

Towards Farsighted Dependencies for Existential Rules

Jean-François Baget^{1,3}, Marie-Laure Mugnier^{2,3}, and Michaël Thomazo^{2,3}

¹ INRIA, France

² University Montpellier 2, France

³ LIRMM, France

Abstract. We consider existential rules (also called Tuple-Generating Dependencies or Datalog+/- rules). These rules are particularly well-suited to the timely ontological query answering problem, which consists of querying data while taking terminological knowledge into account. Since this problem is not decidable in general, various conditions ensuring decidability have been proposed in the literature. In this paper, we focus on conditions that restrict the way rules may interact to ensure that the forward chaining mechanism is finite. After a review of existing proposals, we propose a generalization of the notion of rule dependency, namely k -dependency, that allows to enlarge halting cases. It can also be used to compile the rule base, which leads to improve query answering algorithms.

1 Introduction

First-order Horn rules (without function symbols except constants) have long been used in artificial intelligence, as well as in databases under name Datalog. In this paper, we consider an extension of these rules that have the ability of generating new unknown individuals (an ability called value invention in databases [AHV95]). More precisely, these extended rules are of the form $Body \rightarrow Head$, where $Body$ and $Head$ are conjunctions of atoms, and variables occurring only in the $Head$ are *existentially* quantified. E.g., $\forall x (Human(x) \rightarrow \exists y (Parent(y, x) \wedge Human(y)))$. Hence their name $\forall\exists$ -rules in [BLMS09, BLM10] or existential rules [BMRT11, KR11]. Existential rules are known in databases as Tuple-Generating Dependencies (TGDs) [BV84]. TGDs have been extensively used as a high-level generalization of different kinds of constraints, e.g. for data exchange [FKMP05]. They also correspond to rules in conceptual graphs, a graph-based knowledge representation formalism [Sow84, CM09].

Recently, there has been renewed interest for these rules in the context of ontological query answering, a timely problem both in knowledge representation and in databases. This problem is also known as ontology-based data access. Ontological query answering consists in querying data while taking the semantics of domain knowledge into account. More precisely, let us consider a knowledge base (KB) composed of a terminological part (expressing domain knowledge) and an assertional part (called here the facts). Queries are supposed to be at least as expressive as conjunctive queries in databases, which can be seen as existentially closed conjunctions of atoms. The query problem consists in computing the set of answers to a query on a given KB. A fundamental decision problem is thus (Boolean) conjunctive query answering, which can be expressed as an entailment problem: is a (Boolean) conjunctive query entailed by a KB? In this paper, we will focus on this latter problem, denoted by ENTAILMENT.

In knowledge representation, and in the Semantic Web, terminological knowledge is often represented with description logics (DL). However, DLs traditionally focused on reasoning about the terminology itself (for instance, checking its consistency) and querying tasks were restricted to ground atom entailment. Conjunctive query answering with DLs that were considered as basic ten years ago appears to be extremely complex (e.g., for the classical DL \mathcal{ALCT} , it is 2ExpTime-complete, and still NP-complete in the size of the data). Thus, families of lightweight DLs dedicated to conjunctive query answering on large amounts of data have been designed recently, namely DL-Lite [CGL⁺07], \mathcal{EL} [BBL05,LTW09], and their generalization to Horn-logics (see e.g. [KRH07]). These DLs are the basis of the tractable profiles of OWL 2. Interestingly, existential rules generalize the core of these new DLs [CGL09,BLM10,BMRT11].

Alternatively, querying large amounts of data is the fundamental task of databases. Therefore, the challenge in this domain is now to access data while taking terminological knowledge into account. Deductive databases, and first of all the Datalog language, allow for integrating some terminological knowledge. However, in Datalog rules, variables are range-restricted, i.e., all variables in the rule head necessarily occur in the rule body, which does not allow for value invention. This feature has been recognized as mandatory in an open-world perspective, where it cannot be assumed that all individuals are known in advance. This motivated the recent extension of Datalog to TGDs (i.e., existential rules) which gave rise to the Datalog +/- family [CGK08,CGL09,CGL⁺10].

Existential rules thus provide some particularly interesting features in the context of the Web. On the one hand, they cover the core of lightweight DLs dedicated to query answering, while being more powerful and flexible. In particular, they have no restricted predicate arity (while DLs allow for unary and binary predicates only). On the other hand, they cover Datalog, while allowing for value invention.

The ability to generate existential variables, along with complex conjunctions of atoms, makes entailment with these rules undecidable [BV81,CLM81]. Since the birth of TGDs, and recently within the Datalog +/- and existential rule frameworks, various conditions of decidability have been exhibited. Stated in an abstract way, decidability can be based on the finiteness of classical mechanisms, namely forward chaining [BM02] (called the *chase* in databases [JK84]) or backward chaining [BLMS09]. In [CGK08], it is shown that entailment is still decidable when the generated facts have a tree-like structure (when seen as graphs), even if the forward chaining does not halt. Two kinds of “concrete” criteria ensuring that one of the above abstract conditions is satisfied can be found in the literature. The first kind of criteria achieves decidability by restricting the syntax of rules or rule sets (cf. [BLMS11] for a synthetic presentation of these classes of rules, and [CGP10,KR11] for very recent new classes). The second kind of criteria restricts the way rules can interact. In this paper, we will focus on this latter kind of techniques. More precisely, we consider a directed graph, called the *graph of rule dependencies* (GRD), which encodes dependencies between rules: its nodes are the rules and an arc from a rule R_i to a rule R_j means that R_j depends on R_i , i.e., there exist facts such that applying R_i on these facts leads to trigger a new application of R_j . A circuit (i.e., a directed cycle) in this graph may indicate a potentially infinite sequence of rule applications. Hence, decidability results are obtained by imposing conditions on circuits of this graph (see [BLMS11] for details): if it has no circuit, or more generally

if the rules in all its strongly connected components have the property of ensuring finite forward chaining (resp. backward chaining), then the forward chaining (resp. backward chaining) is finite; more complex conditions allow to combine several abstract criteria.

