Extending Decidable Cases for Rules with Existential Variables

Jean-François Baget
INRIA Sophia-Antipolis
LIRMM
baget@lirmm.fr

Michel Leclère
LIRMM
Univ. Montpellier - CNRS
leclere@lirmm.fr

Marie-Laure Mugnier
LIRMM
Univ. Montpellier - CNRS
mugnier@lirmm.fr

Éric Salvat
IMERIR
LIRMM
salvat@imerir.com

Abstract

In $\forall\exists$-rules, the conclusion may contain existentially quantified variables, which makes reasoning tasks (as deduction) non-decidable. These rules have the same logical form as TGD (tuple-generating dependencies) in databases and as conceptual graph rules. We extend known decidable cases by combining backward and forward chaining schemes, in association with a graph that captures exactly the notion of dependency between rules. Finally, we draw a map of known decidable cases, including an extension obtained by combining our approach with very recent results on TGD.

Version published at IJCAI'09 with minor typos corrections

1 Introduction

Rules have long been considered as an essential component of knowledge-based systems. In this paper, we deal with rules which extend the rules usually considered in logic programming or deductive databases: they are in the form $H \rightarrow C$, where H and C are conjunctions of atoms, respectively called the hypothesis and conclusion of the rule, and there might be variables in the conclusion which are *existentially* quantified. *E.g.* the rule $R = \mathit{Human}(x) \rightarrow \mathit{Parent}(y, x) \wedge \mathit{Human}(y)$ stands for the formula $\forall x(\mathit{Human}(x) \rightarrow \exists y(\mathit{Parent}(y, x) \wedge \mathit{Human}(y)))$. This kind of rule, that we call $\forall\exists$-rule, has the same logical form as a very general kind of dependencies studied in databases called tuple generating dependencies (TGD) [AHV95] and as conceptual graph (CG) rules [Sow84; BM02]. They can also be seen as an abstraction for ontological knowledge expressed in specific KR languages, *e.g.* the RDFS rules [Hay04], constraints in F-logic-Lite [CK06; CGK08], as well as some kinds of inclusions in description logics [BCM$^+$03; CGL$^+$07].

Let us consider knowledge bases (KB) composed of a set of facts (existentially closed conjunctions of atoms) and a set of $\forall\exists$-rules. Several fundamental problems on these KB are equivalent (and very simple reductions allow us to go from one problem to another), namely *fact deduction* (is a fact deducible from a KB?), *rule deduction* (is a rule deducible from a KB?) and *boolean conjunctive query answering* in the presence of incomplete knowledge (is a boolean conjunctive query deducible from a KB?). The database versions of these problems are *conjunctive query containment* w.r.t. a set of TGD, *TGD implication* and *boolean conjunctive query answering* under constraints expressed by TGD. In this paper, we have chosen *fact deduction* to represent this family. All results obtained regarding this problem can be immediately recast in terms of the other problems.

Deduction is not decidable ([Var84] for TGD, [BM02] for CG rules), which is due to the existential variables in the conclusion. Finding expressive decidable cases is thus extremely important. Until now, there were only decidable cases based on the *forward chaining* scheme (called the *chase* in databases). With the previous rule R, forward chaining does not halt: once R has been applied, it can be applied again indefinitely and each application produces a fact which is not equivalent to any of the previous ones. Most exhibited decidable classes of rules correspond to cases where forward chaining halts (with the notable exceptions of the results in [JK84; CGK08], cf. Sect. 5).

We propose a new approach, which allows us to extend the map of decidable cases. Briefly, we combine *backward* and *forward* chaining schemes in association with a graph that captures exactly the notion of *dependency* between rules; this graph is used to define a combination of both schemes preserving completeness and decidability. The key notion, in the definitions of our backward chaining mechanism and our dependency criteria, is that of a *piece*, which can be seen as a "unit of knowledge" brought by a rule application. It allows us to keep accounting for the complex structure of a rule conclusion induced by existential variables, by characterizing sets of atoms which should not be processed separately (*e.g.* in the above rule R, $\{\mathit{Parent}(y, x), \mathit{Human}(y)\}$ is a piece).

