ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-39

Extending Acyclicity Notions for Existential Rules

Jean-François Baget[1] and **Fabien Garreau**[2] and **Marie-Laure Mugnier**[3] and **Swan Rocher**[3]

Abstract. Existential rules have been proposed for representing ontological knowledge, specifically in the context of Ontology-Based Query Answering. Entailment with existential rules is undecidable. We focus in this paper on conditions that ensure the termination of a breadth-first forward chaining algorithm known as the chase. First, we propose a new tool that allows to extend existing acyclicity conditions ensuring chase termination, while keeping good complexity properties. Second, we consider the extension to existential rules with nonmonotonic negation under stable model semantics and further extend acyclicity results obtained in the positive case.

1 INTRODUCTION

Ontology-Based Query Answering is a new paradigm in data management, which aims to exploit ontological knowledge when accessing data. *Existential rules* have been proposed for representing ontological knowledge, specifically in this context [8, 3]. These rules allow to assert the existence of unknown individuals, an essential feature in an open-domain perspective. They generalize lightweight description logics, such as DL-Lite and $\mathcal{EL}$ [10, 1] and overcome some of their limitations by allowing any predicate arity as well as cyclic structures.

In this paper, we focus on a breadth-first forward chaining algorithm, known as the *chase* in the database literature [24]. The chase was originally used in the context of very general database constraints, called tuple-generating dependencies, which have the same logical form as existential rules [6].

Given a knowledge base composed of data and existential rules, the chase triggers the rules and materializes performed inferences in the data. The "saturated" data can then be queried like a classical database. This allows to benefit from optimizations implemented in current data management systems. However, the chase is not ensured to terminate— which applies to any sound and complete mechanism, since entailment with existential rules is undecidable ([5, 11] on tuple-generating dependencies). Various acyclicity notions ensuring chase termination have been proposed in knowledge representation and database theory.

Paper contributions. We generalize known acyclicity conditions, first, for plain existential rules, second, for their extension to nonmonotonic negation with stable semantics.

1. Plain existential rules. Acyclicity conditions found in the literature can be classified into two main families: the first one constrains the way existential variables are propagated during the chase, e.g., [15, 25, 18], and the second one constrains dependencies between rules, i.e., the fact that a rule may lead to trigger another rule, e.g., [2, 14, 4, 12]. These conditions are based on different graphs, but all of them can be seen as forbidding "dangerous" cycles in the considered graph. We define a new family of graphs that allows to unify and strictly generalize these acyclicity notions, without increasing worst-case complexity.

2. Extension to nonmonotonic negation. Nonmonotonic negation is a useful feature in ontology modeling. Nonmontonic extensions to existential rules were recently considered in [8] with stratified negation, [17] with well-founded semantics and [23] with stable model semantics. The latter paper focuses on cases where a unique finite model exists; we consider the same rule framework, however without enforcing a unique model. We further extend acyclicity results obtained on positive rules by exploiting negative information as well.

The paper is organized according to these two issues.

2 PRELIMINARIES

An *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate of arity k and the t_i are terms, i.e., variables or constants. An *atomset* is a finite set of atoms. If F is an atom or an atomset, we denote by $\mathit{terms}(F)$ (resp. $\mathit{vars}(F)$) the set of terms (resp. variables) that occur in F. In the examples illustrating the paper, all the terms are variables (denoted by x, y, z, etc.), unless otherwise specified. Given atomsets A_1 and A_2, a *homomorphism* h from A_1 to A_2 is a substitution of $\mathit{vars}(A_1)$ by $\mathit{terms}(A_2)$ such that $h(A_1) \subseteq A_2$.

An *existential rule* (and simply a rule hereafter) is of the form $R = \forall \vec{x} \forall \vec{y}(B \rightarrow \exists \vec{z} H)$, where B and H are conjunctions of atoms, with $\mathit{vars}(B) = \vec{x} \cup \vec{y}$, and $\mathit{vars}(H) = \vec{x} \cup \vec{z}$. B and H are respectively called the *body* and the *head* of the rule. Variables $\vec{x}$, which appear in both B and H, are called *frontier variables*. Variables $\vec{z}$, which appear only in H, are called *existential variables*. Hereafter, we omit quantifiers in rules as there is no ambiguity. E.g., $p(x, y) \rightarrow p(y, z)$ stands for $\forall x \forall y(p(x, y) \rightarrow \exists z(p(y, z)))$.

An existential rule with an empty body is called a *fact*. A fact is thus an existentially closed conjunction of atoms. A *Boolean conjunctive query* (BCQ) has the same form. A *knowledge base* $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts (which is seen as a single fact) F and a finite set of existential rules $\mathcal{R}$. The fundamental problem associated with query answering, called BCQ ENTAILMENT, is the following: given a knowledge base $(F, \mathcal{R})$ and a BCQ Q, is it true that $F, \mathcal{R} \models Q$, where $\models$ denotes the standard logical consequence? This problem is undecidable (which follows from the undecidability of the implication problem on tuple-generating dependencies [5, 11]).

In the following, we see conjunctions of atoms as atomsets. A rule $R : B \rightarrow H$ is *applicable* to an atomset F if there is a homomorphism π from B to F. The *application of R to F w.r.t. π* produces an atomset $\alpha(F, R, \pi) = F \cup \pi(\mathit{safe}(H))$, where $\mathit{safe}(H)$ is obtained from H by replacing existential variables with fresh variables (see

[1] INRIA, France
[2] University of Angers, France
[3] University of Montpellier, France

Example 1).

