive existential rules (ecr) and also the number of generated (gcq) and rewritten (ecq) conjunctive queries. For the LUBM ontology we can notice that the number of generated CQs and the size of the UCQ-rewriting have a correlation coefficient of 0.9. For the TRAVEL ontology we can see that time, ecq, gcq and mem are all correlated with coefficients greater than or equal to 0.94.

Example 4.1. *One of the queries for the* TRAVEL *ontology was:*

$$ans(X) :- \neg Capital(X), \neg Town(X), \\ Destination(X).$$

It focuses on destinations that cannot be capitals or towns. The UCQ*-rewriting produced by* COMPLETO

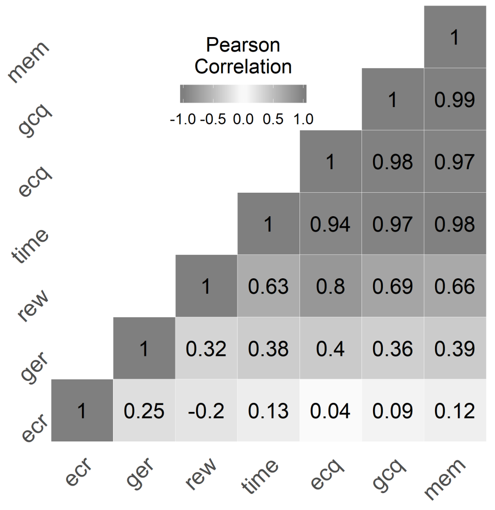

Figure 7. Correlation matrix with different performance parameters for the TRAVEL ontology.

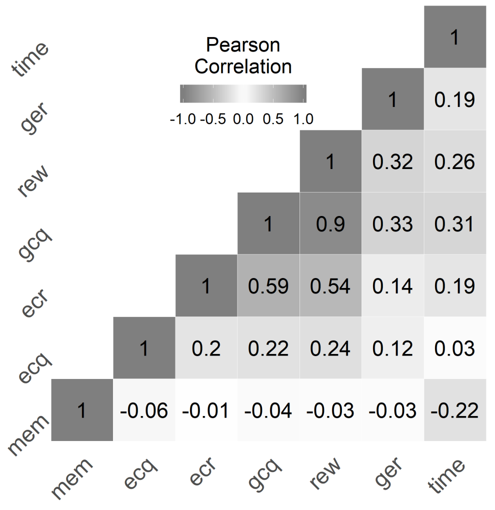

Figure 8. Correlation matrix with different performance parameters for the LUBM ontology.

was the following:

$$
\begin{aligned}
[\ & ans(X)\ :-\ Farmland(X), \\
& ans(X)\ :-\ NationalPark(X), \\
& ans(X)\ :-\ RuralArea(X)\ ].
\end{aligned}
$$

Considering the above interpretation of the query, the answer tells us that only farmlands, national parks and rural areas cannot be town or capital destinations.

5. Conclusions

In this paper, we studied the application of the query rewriting approach on the framework of disjunctive existential rules in order to produce complete UCQ-rewritings that encode the answers of an initial query.

To ensure the completeness of our rewriting approach, we introduced a special case of first-order logic resolution (constraint resolution), where every resolution step involves one clause without positive literals, and the subsumption theorem holds when the consequence is a clause without positive literals. The resolution completeness theorem also holds, allowing constraint resolution to be used in refutation procedures for FOL formulas.

Based on the definition of constraint resolution we proposed an extension of the rewriting approach for existential rules in order to deal with disjunctive existential rules. The rewriting of a disjunctive existential rule produces disjunctive rules with less disjunctions in the head, and eventually produces an existential rule or a conjunctive query. The rules generated from disjunctive rules are then used in order to find additional rewritings of the initial conjunctive query rewriting. The proposed algorithm can be used for general knowledge bases with disjunctive existential rules; it terminates for the cases where there is a finite and complete UCQ-rewriting of the input queries with respect to the (disjunctive) existential rules and the negative constraints. However, there are rather strong conditions that are able to ensure the existence of a finite and complete UCQ-rewriting.