The main contributions of this paper are as follows:

- new decidable classes based on finite backward chaining (Sect. 3.2: Th. 2, Prop. 2 and 3);
- the effective characterization of the notion of dependency between rules, allowing one to build an optimal graph of rule dependencies (GRD) (Sect. 4.1: Th. 3);
- new decidable classes obtained by combining the structure of the GRD and classes of rules for which forward or backward chaining are finite (Sect. 4.2: Th. 4, 5, 6);
- a map of known decidable cases, ordered by inclusion,

including an extension obtained by combining our results with the results in [CGK08] (Sect. 5: Th. 7, 8).

2 Preliminaries

Basic definitions. We consider a first-order logical language with constants but no other function symbols. For space saving, we also do not consider equality in this short paper. Hence, an atom is of form $p(t_1 \dots t_k)$, where p is a predicate of arity k and the t_i are variables or constants. A *fact* is a set of atoms. A *rule* is a couple $R = (H, C)$ of facts where H is called the *hypothesis* of R and C its *conclusion*. We note that $\mathit{term}(X)$, $\mathit{var}(X)$ and $\mathit{const}(X)$ are respectively the sets of terms, variables and constants occurring in the atoms of a fact or a rule X. The *frontier* of a rule R (notation $\mathit{fr}(R)$) is the set of variables occurring both in H and C. Let $F^\wedge$ be the conjunction of atoms in a fact F; the logical translation of F is the existential closure of $F^\wedge$. A rule $R = (H, C)$ is logically translated into the formula $\forall h_1 \dots \forall h_k (H^\wedge \rightarrow (\exists c_1 \dots \exists c_q C^\wedge))$ where $\{h_1, \dots, h_k\} = \mathit{var}(H)$ and $\{c_1, \dots, c_q\} = \mathit{var}(C) \setminus \mathit{fr}(R)$. In the examples, we use the form $H^\wedge \rightarrow C^\wedge$, with the quantification of variables being implicit. A *knowledge base* (KB) is composed of a set of facts and a set of *rules*. As the union of two facts is a fact, we consider the set of facts as a single fact. We thus note $(F, \mathcal{R})$ a KB, where F is a fact and $\mathcal{R}$ is a set of rules. The same notation is used for facts (resp. rules) and their logical translation. We use the classical notions of semantic consequence, noted $\models$, and equivalence, noted $\equiv$.

We address the following deduction problem, called $\mathcal{FR}$-DEDUCTION: "given a KB $K = (F, \mathcal{R})$ and a fact Q, is Q deducible from K, *i.e.* does $\{F\} \cup \mathcal{R} \models Q$ hold?". As explained in the introduction, this problem can be seen as a representative of several equivalent problems. It is semi-decidable (several proofs mentioned in [BM02]). Since Q and F are both facts, only their role in the deduction problem distinguishes them. When we need to distinguish between them, we call F the *assertion* and Q the *query*; Q is also called a *goal* in backward chaining.

Substitution and homomorphism. Given a set of variables V and a set of terms T, a *substitution* of V by T is a mapping from V to T. Let $\sigma : V \rightarrow T$ be a substitution, and F be a fact. $\sigma(F)$ denotes the fact obtained from F by replacing each occurrence of $x \in V \cap \mathit{term}(F)$ by $\sigma(x)$. If $R = (H, C)$ is a rule, $\sigma(R) = (\sigma(H), \sigma(C))$. A *safe substitution* is a bijection from V to a set of new variables (*i.e.* that do not appear in the formulas involved in the reasoning). Given two facts F and Q, a *homomorphism* from Q to F is a substitution σ from $var(Q)$ to $term(F)$ such that $\sigma(Q) \subseteq F$. If there is a homomorphism from Q to F, we say that Q *maps to* F. Homomorphism checking is sound and complete w.r.t. logical deduction in the fragment of facts: given F and Q two facts, $F \models Q$ iff Q maps to F (several proofs of this result in KR and databases).