The *chase* procedure starts from the initial set of facts F and performs rule applications in a breadth-first manner. Several chase variants can be found in the literature, mainly *oblivious* (or naive), e.g., [7], *skolem* [25], *restricted* (or standard) [15], and *core* chase [14]. The oblivious chase performs all possible rule applications. The *skolem chase* relies on a skolemisation of the rules (notation *sk*): for each rule R, *sk*(R) is obtained from R by replacing each occurrence of an existential variable y with a functional term $f_y^R(\vec{x})$, where $\vec{x}$ is the set of frontier variables in R. Then, the oblivious chase is run on skolemized rules.

Example 1 (Oblivious / Skolem chase) *Let $R = p(x,y) \rightarrow p(x,z)$ and $F = \{p(a,b)\}$, where a and b are constants. The oblivious chase does not halt: it applies R according to $h_0 = \{(x,a),(y,b)\}$, hence adds $p(a,z_0)$; then, it applies R again according to $h_1 = \{(x,a),(y,z_0)\}$, and adds $p(a,z_1)$, and so on. The skolem chase considers the rule $p(x,y) \rightarrow p(x,f_z^R(x))$; it adds $p(a,f_y^R(a))$ then halts.*

Due to space restrictions, we do not detail on the other chase variants. Given a chase variant C, we call *C-finite* the class of set of rules $\mathcal{R}$, such that the C-chase halts on $(F,\mathcal{R})$ for any atomset F. It is well-known that oblivious-finite $\subset$ skolem-finite $\subset$ restricted-finite $\subset$ core-finite (see, e.g., [26]). When $\mathcal{R}$ belongs to a C-finite class, BCQ ENTAILMENT can be solved, for any F and Q, by running the C-chase on $(F,\mathcal{R})$, which produces a saturated set of facts F^*, then checking if $F^* \models Q$.

3 KNOWN ACYCLICITY NOTIONS

Acyclicity notions can be divided into two main families, each of them relying on a different graph. The first family relies on a graph encoding variable sharing between *positions* in predicates, while the second one relies on a graph encoding *dependencies* between rules, i.e., the fact that a rule may lead to trigger another rule (or itself).

3.1 Position-based approach

In the position-based approach, dangerous cycles are those passing through positions that may contain existential variables; intuitively, such a cycle means that the creation of an existential variable in a given position may lead to creating another existential variable in the same position, hence an infinite number of existential variables. Acyclicity is then defined by the absence of dangerous cycles. The simplest acyclicity notion in this family is that of *weak-acyclicity (wa)* [15], which has been widely used in databases. It relies on a directed graph whose nodes are the positions in predicates (we denote by (p,i) position i in predicate p). Then, for each rule $R : B \rightarrow H$ and each frontier variable x in B occurring in position (p,i), edges with origin (p,i) are built as follows: there is an edge from (p,i) to each position of x in H; furthermore, for each existential variable y in H occurring in position (q,j), there is a special edge from (p,i) to (q,j). A set of rules is weakly-acyclic if its associated graph has no cycle passing through a special edge (see Example 2).

Example 2 (Weak-acyclicity) *Let $R_1 = h(x) \rightarrow p(x,y)$ and $R_2 = p(u,v), q(v) \rightarrow h(v)$. The position graph of $\{R_1,R_2\}$ contains a special edge from $(h,1)$ to $(p,2)$ due to R_1 and an edge from $(p,2)$ to $(h,1)$ due to R_2. Thus $\{R_1,R_2\}$ is not wa.*

Weak-acyclicity has been generalized, mainly by shifting the focus from positions to existential variables (*joint-acyclicity (ja)*[18]) or to positions in atoms instead of predicates (*super-weak-acyclicity (swa)* [25]). Other related notions can be imported from logic programming, e.g., *finite domain (fd)* [9] and *argument-restricted (ar)* [22]. See the first column in Figure 1, which shows the inclusions between the corresponding classes of rules; all these inclusions are known to be strict.

3.2 Rule dependency-based approach

In the second approach, the aim is to avoid cyclic triggering of rules [2, 14, 3, 4, 12]. We say that a rule R_2 *depends* on a rule R_1 if an application of R_1 may lead to a new application of R_2: there exists an atomset F such that R_1 is applicable to F with homomorphism π and R_2 is applicable to $F' = \alpha(F,R_1,\pi)$ with homomorphism π', which is *new* (π' is not a homomorphism to F) and *useful* (π' cannot be extended to a homomorphism from H_2 to F'). This abstract dependency relation can be computed with a unification operation known as piece-unifier [3]. Piece-unification takes existential variables into account, hence is more complex than the usual unification between atoms. A *piece-unifier* of a rule body B_2 with a rule head H_1 is a substitution μ of *vars*$(B_2') \cup$ *vars*(H_1'), where $B_2' \subseteq B_2$ and $H_1' \subseteq H_1$, such that: *(1)* $\mu(B_2') = \mu(H_1')$, and *(2)* existential variables in H_1' are not unified with *separating* variables of B_2', i.e., variables that occur both in B_2' and in $(B_2 \setminus B_2')$; in other words, if a variable x occuring in B_2' is unified with an existential variable y in H_1', then all atoms in which x occur also belong to B_2'. It holds that R_2 depends on R_1 iff there is a piece-unifier of B_2 with H_1, satisfying some easily checked additional conditions (atom erasing [4] and usefulness [19, 12]). Following Example 3 illustrates the difference between piece-unification and classical unification.