Moreover, we studied some of the sufficient conditions that ensure that the proposed algorithm terminates. One case requires atomic CQs and all elements of the knowledge base to be linear. The other case, requires a *fus* and disconnected disjunctive existential rules without imposing restrictions on the input CQs or the constraints. Both cases impose very strong conditions on the disjunctive existential rules. However, for knowledge bases without disjunctive existential rules, we were able to provide finite and complete deterministic UCQ-rewritings for UCQa¬s with at most one positive atom (with respect to linear existential rules and linear constraints) and for UCQa¬s that include the frontier in the answer variables (with respect to a *fus*). Both types of UCQa¬s are very expressive.

Using the proposed algorithm and taking advantage of the stopping criteria, we implemented a sound and complete rewriting approach for unions of conjunctive queries with negated atoms and answer variables in the

COMPLETO system that specialises in query answering for conjunctive queries with negation. The implementation was evaluated on two ontologies.

The experimental results showed that the implementation is able to provide UCQ-rewritings in reasonable time and using a reasonable amount of RAM memory. Also, rewriting $\mathsf{UCQ}^{\neg}$s with a large number of queries takes considerably less time than the time required to rewrite all the $\mathsf{CQ}^{\neg}$s individually.

In the future, we would like to focus on implementing the proposed algorithm in a more efficient way and also on studying other sufficient conditions that ensure the termination of our rewriting algorithm.

Acknowledgments

We would like to thank Lida Petrou for providing the queries for the experiments. Also, we thank Stathis Delivorias and Michael Giazitzoglou for their support and comments in the writing process of this paper.

References

[1] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)

[2] Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993. pp. 207–216. ACM Press (1993). , https://doi.org/10.1145/170035.170072

[3] Ahmetaj, S., Ortiz, M., Simkus, M.: Rewriting guarded existential rules into small datalog programs. In: Kimelfeld, B., Amsterdamer, Y. (eds.) 21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria. LIPIcs, vol. 98, pp. 4:1–4:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). , https://doi.org/10.4230/LIPIcs.ICDT.2018.4

[4] Baget, J.: Improving the forward chaining algorithm for conceptual graphs rules. In: Dubois, D., Welty, C.A., Williams, M. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004. pp. 407–414. AAAI Press (2004), http://www.aaai.org/Library/KR/2004/kr04-043.php

[5] Baget, J., Gutierrez, A., Leclère, M., Mugnier, M., Rocher, S., Sipieter, C.: Datalog+, ruleml and OWL 2: Formats and translations for existential rules. In: Bassiliades, N., Fodor, P., Giurca, A., Gottlob, G., Kliegr, T., Nalepa, G.J., Palmirani, M., Paschke, A., Proctor, M., Roman, D., Sadri, F., Stojanovic, N. (eds.) Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany, August 2-5, 2015. CEUR Workshop Proceedings, vol. 1417. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1417/paper9.pdf

[6] Baget, J., Leclère, M., Mugnier, M., Salvat, E.: Extending decidable cases for rules with existential variables. In: Boutilier, C. (ed.) IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 677–682 (2009), http://ijcai.org/Proceedings/09/Papers/118.pdf

[7] Baget, J., Leclère, M., Mugnier, M., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. **175**(9-10), 1620–1654 (2011). , https://doi.org/10.1016/j.artint.2011.03.002

[8] Baget, J., Mugnier, M.: Extensions of simple conceptual graphs: the complexity of rules and constraints. J. Artif. Intell. Res. **16**, 425–465 (2002). , https://doi.org/10.1613/jair.918

[9] Bárány, V., Benedikt, M., ten Cate, B.: Rewriting guarded negation queries. In: Chatterjee, K., Sgall, J. (eds.) Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8087, pp. 98–110. Springer (2013). , https://doi.org/10.1007/978-3-642-40313-2_11

[10] Bárány, V., ten Cate, B., Otto, M.: Queries with guarded negation. Proc. VLDB Endow. **5**(11), 1328–1339 (2012). , http://vldb.org/pvldb/vol5/p1328_vincebarany_vldb2012.pdf

[11] Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. ACM Trans. Database Syst. **39**(4), 33:1–33:44 (2014). , https://doi.org/10.1145/2661643

[12] Bourhis, P., Manna, M., Morak, M., Pieris, A.: Guarded-based disjunctive tuple-generating dependencies. ACM Trans. Database Syst. **41**(4), 27:1–27:45 (2016). , https://doi.org/10.1145/2976736