However, a weakness of this graph is that it does not take sequences of rule applications into account. Indeed, let us consider a forward chaining mechanism that proceeds in a breadth-first manner, i.e., at each step it performs in parallel all possible rule applications on the current facts, which produces new facts. Rule dependencies focus on a single step of this mechanism and are not able to take several successive steps into account. Indeed, it may be the case that R_j depends on R_i , i.e., an application of R_i adds knowledge that can be used to trigger a new application of R_j , but that this dependency holds only for a bounded number of forward chaining steps; after that, no application of R_i can contribute to trigger R_j (note that this property has to hold for all possible initial facts). In other words, there may be circuits in the graph of rule dependencies that do not correspond to an infinite sequence of rule applications.

The aim of this paper is to introduce more “farsighted” dependencies. We generalize rule dependencies to k -dependencies, that consider k steps of rule applications; rule dependency is the particular case where $k = 1$. Intuitively, R_j k -depends on R_i if there exists a fact such that applying R_i after $k - 1$ steps of forward chaining may trigger a new application of R_j . Note that this notion is independent of any facts. We show that k -dependency can be effectively computed and that the problem of determining if a rule k -depends on another is NP-complete (the same complexity as for rule dependency).

Paper organization. Sect. 2 provides basic definitions and results on forward chaining and rule dependencies. Sect. 3 defines k -dependency and shows that it overcomes some weaknesses of rule dependencies. Sect. 4 is devoted to the computation of k -dependencies. Due to space requirements, we could not include all definitions needed to fully check proofs; they can be found in [BLMS11]; besides, the reader is referred to [BMT11] for a self-contained version of this paper. Sect. 5 outlines further work.

2 Background

2.1 Preliminaries

As usual, an *atom* is of the form $p(t_1, \dots, t_k)$ where p is a predicate with arity k , and the t_i are terms, i.e., variables or constants. A *conjunct* $C[\mathbf{x}]$ is a finite conjunction of atoms, where \mathbf{x} is the set of variables occurring in C . A *fact* is the existential closure of a conjunct. We thus generalize the usual notion of a fact as a ground atom by keeping into account the existential variables generated by rule applications; with this generalization, a finite set of facts is equivalent to a fact, hence we identify both notions. Furthermore, a (Boolean) *conjunctive query* (CQ) has the same form as a fact, thus we also identify these notions. W.l.o.g. we see conjuncts, facts and CQ as *sets* of atoms. Given an atom or a set of atoms A , we denote by $\text{vars}(A)$ and $\text{terms}(A)$ its set of variables and of terms, respectively. Given conjuncts F and Q , a *homomorphism* π from Q to F is a substitution of $\text{vars}(Q)$ by $\text{terms}(F)$ such that $\pi(Q) \subseteq F$ (we say that Q maps to F by π). Logical consequence is denoted by \models . It is well-known that, given two facts F and Q , it holds that $F \models Q$ iff there is a homomorphism from Q to F .⁴

⁴ We consider here standard logical entailment, i.e., with arbitrary (finite or infinite) models.

Definition 1 ($\forall\exists$ -Rule). A $\forall\exists$ -rule (or existential rule, or simply rule when not ambiguous) is a formula $R = \forall\mathbf{x}\forall\mathbf{y}(B[\mathbf{x}, \mathbf{y}] \rightarrow \exists\mathbf{z}H[\mathbf{y}, \mathbf{z}])$ where $B = \text{body}(R)$ and $H = \text{head}(R)$ are conjuncts, resp. called the body and the head of R . The frontier of R , noted $\text{fr}(R)$, is the set of variables $\text{vars}(B) \cap \text{vars}(H) = \mathbf{y}$.

We represent rules as pairs of the form $\text{body}(R) \rightarrow \text{head}(R)$ of sets of atoms, with implicit quantifiers; a and b denote constants, and x, y, z denote variables.

Definition 2 (Application of a Rule). A rule R is applicable to a fact F if there is a homomorphism π from $\text{body}(R)$ to F ; the result of the application of R on F w.r.t. π is a fact $\alpha(F, R, \pi) = F \cup \pi^{\text{safe}}(\text{head}(R))$ where π^{safe} is a substitution of $\text{head}(R)$, which replaces each $x \in \text{fr}(R)$ with $\pi(x)$ and the other variables with fresh variables.

Example 1. Let $R = p(x) \rightarrow r(x, y)$ be a rule and $F = \{p(a), p(b)\}$ be a fact. The homomorphisms from $\text{body}(R)$ to F are $\pi_1 = \{(x, a)\}$ and $\pi_2 = \{(x, b)\}$. The first application produces the atom $r(a, y_1)$, where y_1 is fresh. Then, $F' = \alpha(F, R, \pi_1) = \{p(a), p(b), r(a, y_1)\}$. By applying R to F or F' w.r.t. π_2 , one produces $r(a, y_2)$.

Definition 3 (Derivation). Let F be a fact, and \mathcal{R} be a set of rules. An \mathcal{R} -derivation of F is a finite sequence $(F_0 = F), \dots, F_k$ s.t. for all $0 \leq i < k$, $F_{i+1} = \alpha(F_i, R_i, \pi_i)$, where $R_i \in \mathcal{R}$ and π_i is a homomorphism from $\text{body}(R_i)$ to F_i .

Theorem 1 (Forward Chaining). Let F and Q be two facts, and \mathcal{R} be a set of rules. Then $F, \mathcal{R} \models Q$ iff there exists an \mathcal{R} -derivation $(F_0 = F), \dots, F_k$ such that $F_k \models Q$.

It follows that a breadth-first forward chaining algorithm, which at each step checks if Q can be mapped to the current fact, and if not performs all possible rule applications on the current fact to produce a new fact, yields a positive answer in finite time when $F, \mathcal{R} \models Q$. See Alg. 1 (FC) for a generic implementation. The input G is assumed to encode an optimization structure based on rule dependencies and can be ignored for now. Let F_0 be the initial fact F . Each step consists of producing a fact F_i from F_{i-1} . First, all new homomorphisms from each rule body to F_{i-1} are computed (the call to *ruleApplicationsToCheck*(F, \mathcal{R}, G) builds the set toDo). By *new* homomorphism to F_{i-1} , we mean a homomorphism that has not been already computed at a previous step, i.e., that uses at least an atom added at rank $i - 1$ ($i \geq 2$). Second, the fact F_i is produced by performing the rule applications using these homomorphisms, provided that they are considered as *useful*, which can be encoded in the *changesSomething*(F, R, π) predicate. In its simplest form, this predicate always returns true; in this case, FC corresponds to the so-called *oblivious chase* in databases. A stronger halting criterion –in the sense that it may allow to stop sooner and in more cases– is that the predicate returns false if the application of R w.r.t. π is locally redundant, i.e., π is extensible to a homomorphism from the head of R to F_{i-1} ; in this case, FC corresponds to so-called *restricted chase* in databases. An even stronger criterion is that the predicate returns false if $\alpha(F, R, \pi)$ is equivalent to F . We call *finite expansion set* (fes) a set of rules for which FC halts on any fact F with the last criterion (see [BM02] for further details). In this paper, the criterion chosen does not matter: in the following results, when FC is shown to be finite, this holds for any of these criteria.