Forward Chaining. We assume in this paper that the reader is familiar with forward and backward chaining paradigms. We recall here some definitions and results about forward chaining. A rule $R = (H, C)$ is *applicable* to a fact F if there is a homomorphism σ from H to F. The *application of R to F according to σ* produces a fact $\alpha(F, R, \sigma) = F \cup \sigma(\sigma'(C))$, where σ' is a safe substitution of $\mathit{var}(C) \setminus \mathit{fr}(R)$. This application is said to be redundant if $\alpha(F, R, \sigma) \equiv F$ (it suffices to check that $\alpha(F, R, \sigma)$ maps to F). A fact F' is called an *$\mathcal{R}$-derivation* of F if there is a finite sequence (called the *derivation sequence*) $F = F_0, F_1, \dots, F_k = F'$ such that for all $1 \leq i \leq k$, there is a rule $R = (H, C) \in \mathcal{R}$ and a homomorphism σ from H to F_{i-1} with $F_i = \alpha(F_{i-1}, R, \sigma)$. Such a derivation yields a sound and complete mechanism: given a KB $K = (F, \mathcal{R})$ and a fact Q, Q is deducible from K iff Q maps to an $\mathcal{R}$-derivation of F [SM96]. The following definition of a *finite expansion set* and associated concrete cases are from [BM02]. A set of rules is called a finite expansion set if it is guaranteed, for any fact, that after a finite number of rule applications, all further rule applications will become redundant, *i.e.* produce facts equivalent to the current fact. More precisely:

Definition 1 (Finite expansion set and full derivation) *A set of rules $\mathcal{R}$ is called a* finite expansion set *(f.e.s.) if for any fact F, there is an $\mathcal{R}$-derivation F' of F such that for every rule $R = (H, C) \in \mathcal{R}$, for every homomorphism σ from H to F', $\alpha(F', R, \sigma)$ maps to F' (i.e. all rule applications on F' are redundant). F' is called a* full *$\mathcal{R}$-derivation of F.*

If $\mathcal{R}$ is a f.e.s., any forward chaining algorithm that builds a derivation sequence and stops when all rule applications are redundant then checks if Q maps to the fact obtained, is complete and halts in finite time. The f.e.s. notion is an abstract characterization of sets of rules allowing for a finite forward chaining mechanism, but it does not help to predict that the mechanism will actually stop on a given set of rules (note, moreover, that all decidable cases of the problem do not rely upon f.e.s.). As concrete classes of f.e.s., let us cite the *range restricted rules* (r.r.) in positive Datalog, which do not have existentially quantified variables (*i.e.* $\mathit{var}(C) \subseteq \mathit{var}(H)$) and *disconnected rules*, whose frontier is empty. The backward chaining paradigm that we present now allows one for new decidable classes, namely by *finite unification sets.*

3 Backward Chaining Revisited

The key operation in a backward chaining mechanism is the *unification* between part of a current goal (a fact in our framework) and a rule conclusion. This mechanism is typically used in logic programming, with rules having a single atom in the conclusion, which is unified with an atom of the current goal. Since the conclusion of a $\forall\exists$-rule has a more complex structure (it may contain several atoms and possibly existentially quantified variables), the associated unification operation is also more complex. It allows one to process conclusions and goals without decomposing them into atoms, and we will show that it is worthwhile to do so. We will rely on the notion of a *piece*, which stems from a graph vision of rules and was introduced in [SM96] for CG rules. We reformulate it, as well as the associated unification notion, in a logical framework. As shown in Sect. 4.1, we will also use unification in a new perspective.