[13] Carral, D., Dragoste, I., Krötzsch, M.: Tractable query answering for expressive ontologies and existential rules. In: d'Amato, C., Fernández, M., Tamma, V.A.M., Lécué, F., Cudré-Mauroux, P., Sequeda, J.F., Lange, C., Heflin, J. (eds.) The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10587, pp. 156–172. Springer (2017). , https://doi.org/10.1007/978-3-319-68288-4_10

[14] Du, J., Pan, J.Z.: Rewriting-based instance retrieval for negated concepts in description logic ontologies. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9366, pp. 339–355. Springer (2015). , https://doi.org/10.1007/978-3-319-25007-6_20

[15] Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: Rovan, B., Sassone, V., Widmayer, P. (eds.) Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7464, pp. 1–18. Springer (2012). , https://doi.org/10.1007/978-3-642-32589-2_1

[16] Gottlob, G., Rudolph, S., Simkus, M.: Expressiveness of guarded existential rule languages. In: Hull, R., Grohe, M. (eds.) Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014. pp. 27–38. ACM (2014). , https://doi.org/10.1145/2594538.2594556

[17] Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semant. **3**(2-3), 158–182 (2005). , https://doi.org/10.1016/j.websem.2005.06.005

[18] Gutiérrez-Basulto, V., Ibáñez-García, Y.A., Kontchakov, R., Kostylev, E.V.: Conjunctive queries with negation over dl-lite: A closer look. In: Faber, W., Lembo, D. (eds.) Web Reasoning and Rule Systems - 7th International Conference, RR 2013, Mannheim, Germany, July 27-29, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7994, pp. 109–122. Springer (2013). , https://doi.org/10.1007/978-3-642-39666-3_9

[19] Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Long rewritings, short rewritings. In: Kazakov, Y., Lembo, D., Wolter, F. (eds.) Proceedings of the 2012 International Workshop on Description Logics, DL-2012, Rome, Italy, June 7-10, 2012. CEUR Workshop Proceedings, vol. 846. CEUR-WS.org (2012), http://ceur-ws.org/Vol-846/paper_41.pdf

[20] König, M., Leclère, M., Mugnier, M., Thomazo, M.: Sound, complete and minimal ucq-rewriting for existential rules. Semantic Web **6**(5), 451–475 (2015). , https://doi.org/10.3233/SW-140153

[21] Libkin, L.: Incomplete information and certain answers in general data models. In: Lenzerini, M., Schwentick, T. (eds.) Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece. pp. 59–70. ACM (2011). , https://doi.org/10.1145/1989284.1989294

[22] Matos Alfonso, E., Stamou, G.: Rewriting queries with negated atoms. In: Costantini, S., Franconi, E., Woensel, W.V., Kontchakov, R., Sadri, F., Roman, D. (eds.) Rules and Reasoning - International Joint Conference, RuleML+RR 2017, London, UK, July 12-15, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10364, pp. 151–167. Springer (2017). , https://doi.org/10.1007/978-3-319-61252-2_11

[23] Matos Alfonso, E., Stamou, G.: On horn conjunctive queries. In: Benzmüller, C., Ricca, F., Parent, X., Roman, D. (eds.) Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11092, pp. 115–130. Springer (2018). , https://doi.org/10.1007/978-3-319-99906-7_8

[24] NCI dictionary of cancer terms: first-degree relative. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/first-degree-relative

[25] Nienhuys-Cheng, S., de Wolf, R. (eds.): Foundations of Inductive Logic Programming, Lecture Notes in Computer Science, vol. 1228. Springer (1997). , https://doi.org/10.1007/3-540-62927-0

[26] Onet, A.: The chase procedure and its applications in data exchange. In: Kolaitis, P.G., Lenzerini, M., Schweikardt, N. (eds.) Data Exchange, Integration, and Streams, Dagstuhl Follow-Ups, vol. 5, pp. 1–37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013). , https://doi.org/10.4230/DFU.Vol5.10452.1

[27] Rosati, R.: On the decidability and finite controllability of query processing in databases with incomplete information. In: Vansummeren, S. (ed.) Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA. pp. 356–365. ACM (2006). , https://doi.org/10.1145/1142351.1142404

[28] Rosati, R.: The limits of querying ontologies. In: Schwentick, T., Suciu, D. (eds.) Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4353, pp. 164–178. Springer (2007). , https://doi.org/10.1007/11965893_12

[29] Tessaris, S.: Questions and answers: reasoning and querying in Description Logic. Ph.D. thesis, University of Manchester (2001)