Example 3 (Rule dependency) *Let R_1 and R_2 from Example 2. Although the atoms $p(u,v) \in B_2$ and $p(x,y) \in H_1$ are unifiable, there is no piece-unifier of B_2 with H_1. Indeed, the most general unifier $\mu = \{(u,x),(v,y)\}$ (or, equivalently, $\{(x,u),(y,v)\}$), with $B_2' = \{p(u,v)\}$ and $H_1' = H_1$, is not a piece-unifier because v is unified with an existential variable, whereas it is a separating variable of B_2' (thus, $q(v)$ should be included in B_2'). It follows that R_2 does not depend on R_1.*

The *graph of rule dependencies* of a set of rules $\mathcal{R}$, denoted by GRD($\mathcal{R}$), is a directed graph with set of nodes $\mathcal{R}$ and an edge (R_i,R_j) if R_j depends on R_i. E.g., with the rules in Example 3, the only edge is (R_2,R_1). When the GRD is acyclic (*aGRD*, [2]), any derivation sequence is finite.

3.3 Combining both approches

Both approaches are incomparable: there may be a dangerous cycle on positions but no cycle w.r.t. rule dependencies (Example 2 and 3), and there may be a cycle w.r.t. rule dependencies whereas rules have no existential variables (e.g., $p(x,y) \rightarrow p(y,x)$). So far, attempts to combine both notions only succeded to combine them in a "modular way", by considering the strongly connected components (s.c.c.) of the GRD [2, 14]; briefly, if a chase variant stops on each subset of rules associated with a s.c.c., then it stops on the whole set of rules. In this paper, we propose an "integrated" way of combining both approaches, which relies on a single graph. This allows to unify preceding results and to generalize them without increasing complexity.

The new acyclicity notions are those with a gray background in Figure 1.

Finally, let us mention *model-faithful acyclicity (mfa)* [12], which cannot be captured by our approach. Briefly, checking *mfa* involves running the skolem chase until termination or a cyclic functional term is found. The price to pay is high complexity: checking if a set of rules is model-faithful acyclic for any set of facts is 2EXPTIME-complete. Checking *model-summarizing acyclicity (msa)* [12], which approximates mfa, remains EXPTIME-complete. In contrast, checking position-based properties is in PTIME and checking aGRD is co-NP-complete.

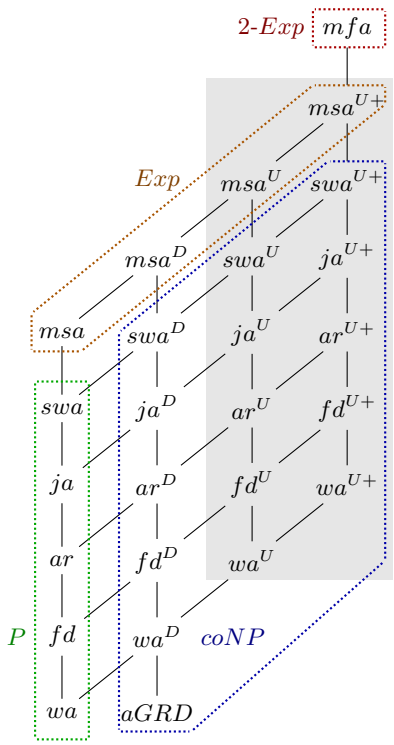

Figure 1. Relations between recognizable acyclicity properties. All inclusions are strict and complete (i.e., if there is no path between two properties then they are incomparable).

It remains to specify for which chase variants the above acyclicity notions ensure termination. Since *mfa* generalizes all properties in Figure 1, and sets of rules satisfying *mfa* are skolem-finite, all these properties ensure C-finiteness, for any chase variant C at least as strong as the skolem chase. We point out that the oblivious chase may not stop on *wa* rules. Actually, the only acyclicity notion in Figure 1 that ensures the termination of the oblivious chase is *aGRD*, since all other notions generalize *wa*.

4 EXTENDING ACYCLICITY NOTIONS

In this section, we combine rule dependency and propagation of existential variables into a single graph. W.l.o.g. we assume that distinct rules do not share any variable. Given an atom $a = p(t_1, \ldots, t_k)$, the i^{th} position in a is denoted by $[a, i]$, with $\mathit{pred}([a, i]) = p$ and $\mathit{term}([a, i]) = t_i$. If A is an atomset such that $a \in A$, we say that $[a, i]$ is in A. If $\mathit{term}([a, i])$ is an existential (resp. frontier) variable, $[a, i]$ is called an *existential* (resp. *frontier*) position. In the following, we use "position graph" as a generic name to denote a graph whose nodes are positions in *atoms*. We define several position graphs of increasing expressivity, i.e., allowing to check termination for increasingly larger classes of rules.

Definition 1 ((Basic) Position Graph (PG)) *The* position graph *of a rule $R : B \to H$ is the directed graph $PG(R)$ defined as follows:*

- *there is a node for each $[a, i]$ in B or in H;*
- *for all frontier positions $[b, i] \in B$ and all $[h, j] \in H$, there is an edge from $[b, i]$ to $[h, j]$ if $\mathit{term}([b, i]) = \mathit{term}([h, j])$ or if $[h, j]$ is existential.*

Given a set of rules $\mathcal{R}$, the basic position graph *of $\mathcal{R}$, denoted by $PG(\mathcal{R})$, is the disjoint union of $PG(R_i)$, for all $R_i \in \mathcal{R}$.*

An existential position $[a, i]$ is said to be *infinite* if there is an atomset F such that running the chase on F produces an unbounded number of instantiations of $\mathit{term}([a, i])$. To detect infinite positions, we encode how variables may be "propagated" among rules by adding edges to $PG(\mathcal{R})$, called *transition edges*, which go from positions in rule heads to positions in rule bodies. The set of transition edges has to be *correct*: if an existential position $[a, i]$ is infinite, there must be a cycle going through $[a, i]$ in the graph.