Notations. F_k is called the \mathcal{R} -saturation of F at rank k and is denoted by $\alpha_k(F, \mathcal{R})$. An atom has *order* k if it belongs to $F^k \setminus F^{k-1}$ (i.e., it has been created at rank k).

Algorithm 1: FC: Generic Forward Chaining Algorithm

Data: \mathcal{R} , F and Q ; optionally, a structure G encoding dependencies in \mathcal{R}
Result: TRUE if $F, \mathcal{R} \models Q$, FALSE (or infinite calculus) otherwise

```

for ( $rank = 1$ ; ; $rank++$ ) do
  if  $Q$  maps to  $F$  then
     $\perp$  return TRUE;
   $toDo \leftarrow ruleApplicationsToCheck(F, \mathcal{R}, G)$ ;
   $hasChanged \leftarrow FALSE$ ;
  for  $(R, \pi) \in toDo$  do
    if  $changesSomething(F, R, \pi)$  then
       $F \leftarrow \alpha(F, R, \pi)$ ;
       $G \leftarrow updateOptimizationStructure(G, R, \pi)$ ;
       $hasChanged \leftarrow TRUE$ ;
    if not  $hasChanged$  then
       $\perp$  return FALSE;

```

2.2 Dependencies

Several conditions ensuring safe interactions between rules have been defined in the literature. They can be grouped in two families, which rely on two different graphs.

The first family of conditions relies on a graph encoding variable sharing between positions in predicates. This graph, called (*position*) *dependency graph*, was introduced for TGDs.⁵ Its nodes represent positions in predicates ((p, i) represents position i in predicate p). Then, for each rule R and each variable x in $body(R)$ occurring in position (p, i) , the following arcs with origin (p, i) are created: if $x \in fr(R)$, there is an arc from (p, i) to each position of x in $head(R)$; furthermore, for each existential variable y in $head(R)$ (i.e., $y \in vars(head(R)) \setminus fr(R)$) occurring in position (q, j) , there is a special arc from (p, i) to (q, j) . A set of rules is said to be *weakly acyclic* (*wa*) if its position dependency graph has no circuit passing through a special arc. Intuitively, such a circuit means that the introduction of an existential variable in a given position may lead to create another existential variable in the same position, hence an infinite number of existential variables. The weak-acyclicity property is a sufficient condition (but of course not necessary) for FC to be finite [FKMP03,DT03]. Recently, weak-acyclicity has been independently generalized in various ways, namely *safety* [MSL09], *super-weak-acyclicity* [Mar09], and *joint-acyclicity* [KR11].

Example 2. Let $\mathcal{R}_1 = \{R_1, R_2\}$, with: $R_1 = p(x) \rightarrow r(x, y), r(y, z), r(z, x)$ and $R_2 = r(x, y), r(y, x) \rightarrow p(x)$. In the position dependency graph, among other arcs, there is a special arc from $(p, 1)$ to $(r, 2)$, which intuitively translates the fact that an application of R_1 , which requires to find a value in position $(p, 1)$, leads to create an existential variable (y or z) in position $(r, 2)$. In turn, there is an arc from $(r, 2)$ to $(p, 1)$, which translates the fact that an application of R_2 leads to propagate the value found for variable x in position $(r, 2)$ to position $(p, 1)$. Hence, \mathcal{R}_1 is not *wa*.

⁵ We use here the terminology of [FKMP03], developed in [FKMP05].

The second family of conditions relies on another graph, called the *graph of rule dependencies* (*GRD*), which encodes possible interactions between the rules themselves: the nodes represent the rules and there is an arc from R_i to R_j if an application of the rule R_i may lead to a new application of the rule R_j ; more precisely, there exists a fact F such that the application of R_i on F , leading to F' , may enable a homomorphism from $\text{body}(R_j)$ to F' that is not a homomorphism to F ; we say that R_j depends on R_i . This abstract condition can be effectively computed with a *unification* operation: R_j depends on R_i iff there is a *piece-unifier* between $\text{body}(R_j)$ and $\text{head}(R_i)$ (piece-unifier being a generalization of the usual notion of atom unifier, that takes the complex structure of rule heads into account, hence processes whole subsets of atoms together, see [BLMS09,BLMS11] for details). The GRD notion has been first introduced for conceptual graph rules in [Bag04], then adapted to existential rules in [BLMS09]; the notion of a piece-unifier defined in [BLMS09] is itself adapted from a similar notion defined for backward chaining of conceptual graph rules [SM96]. A notion equivalent to the GRD, called the *chase graph*, is independently defined for TGDs in [DNR08].

It is easily checked that if $\text{GRD}(\mathcal{R})$ is acyclic (i.e., has no circuit), then $\text{FC}(\mathcal{R})$ is finite; moreover, if all strongly connected components of $\text{GRD}(\mathcal{R})$ have finite FC (f.i. the corresponding subsets of rules are *fes*), then $\text{FC}(\mathcal{R})$ is finite [Bag04,BLMS09] (see this latter paper for similar results on backward chaining and decidable combinations of forward and backward chaining). [DNR08] prove that when all strongly connected components of the GRD are weakly-acyclic (the chase graph is said *stratified*), then FC is finite, which can be seen as a special case of the previous result.