3.1 Piece-based rewriting

Given a subset T of its terms, a fact can be partitioned into pieces according to T. The piece notion is easier to grasp if we see a fact as an undirected bipartite graph, with two kinds of nodes, representing respectively terms and atoms, and such that there is an edge between a term node t and an atom node A if t occurs in A. Then, given a set of term nodes T, two atom nodes A_1 and A_2 are in the same piece if there is a path between them that does not go *through* a node of T. If $T = \emptyset$, each connected component of T is a piece.

Definition 2 (Pieces, Cutpoints) *Let F be a fact and $T \subseteq$ term(F). A* piece *of F according to T is a non-empty subset of F recursively defined as follows: each atom is in a piece; two distinct atoms A_1 and A_2 are in the same piece iff either $(var(A_1) \cap var(A_2)) \setminus T \neq \emptyset$, or there is an atom $A_3 \in F$ such that A_1 and A_3 are in the same piece and A_3 and A_2 are in the same piece. Let $R = (H, C)$ be a rule. The* cutpoints *of R are the set of terms cutp$(R) =$ fr$(R) \cup$ const(C). A* piece *of R is a piece of C according to cutp(R).*

Example 1 (Pieces) $R = p(x, y) \rightarrow p(x, z) \wedge p(z, t) \wedge p(t, x) \wedge p(x, u)$ has one cutpoint, which is x, hence two pieces $\{p(x, z), p(z, t), p(t, x)\}$ and $\{p(x, u)\}$.

A piece can be seen as a "unit" of knowledge brought by a rule application in forward chaining. Indeed, a rule R can be decomposed into an equivalent set of rules with the same hypothesis and exactly one piece in the conclusion. More precisely, any rule $R = (H, C)$, such that C contains k pieces $C_1, \ldots, C_k$, is semantically equivalent to the set of rules $\{(H, C_i)\}_{1 \leq i \leq k}$. Moreover, the conclusions of these rules cannot be further decomposed while keeping a set of $\forall\exists$-rules with the same semantics as R (provided that H is not modified, see another decomposition later).

Backward chaining erases whole pieces of a goal Q. To explain the key ideas of the following unifier definition, let us present it as performing the inverse of a rule application to a potential fact. Given Q and a rule $R = (H, C)$, assume that Q can be proven by an application of R to a fact F according to a homomorphism π_R, *i.e.* there is a homomorphism π_Q from Q to $\alpha(F, R, \pi_R)$ and $\pi_Q(Q)$ is not included in F. Let σ_R be the substitution of fr(R) (extracted from π_R) used to apply R to F. Q can then be partitioned into Q' and Q'', such that π_Q maps Q' to $\sigma_R(C)$ and Q'' to F. Let T_Q be the set of terms t in Q such that $\pi_Q(t)$ is in const$(C) \cup \sigma_R($fr$(R))$. T_Q defines pieces of Q, which can be partitioned into pieces of Q' and pieces of Q''. Each piece of Q' is mapped by π_Q to a piece of $\sigma_R(C)$. Briefly said, the backward chaining step associated with σ_R erases from Q the pieces composing Q' and adds $\sigma_R(H)$ to it.

Definition 3 (Unifier) *Let Q be a fact and $R = (H, C)$ be a rule. A* unifier *of Q with R is a tuple $\mu = (T_Q, Q', \sigma_Q, \sigma_R)$ where:*

- *T_Q is a subset of term(Q), which thus defines pieces in Q (T_Q is possibly empty);*
- *Q' is the union of one or more pieces of Q (defined by T_Q);*
- *σ_Q is a substitution from the variables of T_Q to const$(C) \cup T_Q$ and σ_R is a substitution from fr(R) to const$(C) \cup T_Q$;*
- *there is a homomorphism σ from $\sigma_Q(Q')$ to $\sigma_R(C)$ such that for all $t \in T_Q$, there is $t' \in$ cutp(R) with $\sigma(\sigma_Q(t)) = \sigma_R(t')$.*