Definition 2 (PG^X) *Let $\mathcal{R}$ be a set of rules. The three following position graphs are obtained from $PG(\mathcal{R})$ by adding a (transition) edge from each position p_h in a rule head H_i to each position p_b in a rule body B_j, with the* same predicate*, provided that some condition is satisfied :*

- full PG*, denoted by $PG^F(\mathcal{R})$: no additional condition,*
- dependency PG *($PG^D(\mathcal{R})$) : if R_j depends on R_i,*
- PG with unifiers *($PG^U(\mathcal{R})$) : if there is a piece-unifier μ of B_j with H_i such that $\mu(\mathit{term}(p_h)) = \mu(\mathit{term}(p_b))$.*

All three position graphs are correct. Intuitively, $PG^F(\mathcal{R})$ corresponds to the case where all rules are supposed to depend on all rules; its set of cycles is in bijection with the set of cycles in the predicate position graph defining weak-acyclicity. $PG^D(\mathcal{R})$ encodes actual rule dependencies. Finally, $PG^U(\mathcal{R})$ adds information about the piece-unifiers themselves. This provides an accurate encoding of variable propagation from an atom position to another.

Proposition 1 (Inclusions between PG^X) *Let $\mathcal{R}$ be a set of rules. $PG^U(\mathcal{R}) \subseteq PG^D(\mathcal{R}) \subseteq PG^F(\mathcal{R})$. Furthermore, $PG^D(\mathcal{R}) = PG^F(\mathcal{R})$ if $GRD(\mathcal{R})$ is a complete graph.*

Example 4 (PG^F and PG^D) *Let $\mathcal{R} = \{R_1, R_2\}$ from Example 2. Figure 2 pictures $PG^F(\mathcal{R})$ and $PG^D(\mathcal{R})$. The dashed edges belong to $PG^F(\mathcal{R})$ but not to $PG^D(\mathcal{R})$. Indeed, R_2 does not depend on R_1. $PG^F(\mathcal{R})$ has a cycle while $PG^D(\mathcal{R})$ has not.*

Example 5 (PG^D and PG^U) *Let $\mathcal{R} = \{R_1, R_2\}$, with $R_1 = t(x, y) \to p(z, y), q(y)$ and $R_2 = p(u, v), q(u) \to t(v, w)$. In Figure 3, the dashed edges belong to $PG^D(\mathcal{R})$ but not to $PG^U(\mathcal{R})$. Indeed, the only piece-unifier of B_2 with H_1 unifies u and y. Hence, the cycle in $PG^D(\mathcal{R})$ disappears in $PG^U(\mathcal{R})$.*

We now study how acyclicity properties can be expressed on position graphs. The idea is to associate, with an acyclicity property, a function that assigns to each position a subset of positions reachable

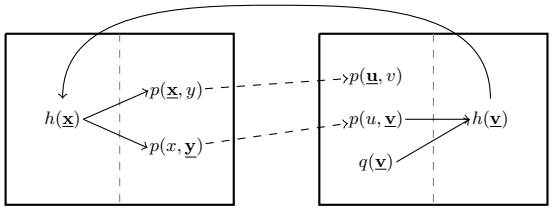

Figure 2. $PG^F(\mathcal{R})$ and $PG^D(\mathcal{R})$ from Example 4. Position $[a, i]$ is represented by underlining the i-th term in a. Dashed edges do not belong to $PG^D(\mathcal{R})$.

from this position, according to some propagation constraints; then, the property is fulfilled if no existential position can be reached from itself. More precisely, a *marking function* Y assigns to each node $[a, i]$ in a position graph PG^X, a subset of its (direct or indirect) successors, called its *marking*. A *marked cycle* for $[a, i]$ (w.r.t. X and Y) is a cycle C in PG^X such that $[a, i] \in C$ and for all $[a', i'] \in C$, $[a', i']$ belongs to the marking of $[a, i]$. Obviously, the less situations there are in which the marking may "propagate" in a position graph, the stronger the acyclicity property is.

Definition 3 (Acyclicity property) *Let Y be a marking function and PG^X be a position graph. The* acyclicity property *associated with Y in PG^X, denoted by Y^X, is satisfied if there is no marked cycle for an existential position in PG^X. If Y^X is satisfied, we also say that $PG^X(\mathcal{R})$* satisfies *Y.*

For instance, the marking function associated with weak-acyclicity assigns to each node the set of its successors in $PG^F(\mathcal{R})$, without any additional constraint. The next proposition states that such marking functions can be defined for each class of rules between wa and swa (first column in Figure 1), in such a way that the associated acyclicity property in PG^F characterizes this class.

Proposition 2 *A set of rules $\mathcal{R}$ is wa (resp. fd, ar, ja, swa) iff $PG^F(\mathcal{R})$ satisfies the acyclicity property associated with wa- (resp. fd-, ar-, ja-, swa-) marking.*

As already mentioned, all these classes can be safely extended by combining them with the GRD. To formalize this, we recall the notion $Y^<$ from [12]: given an acyclicity property Y, a set of rules $\mathcal{R}$ is said to satisfy $Y^<$ if each s.c.c. of $GRD(\mathcal{R})$ satisfies Y, except for those composed of a single rule with no loop.[4] Whether $\mathcal{R}$ satisfies $Y^<$ can be checked on $PG^D(\mathcal{R})$:

Proposition 3 *Let $\mathcal{R}$ be a set of rules, and Y be an acyclicity property. $\mathcal{R}$ satisfies $Y^<$ iff $PG^D(\mathcal{R})$ satisfies Y, i.e., $Y^< = Y^D$.*

[4] This particular case is to cover *aGRD*, in which each s.c.c. is an isolated node.