Example 2. (continued) \mathcal{R}_1 is not *wa*, however $\text{FC}(\mathcal{R}_1)$ is finite, which can be detected via rule dependencies. Indeed, let us see $\text{head}(R_1)$ and $\text{body}(R_2)$ as graphs, with variables as the nodes, and atoms as arcs from their first argument to their second argument; then, $\text{head}(R_1)$ is a circuit of length three and $\text{body}(R_2)$ is a circuit of length two. In this example, any piece-unifier of $\text{body}(R_2)$ with $\text{head}(R_1)$ is necessarily a homomorphism from $\text{body}(R_2)$ to $\text{head}(R_1)$. Since a circuit of length two cannot be mapped by homomorphism to a circuit of length three, R_2 does not depend on R_1 . Obviously, no rule depends on itself. Thus, $\text{GRD}(\mathcal{R}_1)$ is acyclic ((R_2, R_1) is the only edge).

Example 3. Let $\mathcal{R}_2 = \{R\}$, with $R = p(x), r(x, y) \rightarrow p(y)$. \mathcal{R}_2 is *wa* (there is no special arc) but R depends on itself, thus $\text{GRD}(\mathcal{R}_2)$ is cyclic.

Weak-acyclicity and GRD-acyclicity are incomparable notions, as can be checked on examples 2 and 3. This incomparability still holds for generalizations of weak-acyclicity. Intuitively, rule dependency allows to capture exactly the conditions for rule applicability but only for one step of FC, while position dependency is not as accurate but studies the propagation of variable creations along a whole derivation sequence.

Let us point out that the GRD not only yields sufficient conditions for decidability but can also be used to improve forward chaining (and similarly backward chaining). Indeed, if a subset of rules $\mathcal{R}_c \subseteq \mathcal{R}$ has been applied at step $i - 1$, then the only rules that have to be checked for applicability at step i are the successors of \mathcal{R}_c in the GRD. This can be implemented by the functions *ruleApplicationsToCheck*(F, \mathcal{R}, G) and *updateOptimizationStructure*(G, R, π) in Alg. 1: initially, all rules in the GRD are

flagged for applicability. Then, in *ruleApplicationsToCheck*, we only test for applicability rules that depend on flagged ones, and unflag all rules at the end of the call. Finally, each call to *updateOptimizationStructure*(G, R, π) flags the rule R , which has been usefully applied. Other optimizations can be considered. For example, by integrating F and Q in the GRD (respectively as the rules $\emptyset \rightarrow F$ and $Q \rightarrow \text{win}()$), we can restrict rules to those that appear in a path from F to Q . We call FC^+ the obtained algorithm.⁶

Since deciding whether FC halts (whatever the chosen halting condition is) is not decidable [DNR08, BLM10], dependency conditions can only try to better approximate the set of halting instances. Following example 4 shows a case where FC halts while none of previous notions of acyclicity allows one to detect it.

Example 4. Let $R = p(x), r(x, y) \rightarrow r(y, z)$. $\{R\}$ is not *wa* (nor acyclic with the above mentioned generalizations of *wa*) and $\text{GRD}(\{R\})$ has a loop. However, the number of steps of $\text{FC}(\{R\})$ is bounded by 2, since R does not create any atom with predicate p . E.g., with $F = \{p(a), r(a, b), p(b)\}$, the first step adds $r(b, z_1)$ and the second step adds $r(z_1, z_2)$; after that, R is not applicable anymore.

3 From rule dependencies to rule k -dependencies

As seen in example 4, rule dependency does not consider “how long” a dependency exists. Indeed, R triggers itself, but it can do so only once. We thus introduce the notion of k -dependency which allows one to deal with such phenomenon. Intuitively, k -dependency checks if two rules are dependent, after having restrained the set of facts on which this dependence could appear to facts that can be obtained with k steps of FC.

3.1 Definition of k -dependency and link with the usual dependency

Definition 4 (*k -dependency*). Let \mathcal{R} be a set of rules, $k \geq 1$ and $R_i, R_j \in \mathcal{R}$. We say that $R_j = (B_j, H_j)$ is dependent at order k on R_i among \mathcal{R} (notation: R_j k -depends on R_i) if there exists a fact F and a homomorphism π from B_j to $\alpha_k(F, \mathcal{R})$ such that there exists an atom a of B_j that is mapped to an atom of order k generated by R_i . When \mathcal{R} is clear from the context, we will only write that R_j is k -dependent on R_i .

The k -dependency relationships are compiled in the *graph of rule dependencies at order k* , in a similar way as for the dependency relationships.

Definition 5 ($\text{GRD}_k(\mathcal{R})$). Let \mathcal{R} be a set of rules. The graph of rule dependencies at order k of \mathcal{R} has as nodes the elements of \mathcal{R} , and possesses an arc from R_i to R_j iff R_j is dependent at order k on R_i . It is denoted by $\text{GRD}_k(\mathcal{R})$.

We first prove that k -dependency is a refinement of the usual dependency, by showing that there are less rules k -dependent than dependent.

⁶ Furthermore, as shown in [BS06], the GRD can be used not only to reduce the number of rule applicability checks, but also the computation time of these checks: if R is a rule applied at step k according to some homomorphism π , and μ_1, \dots, μ_p are the piece-unifiers proving that R' depends on R , then, at step $k + 1$, we only have to check for each partial homomorphism $\pi \circ \mu_i$ from the body B' of R' if it can be extended to a homomorphism from B' to $\alpha_k(F, \mathcal{R})$.

Property 1. Let \mathcal{R} be a set of rules. The following holds for any $1 \leq k \leq k'$:

$$GRD_{k'}(\mathcal{R}) \subseteq GRD_k(\mathcal{R}) \subseteq GRD_1(\mathcal{R}) = GRD(\mathcal{R})$$

Proof. We first show that if R_j k' -depends on R_i , then it k -depends on R_i for any $k \leq k'$. By definition of k' -dependency, there exists a fact F and a homomorphism π from B_j to $\alpha_{k'}(F, \mathcal{R})$ such that there exists an atom a of B_j that is mapped to an atom of order k generated by R_i . Then, a is an atom of order k' when the initial fact is $\alpha_{k'-k}(F, \mathcal{R})$, thus, R_j k -depends on R_i . We now show that $GRD_1(\mathcal{R}) = GRD(\mathcal{R})$. If R_j depends (usual notion) on R_i , then there is a fact F and a homomorphism from B_j to F such that a new application of R_j exists in $\alpha(F, R_i, \pi)$. This new homomorphism necessarily uses an atom newly created, thus is still new if we perform all possible rule applications from \mathcal{R} to F at the first step. Thus, R_j 1-depends on R_i . If R_j 1-depends on R_i , then let F be a fact to which B_j maps in a new way using a created by an application π of R_i . Let $F' = \alpha(F, \mathcal{R}) \setminus a$. R_i is applicable on F' by π , and B_j can be mapped to $\alpha(F', R_i, \pi)$ in a way that was not possible on F' , thus R_j depends on R_i .

We can use the k -GRD to optimize the FC algorithm, in a similar way as with the GRD. Suppose our optimization structure G contains *some* k -GRDs. Then, at rank k of the algorithm, we can use the q -GRD in G such that $q \leq k$ and q is maximal for this property. We then use that q -GRD exactly as we used the GRD in FC^+ .

3.2 Decidability properties related to the structure of $GRD_k(\mathcal{R})$

We now focus on decidability properties related to $GRD_k(\mathcal{R})$, which are generalizations of properties of $GRD(\mathcal{R})$.

Property 2. Let \mathcal{R} be a set of rules. If there exists k such that $GRD_k(\mathcal{R})$ is acyclic, then \mathcal{R} is a finite expansion set.

Proof. Let F be a fact and \mathcal{R} be a set of rules having an acyclic GRD_k . Let $F_k = \alpha_k(F, \mathcal{R})$. Let \mathcal{R}_1 be the set of rules that can be applied in a new way to F_k . The rules that could be applied in a new way to F^{k+l} are those that can be reached with a path of length l from a rule of \mathcal{R}_1 in $GRD_k(\mathcal{R})$. Since $GRD_k(\mathcal{R})$ is acyclic, there exists m such that m is the length of the longest path in $GRD_k(\mathcal{R})$. We then have $F_{k+m} = F_{k+m+1}$, thus \mathcal{R} is a fes.

This yields a strictly more general criterion than the acyclicity of $GRD(\mathcal{R})$ to determine if a given set of rules is a fes. Indeed, the set $\{R\}$ in example 4 has an acyclic GRD_2 but not an acyclic GRD . However, this criterion does not generalize weak-acyclicity, as can be checked on example 3, where $GRD_k(\mathcal{R}_2)$ has a loop for any k . In order to subsume this class (and any other (monotonic) fes criteria), we generalize a property that holds for the GRD as follows:

Property 3 (Decomposition into s.c.c). Let \mathcal{R} be a set of rules. If there exists k such that all strongly connected components of $GRD_k(\mathcal{R})$ are finite expansion sets, then \mathcal{R} is a finite expansion set.

In particular, if each s.c.c. of $GRD_k(\mathcal{R})$ is a set of weakly-acyclic rules, then \mathcal{R} is fes.

Proof. We adapt the proof of [BLMS11], Theorem 17. The only difference is that instead of starting to use GRD_k from F , we use it from $\alpha_k(F, \mathcal{R})$.

3.3 Dependency and syntactic transformation

We finally present a nice feature of k -dependency related to syntactic transformations. First, note that in Datalog, the head of a rule is a single atom. Such a restriction of rule head cardinality can be done without loss of generality when rules have no existential variables; indeed, in this case, a rule can be decomposed into an equivalent set of rules by splitting its head into single atoms. In the case of existential rules, such an equivalent rewriting can still be done, at the cost of introducing some auxiliary atoms (see the transformation below), which are used to memorize multi-occurrences of an existential variable in a rule head.

Definition 6 (Atomic head transformation). Let $R = (B, H = \{h_1, \dots, h_k\})$ be a rule. We define $\mathcal{T}_a(R) = \{R^0, R^1, \dots, R^k\}$, where:

- $R^0 = (B, \{p_R(\mathbf{x})\})$, where \mathbf{x} is the set of variables appearing in H and p_R a fresh predicate,
- for any $1 \leq i \leq k$, $R^i = (\{p_R(\mathbf{x})\}, \{h_i\})$.

For a rule set \mathcal{R} , we define $\mathcal{T}_a(\mathcal{R}) = \cup_{R \in \mathcal{R}} \mathcal{T}_a(R)$.

Example 5. Let \mathcal{R}_1 be the set of rules in Example 2. $\mathcal{T}_a(\mathcal{R}_1)$ contains the following rules (note that R_2 could be kept unchanged since its head is already atomic):

- $R_1^0 = p(x) \rightarrow p_{R_1}(x, y, z)$
- $R_1^1 = p_{R_1}(x, y, z) \rightarrow r(x, y)$
- $R_1^2 = p_{R_1}(x, y, z) \rightarrow r(y, z)$
- $R_1^3 = p_{R_1}(x, y, z) \rightarrow r(z, x)$
- $R_2^0 = r(x, y), r(y, x) \rightarrow p_{R_2}(x)$
- $R_2^1 = p_{R_2}(x) \rightarrow p(x)$

$R_1 \longleftarrow R_2$

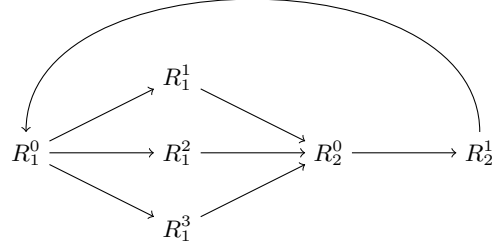


Fig. 1. The GRD of \mathcal{R}_1 (left) and of $\mathcal{T}_a(\mathcal{R}_1)$ (right)

However, as illustrated in Figure 1, even if both rule sets are equivalent from a semantic point of view, they do not behave similarly with respect to the graph of rule dependencies: indeed, the *GRD* of the initial rule set has no circuit while the *GRD* of the new rule set possesses one. This is due to the arcs (R_1^i, R_2^0) for $i = 1 \dots 3$, while there is no arc (R_1, R_2) . In other words, the rule dependency notion is not resistant to the atomic-head transformation. This trouble is solved with the k -dependency notion, thanks to its ability to see beyond a single FC step. Indeed, the structure of $\text{GRD}_{2k}(\mathcal{T}_a(\mathcal{R}))$ is similar to $\text{GRD}_k(\mathcal{R})$, in a sense specified by the following property:

Property 4 (Structural similarity). Let \mathcal{R} be a set of rules, and $\mathcal{T}_a(\mathcal{R})$ be obtained as above. Let R_i and R_j be two rules of \mathcal{R} . There exists m such that R_j^0 $2k$ -depends on R_i^m iff R_j k -depends on R_i .