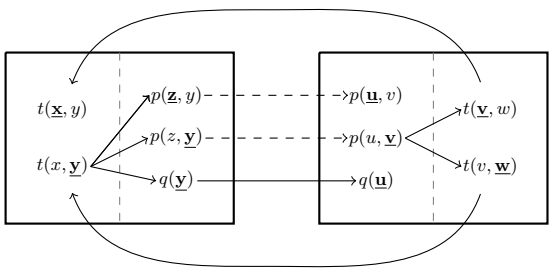

Figure 3. $PG^D(\mathcal{R})$ and $PG^U(\mathcal{R})$ from Example 5. Dashed edges do not belong to $PG^U(\mathcal{R})$.

For the sake of brevity, if Y_1 and Y_2 are two acyclicity properties, we write $Y_1 \subseteq Y_2$ if any set of rules satisfying Y_1 also satisfies Y_2. The following results are straightforward.

Proposition 4 *Let Y_1, Y_2 be two acyclicity properties. If $Y_1 \subseteq Y_2$, then $Y_1^D \subseteq Y_2^D$.*

Proposition 5 *Let Y be an acyclicity property. If $aGRD \not\subseteq Y$ then $Y \subset Y^D$.*

Hence, any class of rules satisfying a property Y^D strictly includes both $aGRD$ and the class characterized by Y; (e.g., Figure 1, from Column 1 to Column 2). More generally, strict inclusion in the first column leads to strict inclusion in the second one:

Proposition 6 *Let Y_1, Y_2 be two acyclicity properties such that $Y_1 \subset Y_2$, $wa \subseteq Y_1$ and $Y_2 \not\subseteq Y_1^D$. Then $Y_1^D \subset Y_2^D$.*

The next theorem states that PG^U is strictly more powerful than PG^D; moreover, the "jump" from Y^D to Y^U is at least as large as from Y to Y^D.

Theorem 1 *Let Y be an acyclicity property. If $Y \subset Y^D$ then $Y^D \subset Y^U$. Furthermore, there is an injective mapping from the sets of rules satisfying Y^D but not Y, to the sets of rules satisfying Y^U but not Y^D.*

Proof: Assume $Y \subset Y^D$ and $\mathcal{R}$ satisfies Y^D but not Y. $\mathcal{R}$ can be rewritten into $\mathcal{R}'$ by applying the following steps. First, for each rule $R_i = B_i[\vec{x}, \vec{y}] \to H_i[\vec{y}, \vec{z}] \in \mathcal{R}$, let $R_{i,1} = B_i[\vec{x}, \vec{y}] \to p_i(\vec{x}, \vec{y})$ where p_i is a fresh predicate; and $R_{i,2} = p_i(\vec{x}, \vec{y}) \to H_i[\vec{y}, \vec{z}]$. Then, for each rule $R_{i,1}$, let $R'_{i,1}$ be the rule $(B'_{i,1} \to H_{i,1})$ with $B'_{i,1} = B_{i,1} \cup \{p'_{j,i}(x_{j,i}) : \forall R_j \in \mathcal{R}\}$, where $p'_{j,i}$ are fresh predicates and $x_{j,i}$ fresh variables. Now, for each rule $R_{i,2}$, let $R'_{i,2}$ be the rule $(B_{i,2} \to H'_{i,2})$ with $H'_{i,2} = H_{i,2} \cup \{p'_{i,j}(z_{i,j}) : \forall R_j \in \mathcal{R}\}$, where $z_{i,j}$ are fresh existential variables. Let $\mathcal{R}' = \bigcup_{R_i \in \mathcal{R}} \{R'_{i,1}, R'_{i,2}\}$. This construction ensures that each $R'_{i,2}$ depends on $R'_{i,1}$, and each $R'_{i,1}$ depends on each $R'_{j,2}$, thus, there is a *transition* edge from each $R'_{i,1}$ to $R'_{i,2}$ and from each $R'_{j,2}$ to each $R'_{i,1}$. Hence, $PG^D(\mathcal{R}')$ contains exactly one cycle for each cycle in $PG^F(\mathcal{R})$. Furthermore, $PG^D(\mathcal{R}')$ contains at least one marked cycle w.r.t. Y, and then $\mathcal{R}'$ does not satisfy Y^D. Now, each cycle in $PG^U(\mathcal{R}')$ is also a cycle in $PG^D(\mathcal{R})$, and, since $PG^D(\mathcal{R})$ satisfies Y, $PG^U(\mathcal{R}')$ also does. Hence, $\mathcal{R}'$ does not belong to Y^D but to Y^U. □

We also check that strict inclusions in the second column in Figure 1 lead to strict inclusions in the third column.