Proof. See Lemmas 3 and 4 in Appendix.

This structural similarity of $\text{GRD}_k(\mathcal{R})$ and $\text{GRD}_{2k}(\mathcal{T}_a(\mathcal{R}))$ yields the following corollary:

Corollary 1. *Let \mathcal{R} be a set of rules. $\text{GRD}_k(\mathcal{R})$ is acyclic iff $\text{GRD}_{2k}(\mathcal{T}_a(\mathcal{R}))$ is acyclic.*

More generally, the reduced graph of $\text{GRD}_k(\mathcal{R})$ –according to its s.c.c.– is isomorphic to the reduced graph of $\text{GRD}_{2k}(\mathcal{T}_a(\mathcal{R}))$.

4 Computation of k -dependencies

While forward chaining uses rules to enrich facts and produce a fact to which the query maps, backward chaining proceeds in the “reverse” manner: it uses the rules to rewrite the query in different ways and produce a query that maps to the facts. The key operation in a backward chaining mechanism is the *unification* operation between part of a current goal (a fact in our framework) and a rule head. This mechanism is typically used in logic programming, with rules having a single atom in the head, which is unified with an atom of the current goal. Since the head of an existential rule has a more complex structure, the associated unification operation is also more complex. We rely on the notion of a *piece*, which stems from a graph view of rules and was introduced in [SM96] for conceptual graph rules. We recall here the definitions of a rewriting and a rewriting sequence, which are necessary to understand this section (since the formal definition of a piece-unifier is only needed to fully check proofs, we refer the reader to [BLMS11] or [BMT11] for it). Then, we generalize these notions to effectively compute k -dependencies, and study the complexity of the associated decision problem.

4.1 Rewriting and rewriting sequences

As seen in Section 2, the forward chaining α operator enriches facts by drawing conclusions from a rule application, which is determined by a given homomorphism. Indeed, if F is a fact, $R = (B, H)$ is a rule, and π is a homomorphism from B to F , then $\alpha(F, R, \pi) = F \cup H_\pi$, where H_π is a specialization of H determined by π . On the other hand, backward chaining relies upon the β operator that uses a *piece-unifier* to rewrite the goal. Indeed, if Q is a fact (goal), $R = (B, H)$ is a rule, and μ is a unifier of Q with R , then $\beta(Q, R, \mu) = Q_\mu \cup B_\mu$ where Q_μ is a specialization of a subset of Q determined by the unifier, and B_μ is a specialization of the body of R (also determined by μ). Note that more than one atom of Q can be suppressed in Q_μ (in fact, subsets of atoms corresponding to *pieces* are erased), and we can restrict our work to the case when at least one atom is erased (using *atom-erasing* unifiers).

The following lemmas, rephrased from [BLMS11] (Lemmas 7 and 8) using the above notations, state the precise relationships between the β and α operators.

Lemma 1. *Let $Q' = \beta(Q, R, \mu)$. Then there exists a homomorphism π_μ from the body of R to Q' such that Q maps to $\alpha(Q', R, \pi_\mu)$.*

Lemma 2. *Let $F' = \alpha(F, R, \pi) \neq F$. Then there exists a piece-unifier μ_π of F' with R such that $\beta(F', R, \mu_\pi)$ maps to F .*

Rewriting Sequence These two lemmas are then used to prove the correspondence between the forward chaining algorithm (using a sequence of rule applications) and the backward chaining algorithm (that relies upon a sequence of rewritings).

Definition 7 (Rewriting sequence). *Let Q and Q' be two facts, and \mathcal{R} be a set of rules. We say that Q' is an \mathcal{R} -rewriting of Q if there is a finite sequence (called the rewriting sequence) $Q = Q_0, Q_1, \dots, Q_k = Q'$ such that for all $1 \leq i \leq k$, there is a rule $R \in \mathcal{R}$ and a unifier μ of Q_{i-1} with R such that $Q_i = \beta(Q_{i-1}, R, \mu)$.*

Theorem 2. [BLMS11] *Let F and Q be two facts, and \mathcal{R} be a set of rules. There is an \mathcal{R} -rewriting of Q that maps to F iff there is an \mathcal{R} -derivation sequence from F to Q such that Q maps to F .*

Unifiers and dependencies The unifiers used to define fact rewritings are also used to compute dependencies, as stated by the following theorem.

Theorem 3. [BLMS11] *A fact Q depends on a rule R if and only if there exists an atom-erasing unifier of Q with R , i.e., a unifier μ such that $Q \not\subseteq \beta(Q, R, \mu)$.*

4.2 Rewriting depth and k -dependency

In section 3, we have defined k -dependency with respect to a particular order of rule-applications in forward chaining, using a “saturation” mechanism. Wanting now to generalize Theorem 3 to be able to compute k -dependencies, we will have to consider particular rewritings that correspond to that particular order in forward chaining. In the following definitions, we prepare that by examining how atoms appear and disappear in a rewriting sequence.

Definition 8 (Atom Erasure and Creation). *Let $\mathcal{S} = Q_0, \dots, Q_k$ be an \mathcal{R} -rewriting sequence from Q_0 to Q_k . We say that Q_i erases an atom a in \mathcal{S} when $a \in Q_{i-1}$ but $a \notin Q_i$ (note that Q_0 erases no atom). We say that Q_i creates an atom a in \mathcal{S} when $a \in Q_i$ but $a \notin Q_{i-1}$ (note that Q_0 creates all its atoms). We say that Q_i requires Q_j (with $j < i$) when Q_i erases an atom created by Q_j .*

We first generalize the notion of atom-erasing unifiers used to characterize dependencies.

Definition 9 (Atom-erasing sequence). *Let $\mathcal{S} = Q_0, \dots, Q_k$ be an \mathcal{R} -rewriting sequence from Q_0 to Q_k . We say that \mathcal{S} is atom-erasing when, $\forall 1 \leq i \leq k$, Q_i erases at least one atom that is not created again by some Q_j with $j \geq i$.*

Note that a sequence Q_0, Q_1 is atom-erasing if and only if the unifier used to obtain Q_1 from Q_0 is atom-erasing in the sense of [BLMS11]. The notion of atom-erasing sequence is indeed a generalization of atom-erasing unifiers.

Property 5. If there exists an \mathcal{R} -rewriting sequence \mathcal{S} from Q to Q' such that Q' maps to F , then there exists an *atom-erasing* \mathcal{R} -rewriting sequence \mathcal{S}' from Q to Q'' such that Q'' maps to F .

Proof. See that if \mathcal{S} is not atom-erasing, then there exists Q_i that either erases no atom, or such that all atoms erased by Q_i appear again in the rewriting sequence. This rewriting only adds information to what has been proven, and is thus useless.

As a consequence, when considering rewriting sequences, we can restrict our search to atom-erasing ones. The depth of a rewriting will be used to establish a correspondence between rewritings and the saturation rank in forward chaining.

Definition 10 (Depth of a fact). Let $\mathcal{S} = Q_0, \dots, Q_k$ be an \mathcal{R} -rewriting sequence from Q_0 to Q_k . The depth of a fact in \mathcal{S} is recursively defined as follows:

- if no fact of \mathcal{S} is required by Q_i , then $\text{depth}(Q_i) = 0$;
- otherwise, $\text{depth}(Q_i) = \max_{Q_j} \text{required by } Q_i \{ \text{depth}(Q_j) \}$

The maximal depth of Q_i is the maximal depth of all Q_j , for $i \leq j \leq k$.

Theorem 4. Let $\mathcal{S} = Q_0, \dots, Q_k$ be an atom erasing \mathcal{R} -rewriting sequence from Q_0 to Q_q , with $q \geq 1$. If Q_i has depth 0 in \mathcal{S} , then there is a homomorphism from Q_i to Q_q . Otherwise, if Q_i has maximal depth k , then there is a homomorphism from Q_i to $\alpha_k(Q_q, \mathcal{R})$ that is not a homomorphism from Q_i to $\alpha_{k-1}(Q_q, \mathcal{R})$.

Proof. We prove that theorem by induction of the depth of the rewriting. If Q_i has depth 0 in \mathcal{S} , it means that no Q_j of \mathcal{S} (for $j > i$) requires any atom of Q_i , thus in particular $Q_q = Q_i \cup X$ and there is a homomorphism from Q_i to Q_q . Suppose now that the property is true at rank n . We prove that the property remains true at rank (maximal depth) $n + 1$. Suppose that Q_i has maximal depth $n + 1$. We consider Q_j with the greatest $j \geq i$ with Q_j having depth $n + 1$. Then there exists a Q_p that requires Q_j and has depth n (and also maximum depth n , having chosen j as the greatest). According to our induction hypothesis, there is a homomorphism from Q_p to $F^n = \alpha_n(Q_q, \mathcal{R})$ that is not a homomorphism from Q_p to $\alpha_{n-1}(Q_q, \mathcal{R})$.

It remains now to check (i) that there is a homomorphism from Q_i to the fact $\alpha_1(F^n, \mathcal{R}) = \alpha_{n+1}(Q_q, \mathcal{R})$, and (ii) this homomorphism is not a homomorphism from Q_i to F^n .

(i) We know (soundness and completeness) that there is a homomorphism from Q_i to a finite saturation of Q_q . If it is not in $\alpha_{n+1}(Q_q, \mathcal{R})$, it means that the first homomorphism is at least in $\alpha_2(F^n, \mathcal{R})$. Then for any rule application sequence from F^n to $\alpha_2(F^n, \mathcal{R})$, there is one rule that uses for application an atom that appears in $\alpha_1(F^n, \mathcal{R})$. This is true in particular for the rule application sequence that corresponds (see proof of Theorem 2 in [BLMS11]) to the rewriting sequence from Q_i to Q_p . It would mean (Lemma 2), that the depth of Q_i is at least $2 + \text{depth}(Q_p)$, which is absurd. Thus Q_i necessarily maps to $\alpha_{n+1}(Q_q, \mathcal{R})$.

(ii) See now that there is an atom of Q_j that is erased in Q_p . Thus in the rule application sequence that corresponds to our rewriting sequence, the body of the rule used to obtain a map of Q_j uses an atom created by the rule used to obtain Q_p . The rule application sequence used to obtain the mapping of Q_i thus relies upon the saturation at rank $n + 1$.

The latter theorem can thus be used to characterize k -dependencies.

Corollary 2. *Let \mathcal{R} be a set of rules, and R_1, R_2 be two rules of \mathcal{R} . Then R_2 k -depends on R_1 if and only if there exists an atom erasing \mathcal{R} -rewriting sequence \mathcal{S} from $Q_0 = B_2$ to Q_q such that $Q_1 = \beta(Q_0, R_1, \mu)$ and Q_1 has depth (and thus maximal depth) $k - 1$ in \mathcal{S} .*

4.3 Complexity of k -dependencies

For any k , we define the decision problem called k -DEPENDENCY.

k -DEPENDENCY

Input: a set of rules \mathcal{R} and two rules $R_1, R_2 \in \mathcal{R}$

Output: yes if R_1 k -depends on R_2 among \mathcal{R} , no otherwise.

Property 6. For any k , the decision problem k -DEPENDENCY is NP-complete.

Proof. NP-hardness follows from a straightforward reduction from 1-DEPENDENCY. Let \mathcal{R} be a set of rules, $k \in \mathbb{N}$, and $R_i, R_j \in \mathcal{R}$. From Corollary 2, R_j depends on R_i iff there exists an atom-erasing rewriting sequence such that Q_i is of depth $k - 1$. As a certificate, we provide this rewriting sequence (and all the necessary information to check it is a correct rewriting). If m denotes the maximum size (in number of atoms) of a rule body, the length of a rewriting sequence of depth $k - 1$ is at most m^k . The certificate provides then a polynomial number of facts of size polynomial in Q , and a polynomial number of unifiers. Given a rewriting sequence $\mathcal{S} : Q_0, Q_1, \dots, Q_r$ (including the relevant unifiers), one can check in polynomial time if it is atom-erasing and if Q_1 has maximal depth $k - 1$. Then k -DEPENDENCY belongs to NP.