Theorem 2 *Let Y_1 and Y_2 be two acyclicity properties. If $Y_1^D \subset Y_2^D$ then $Y_1^U \subset Y_2^U$.*

Proof: Let $\mathcal{R}$ be a set of rules such that $\mathcal{R}$ satisfies Y_2^D but does not satisfy Y_1^D. We rewrite $\mathcal{R}$ into $\mathcal{R}'$ by applying the following steps. For each pair of rules $R_i, R_j \in \mathcal{R}$ such that R_j depends on R_i, for each variable x in the frontier of R_j and each variable y in the head of R_i, if x and y occur both in a given predicate position, we add to the body of R_j a new atom $p_{i,j,x,y}(x)$ and to the head of R_i a new atom $p_{i,j,x,y}(y)$, where $p_{i,j,x,y}$ denotes a fresh predicate. This construction allows each term from the head of R_i to propagate to each term from the body of R_j, if they share some predicate position in $\mathcal{R}$. Thus, any cycle in $PG^D(\mathcal{R})$ is also in $PG^U(\mathcal{R}')$, without any change in the behavior w.r.t. the acyclicity properties. Hence $\mathcal{R}'$ satisfies Y_2^U but does not satisfy Y_1^U. □

The next result states that Y^U is a sufficient condition for chase termination:

Theorem 3 *Let Y be an acyclicity property ensuring the halting of some chase variant C. Then, the C-chase halts for any set of rules $\mathcal{R}$ that satisfies Y^U (hence Y^D).*

Example 6 *Consider again the set of rules $\mathcal{R}$ from Example 5. Figure 3 pictures the associated position graphs $PG^D(\mathcal{R})$ and $PG^U(\mathcal{R})$. $\mathcal{R}$ is not* aGRD*, nor* wa*, nor* wa^D *since $PG^D(\mathcal{R})$ contains a (marked) cycle that goes through the existential position $[t(v,w),2]$. However, $\mathcal{R}$ is obviously* wa^U *since $PG^U(\mathcal{R})$ is acyclic. Hence, the skolem chase and stronger chase variants halt for $\mathcal{R}$ and any set of facts.*

Finally, we remind that classes from wa to swa can be recognized in PTIME, and checking $aGRD$ is co-NP-complete. Hence, as stated by the next result, the expressiveness gain is without increasing worst-case complexity.

Theorem 4 (Complexity) *Let Y be an acyclicity property, and $\mathcal{R}$ be a set of rules. If checking that $\mathcal{R}$ satisfies Y is in co-NP, then checking that $\mathcal{R}$ satisfies Y^D or Y^U is co-NP-complete.*

5 FURTHER REFINEMENTS

In this section, we show how to further extend Y^U into Y^{U^+} by a finer analysis of marked cycles and unifiers. This extension can be performed without increasing complexity. We define the notion of *incompatible* sequence of unifiers, which ensures that a given sequence of rule applications is impossible. Briefly, a marked cycle for which all sequences of unifiers are incompatible can be ignored. Beside the gain for positive rules, this refinement will allow one to take better advantage of negation.

We first point out that the notion of piece-unifier is not appropriate to our purpose. We have to relax it, as illustrated by the next example. We call *unifier*, of a rule body B_2 with a rule head H_1, a substitution μ of $vars(B_2') \cup vars(H_1')$, where $B_2' \subseteq B_2$ and $H_1' \subseteq H_1$, such that $\mu(B_2') = \mu(H_1')$ (thus, it satisfies Condition (1) of a piece-unifier).

Example 7 *Let $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$ with:*
$R_1 : p(x_1, y_1) \rightarrow q(y_1, z_1)$
$R_2 : q(x_2, y_2) \rightarrow r(x_2, y_2)$
$R_3 : r(x_3, y_3) \wedge s(x_3, y_3) \rightarrow p(x_3, y_3)$
$R_4 : q(x_4, y_4) \rightarrow s(x_4, y_4)$
There is a dependency cycle (R_1, R_2, R_3, R_1) and a corresponding cycle in PG^U. We want to know if such a sequence of rule applications is possible. We build the following new rule, which is a composition of R_1 and R_2 (formally defined later): $R_1 \diamond_\mu R_2 : p(x_1, y_1) \rightarrow q(y_1, z_1) \wedge r(y_1, z_1)$
There is no piece-unifier of R_3 with $R_1 \diamond_\mu R_2$, since y_3 would be a separating variable mapped to the existential variable z_1. This actually means that R_3 is not applicable right after *$R_1 \diamond_\mu R_2$. However, the atom needed to apply $s(x_3, y_3)$ can be brought by a sequence of rule applications (R_1, R_4). We thus relax the notion of piece-unifier to take into account arbitrarily long sequences of rule applications.*

Definition 4 (Compatible unifier) *Let R_1 and R_2 be rules. A unifier μ of B_2 with H_1 is* compatible *if, for each position $[a, i]$ in B_2', such that $\mu(\mathsf{term}([a, i]))$ is an existential variable z in H_1', $PG^U(\mathcal{R})$ contains a path, from a position in which z occurs, to $[a, i]$, that does not go through another existential position. Otherwise, μ is* incompatible.

Note that a piece-unifier is necessarily compatible.

Proposition 7 *Let R_1 and R_2 be rules, and let μ be a unifier of B_2 with H_1. If μ is incompatible, then no application of R_2 can use an atom in $\mu(H_1)$.*

We define the rule corresponding to the composition of R_1 and R_2 according to a compatible unifier, then use this notion to define a compatible sequence of unifiers.

Definition 5 (Unified rule, Compatible sequence of unifiers)
• *Let R_1 and R_2 be rules such that there is a compatible unifier μ of B_2 with H_1. The associated* unified rule *$R_\mu = R_1 \diamond_\mu R_2$ is defined by $H_\mu = \mu(H_1) \cup \mu(H_2)$, and $B_\mu = \mu(B_1) \cup (\mu(B_2) \setminus \mu(H_1))$.*
• *Let $(R_1, \ldots, R_{k+1})$ be a sequence of rules. A sequence $s = (R_1\ \mu_1\ R_2 \ldots\ \mu_k\ R_{k+1})$, where, for $1 \leq i \leq k$, μ_i is a unifier of B_{i+1} with H_i, is a* compatible sequence *of unifiers if:* (1) *μ_1 is a compatible unifier of B_2 with H_1, and* (2) *if $k > 0$, the sequence obtained from s by replacing $(R_1\ \mu_1\ R_2)$ with $R_1 \diamond_{\mu_1} R_2$ is a* compatible *sequence of unifiers.*

E.g., in Example 7, the sequence $(R_1\ \mu_1\ R_2\ \mu_2\ R_3\ \mu_3\ R_1)$, with the obvious μ_i, is compatible. We can now improve all previous acyclicity properties (see the fourth column in Figure 1).

Definition 6 (Compatible cycles) *Let Y be an acyclicity property, and PG^U be a position graph with unifiers. The* compatible cycles for $[a, i]$ *in PG^U are all marked cycles C for $[a, i]$ w.r.t. Y, such that there is a compatible sequence of unifiers induced by C. Property Y^{U+} is satisfied if, for each existential position $[a, i]$, there is no compatible cycle for $[a, i]$ in PG^U.*

Results similar to Theorem 1 and Theorem 2 are obtained for Y^{U^+} w.r.t. Y^U, namely:

- For any acyclicity property Y, $Y^U \subset Y^{U+}$.
- For any acyclicity properties Y_1 and Y_2, if $Y_1^U \subset Y_2^U$, then $Y_1^{U+} \subset Y_2^{U+}$.

Moreover, Theorem 3 can be extended to Y^{U^+}: let Y be an acyclicity property ensuring the halting of some chase variant C; then the C-chase halts for any set of rules $\mathcal{R}$ that satisfies Y^{U^+} (hence Y^U). Finally, the complexity result from Theorem 4 still holds for this improvement.

6 EXTENSION TO NONMONOTONIC NEGATION

We now add nonmonotonic negation, which we denote by **not**. A *nonmonotonic existential* (NME) rule R is of the form $\forall\vec{x}\forall\vec{y}(B^+ \wedge \mathbf{not}B_1^- \wedge \ldots \wedge \mathbf{not}B_k^- \rightarrow \exists\vec{z}H)$, where B^+, $B^- = \{B_1^- \ldots B_k^-\}$ and H are atomsets, respectively called the *positive* body, the *negative* body and the head of R; furthermore, $vars(B^-) \subseteq vars(B^+)$. R is *applicable* to F if there is a homomorphism h from B^+ to F such that $h(B^-) \cap F = \emptyset$. In this section, we rely on a skolemization of the knowledge base. Then, the application of R to F w.r.t. h produces $h(sk(H))$. R is *self-blocking* if $B^- \cap (B^+ \cup H) \neq \emptyset$, i.e., R is never applicable.

Since skolemized NME rules can be seen as normal logic programs, we can rely on the standard definition of stable models [16], which we omit here since it is not needed to understand the sequel. Indeed, our acyclicity criteria essentially ensure that there is a finite number of skolemized rule applications. Although the usual definition of stable models relies on grounding (i.e., instantiating) skolemized rules, stable models of $(F, \mathcal{R})$ can be computed by a skolem

chase-like procedure, as performed by Answer Set Programming solvers that instantiate rules on the fly [21, 13].

We check that, when the skolem chase halts on the positive part of NME rules (i.e., obtained by ignoring the negative body), the stable computation based on the skolem chase halts. We can thus rely on preceding acyclicity conditions, which already generalize known acyclicity conditions applicable to skolemized NME rules (for instance *finite-domain* and *argument-restricted*, which were defined for normal logic programs). We can also extend them by exploiting negation.

First, we consider the natural extensions of a unified rule (Def. 5) and of rule dependency: to define $R_\mu = R_1 \diamond_\mu R_2$, we add that $B^-_\mu = \mu(B^-_1) \cup \mu(B^-_2)$; besides, R_2 depends on R_1 if there is a piece-unifier μ of H_2 with B_1 such that $R_1 \diamond_\mu R_2$ is not self-blocking; if $R_1 \diamond_\mu R_2$ is self-blocking, we say that μ is self-blocking. Note that this extended dependency is equivalent to the *positive reliance* from [23]. In this latter paper, positive reliance is used to define an acyclicity condition: a set of NME rules is said to be *R-acyclic* if no cycle of positive reliance involves a rule with an existential variable. Consider now PG^D with extended dependency: then, R-acyclicity is stronger than aGRD (since cycles are allowed on rules without existential variables) but weaker than wa^D (since all s.c.c. are necessarily wa).

By considering extended dependency, we can extend the results obtained with PG^D and PG^U (note that for PG^U we only encode non-self-blocking unifiers). We can further extend Y^{U+} classes by considering *self-blocking compatible sequences* of unifiers. Let C be a compatible cycle for $[a, i]$ in PG^U, and C_μ be the set of all compatible sequences of unifiers induced by C. A sequence $\mu_1 \dots \mu_k \in \mathcal{C}_\mu$ is said to be self-blocking if the rule $R_1 \diamond_{\mu_1} R_2 \dots R_k \diamond_{\mu_k} R_1$ is self-blocking. When all sequences in C_μ are self-blocking, C is said to be self-blocking.