5 Conclusion

In this paper, we have proposed a generalization of the notion of dependency for existential rules, namely k -dependency, which allows to extend decidable cases for the ontological query answering problem. This notion can be used to compile the rule base, which allows for optimizing online query answering algorithms. We have also shown that this notion resists to the decomposition of rules into rules with atomic head, whereas it was not the case for the simple rule dependency notion. Further work includes the following two directions. First, we are interested in efficient algorithms to decide k -dependency. Second, it is worth to note that the acyclicity of the graph of rule dependencies and of the graph of position dependencies are incomparable criteria, it would be very interesting to define a notion that subsumes both of them. Property 3 allows to build such a class, but doing it in an integrated fashion should give more insight on interactions between rules.

References

- AHV95. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- Bag04. J.-F. Baget. Improving the forward chaining algorithm for conceptual graphs rules. In *KR'04*, pages 407–414. AAAI Press, 2004.

- BBL05. F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. In *IJCAI'05*, pages 364–369, 2005.
- BLM10. J.-F. Baget, M. Leclère, and M.-L. Mugnier. Walking the decidability line for rules with existential variables. In *KR'10*, pages 466–476. AAAI Press, 2010.
- BLMS09. J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *IJCAI'09*, pages 677–682, 2009.
- BLMS11. J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.
- BM02. J.-F. Baget and M.-L. Mugnier. The Complexity of Rules and Constraints. *J. Artif. Intell. Res. (JAIR)*, 16:425–465, 2002.
- BMRT11. J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI'11*, to appear, 2011.
- BMT11. J.-F. Baget, M.-L. Mugnier, and M. Thomazo. Towards Farsighted Dependencies for Existential Rules. Research Report RR-LIRMM 11-016, 2011.
- BS06. J.-F. Baget and E. Salvat. Rules dependencies in backward chaining of conceptual graphs rules. In *ICCS*, volume 4068 of *LNCS*, pages 102–116. Springer, 2006.
- BV81. C. Beeri and M. Vardi. The implication problem for data dependencies. In *ICALP'81*, volume 115 of *LNCS*, pages 73–85, 1981.
- BV84. C. Beeri and M.Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.
- CGK08. A. Cali, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *KR'08*, pages 70–80, 2008.
- CGL⁺07. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
- CGL09. A. Cali, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS'09*, pages 77–86, 2009.
- CGL⁺10. A. Cali, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog⁺:- A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242. IEEE Computer Society, 2010.
- CGP10. A. Cali, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in datalog⁺/. In *RR*, pages 1–17, 2010.
- CLM81. A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *STOC'81*, pages 342–354. ACM, 1981.
- CM09. M. Chein and M.-L. Mugnier. *Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.
- DNR08. A. Deutsch, A. Nash, and J.B. Remmel. The chase revisited. In *PODS'08*, pages 149–158, 2008.
- DT03. A. Deutsch and V. Tannen. Reformulation of xml queries and constraints. In *ICDT'03*, pages 225–241, 2003.
- FKMP03. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT'03*, pages 207–224, 2003.
- FKMP05. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
- JK84. D.S. Johnson and A.C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.
- KR11. M. Krötzsch and S. Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI'11*, to appear, 2011.

- KRH07. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for Horn description logics. In *AAAI'07*, pages 452–457. AAAI Press, 2007.
- LTW09. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic el using a relational database system. In *IJCAI'09*, pages 2070–2075, 2009.
- Mar09. B. Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, pages 13–22, 2009.
- MSL09. M. Meier, M. Schmidt, and G. Lausen. On chase termination beyond stratification. *PVLDB*, 2(1):970–981, 2009.
- SM96. E. Salvat and M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *ICCS'96*, volume 1115 of *LNAI*, pages 248–262. Springer, 1996.
- Sow84. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

Appendix: Proof of Property 4

Lemma 3. *Let \mathcal{R} be a set of rules, $P = \{p_{R_1}, \dots, p_{R_k}\}$ be the set of fresh predicates occurring in $\mathcal{T}_a(\mathcal{R})$. Let Q be a fact without atom of predicate belonging to P . For any Q' without atom of predicate belonging to P it holds that Q' is an \mathcal{R} -rewriting of Q with Q is of depth k iff Q' is a $\mathcal{T}_a(\mathcal{R})$ -rewriting of Q with Q of depth $2k$.*

Proof. (sketch) We explain how to build the $\mathcal{T}_a(\mathcal{R})$ -rewriting sequence given the \mathcal{R} -rewriting sequence, and conversely. Let $\mathcal{S} : Q = Q_0, \dots, Q_q = Q'$ be an \mathcal{R} -rewriting sequence of Q such that Q' has no atom on P and Q is of depth k . We change each rewriting step (say with rule R) by a rewriting sequence of "depth 2" in the following way. We first use the created rules of body $p_R(\mathbf{x})$, then erase all the created atoms by unifying with the R^0 rule. We thus get a $\mathcal{T}_a(\mathcal{R})$ rewriting sequence, for which Q is of depth $2k$. In the other direction, let $\mathcal{S}' : Q = Q'_0, \dots, Q'_{q'} = Q''$ we partition set of facts along the following lines: we put in different sets the facts erasing an atom on P . For a set containing a fact erasing an atom on P , we add the fact that created that atom. These sets of facts correspond to a rewriting step with the rules from \mathcal{R} . We thus build an \mathcal{R} -rewriting sequence, by having facts that erase the atoms erased by the R^i ($i \neq 0$) rules, and create the atoms created by the R^0 rules.

Lemma 4. *Let \mathcal{R} be a set of rules, $P = \{p_{R_1}, \dots, p_{R_k}\}$ be the set of fresh predicates occurring in $\mathcal{T}_a(\mathcal{R})$. Let Q be a fact without atom of predicate belonging to P . If there is an atom-erasing sequence rewriting from Q to Q' of depth $2k$, then there is one such that Q' does not have any atom of predicate belonging to P .*

Proof. Let Q without any atom with predicate belonging to P . Let $\mathcal{S} = Q = Q_0, \dots, Q_q = Q'$ a $\mathcal{T}_a(\mathcal{R})$ rewriting sequence such that Q' is of depth $2k$. Let assume that a is an atom of predicate $p \in P$ in Q' . Let assume p has been created by Q_i (Q_i is necessarily unique since \mathcal{S} is atom-erasing). Q_i creating only p , and p being not used by any other fact, Q_i is not required by any other fact of the rewriting sequence, and we can remove this rewriting step, yielding a fact having strictly less atoms of predicate in P . By induction, we show that there is Q' having no such atom.