Example 8 *Let* $R_1 = q(x_1), \mathbf{not}p(x_1) \to r(x_1, y_1)$, $R_2 = r(x_2, y_2) \to s(x_2, y_2)$, $R_3 = s(x_3, y_3) \to p(x_3), q(y_3)$. $PG^{U+}(\{R_1, R_2, R_3\})$ *has a unique cycle, with a unique induced compatible unifier sequence. The rule* $R_1 \diamond R_2 \diamond R_3 = q(x_1), \mathbf{not}p(x_1) \to r(x_1, y_1), s(x_1, y_1), p(x_1), q(y_1)$ *is self-blocking, hence* $R_1 \diamond R_2 \diamond R_3 \diamond R_1$ *also is. Thus, there is no "dangerous" cycle.*

Proposition 8 *If, for each existential position* $[a, i]$*, all compatible cycles for* $[a, i]$ *in* PG^U *are self-blocking, then the stable computation based on the skolem chase halts.*

Finally, we point out that these improvements do not increase worst-case complexity of the acyclicity test.

7 CONCLUSION

We have proposed a tool that allows to unify and generalize most existing acyclicity conditions for existential rules, without increasing worst-case complexity. This tool can be further refined to deal with nonmonotonic (skolemized) existential rules, which, to the best of our knowledge, extends all known acyclicity conditions for this kind of rules.

Further work includes the implementation of the tool[5] and experiments on real-world ontologies, as well as the study of chase variants that would allow to process existential rules with stable negation without skolemization.

[5] It will be developed as an extension of KIABORA, an analyzer of existential rule bases [20].

Acknowledgements. This work was partially supported by French *Agence Nationale de la Recherche* (ANR), under project grants ASPIQ (ANR-12-BS02-0003), Pagoda (ANR-12-JS02-0007) and Qualinca (ANR-12-CORD-0012).

REFERENCES

[1] F. Baader, S. Brandt, and C. Lutz, 'Pushing the EL envelope', in *IJCAI'05*, pp. 364–369, (2005).
[2] J.-F. Baget, 'Improving the forward chaining algorithm for conceptual graphs rules', in *KR'04*, pp. 407–414. AAAI Press, (2004).
[3] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat, 'Extending decidable cases for rules with existential variables', in *IJCAI'09*, pp. 677–682, (2009).
[4] 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).
[5] C. Beeri and M. Vardi, 'The implication problem for data dependencies', in *ICALP'81*, volume 115 of *LNCS*, pp. 73–85, (1981).
[6] C. Beeri and M.Y. Vardi, 'A proof procedure for data dependencies', *Journal of the ACM*, **31**(4), 718–741, (1984).
[7] A. Calì, G. Gottlob, and M. Kifer, 'Taming the infinite chase: Query answering under expressive relational constraints', in *KR'08*, pp. 70–80, (2008).
[8] A. Calì, G. Gottlob, and T. Lukasiewicz, 'A general datalog-based framework for tractable query answering over ontologies', in *PODS'09*, pp. 77–86, (2009).
[9] F. Calimeri, S. Cozza, G. Ianni, and N. Leone, 'Computable functions in asp: Theory and implementation', in *Logic Programming*, 407–424, (2008).
[10] 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).
[11] A. K. Chandra, H. R. Lewis, and J. A. Makowsky, 'Embedded implicational dependencies and their inference problem', in *STOC'81*, pp. 342–354. ACM, (1981).
[12] B. Cuenca Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang, 'Acyclicity notions for existential rules and their application to query answering in ontologies', *J. Art. Intell. Res.*, **47**, 741–808, (2013).
[13] M. Dao-Tran, T. Eiter, M. Fink, G. Weidinger, and A. Weinzierl, 'Omiga: an open minded grounding on-the-fly answer set solver', in *Logics in Artificial Intelligence*, 480–483, (2012).
[14] A. Deutsch, A. Nash, and J.B. Remmel, 'The chase revisited', in *PODS'08*, pp. 149–158, (2008).
[15] 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).
[16] M. Gelfond and V. Lifschitz, 'The stable model semantics for logic programming', in *ICLP/SLP*, pp. 1070–1080, (1988).
[17] A. Hernich, C. Kupke, T. Lukasiewicz, and G. Gottlob, 'Well-founded semantics for extended datalog and ontological reasoning', in *PODS*, pp. 225–236, (2013).
[18] M. Krötzsch and S. Rudolph, 'Extending decidable existential rules by joining acyclicity and guardedness', in *IJCAI'11*, pp. 963–968, (2011).
[19] B. Lamare, 'Optimisation de la notion de dépendance', Internship report, ENS Cachan and LIRMM/ INRIA, (Sept. 2012).
[20] M. Leclère, M.-L. Mugnier, and S. Rocher, 'Kiabora: An analyzer of existential rule bases', in *RR*, pp. 241–246, (2013).
[21] C. Lefèvre and P. Nicolas, 'A first order forward chaining approach for answer set computing', in *LPNMR*, 196–208, (2009).
[22] Y. Lierler and V. Lifschitz, 'One more decidable class of finitely ground programs', in *Logic Programming*, 489–493, (2009).
[23] D. Magka, M. Krötzsch, and I. Horrocks, 'Computing stable models for nonmonotonic existential rules', in *IJCAI*, (2013).
[24] D. Maier, A. O. Mendelzon, and Y. Sagiv, 'Testing implications of data dependencies', *ACM Trans. Database Syst.*, **4**(4), 455–469, (1979).
[25] B. Marnette, 'Generalized schema-mappings: from termination to tractability', in *PODS*, pp. 13–22, (2009).
[26] A. Onet, 'The chase procedure and its applications in data exchange', in *Data Exchange, Information, and Streams*, pp. 1–37, (2013).