Definition 4 (Rewriting a fact) *Let Q be a fact, $R = (H, C)$ be a rule, and $\mu = (T_Q, Q', \sigma_Q, \sigma_R)$ be a unifier of Q with R. A* rewriting of Q according to R and μ *produces a fact $\beta(Q, R, \mu) = \sigma'(\sigma_R(H)) \cup \sigma_Q(Q \setminus Q')$, where σ' is a safe substitution from var$(H) \setminus$ fr(R).*

Example 2 (Unification) Let $R = h(x, y) \rightarrow p(x, z) \wedge q(z, y) \wedge r(z, t)$. Let $Q = p(u, v) \wedge q(v, u) \wedge s(u, w)$. Q is unifiable with R, with $T_Q = \{u\}$, which defines two pieces $Q_1 = \{p(u, v), q(v, u)\}$ and $Q_2 = \{s(u, w)\}$. $Q' = Q_1$, $\sigma_Q = \{(u, u)\}$, $\sigma_R = \{(x, u), (y, u)\}$ (with $\sigma = \{(u, u), (v, z)\}$). The new fact is $h(u, u) \wedge s(u, w)$ (with $\sigma' = \emptyset$).

Definition 5 (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*) $Q = Q_0, Q_1, \ldots, Q_k = Q'$ such that for all $1 \leq i \leq k$, there is a rule $R = (H, C) \in \mathcal{R}$ and a unifier μ of Q_{i-1} with R such that $Q_i = \beta(Q_{i-1}, R, \mu)$.*

The soundness and completeness of backward chaining (next theorem) relies on the following equivalence between $\mathcal{R}$-rewriting and $\mathcal{R}$-derivation:

Property 1 *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 F' of F such that Q maps to F'.*

Theorem 1 *Let $K = (F, \mathcal{R})$ be a KB, and Q be a fact. Then $F, \mathcal{R} \models Q$ iff there is an $\mathcal{R}$-rewriting of Q that maps to F.*

Let us end with a remark about the piece notion. It might be argued that the unification operation would be much simpler if rule conclusions were decomposed, not only into single pieces (which would not fundamentally simplify the unification definition), but into single atoms. Indeed, a rule (H, C) can be equivalently encoded by a set of rules $\{H \rightarrow R(t_1, ..., t_k), (R(t_1, ..., tk) \rightarrow A_c)_{A_c \in C)}\}$, where R is a new predicate assigned to the rule and $t_1 ... t_k$ are the terms of C. However, the rewriting mechanism would then build "no-good" unifications that would have been avoided with piece-based unification (see the example 3 below). Besides the loss of efficiency in backward chaining, this decomposition of rule conclusions beyond single pieces weakens the characterization of decidable cases, as shown in Sect. 4.1.

Example 3 (Atomic Decomposition) The rule $R = h(x, y) \rightarrow p(x, z) \wedge q(z, y)$, which has a single piece, could be replaced by three rules: $R_1^A = h(x, y) \rightarrow R(x, y, z)$, $R_2^A = R(x, y, z) \rightarrow p(x, z)$ and $R_3^A = R(x, y, z) \rightarrow q(z, y)$. Let $Q = s(u, v) \wedge q(v, w)$. Q is not unifiable with R, but it is with R_3^A, because the information that R_2^A and R_3^A cannot be considered independently has been lost.

3.2 Finite unification sets

Consider a *backward chaining mechanism* that builds $\mathcal{R}$-rewritings of Q in a breadth-first way and maintains a set $\mathcal{Q}$ of the most general $\mathcal{R}$-rewritings built (*i.e.* it does not add a new $\mathcal{R}$-rewriting Q'' to $\mathcal{Q}$ if there is $Q' \in \mathcal{Q}$ with $Q'' \models Q'$); it answers yes if it finds an $\mathcal{R}$-rewriting Q' s.t. $F \models Q'$. This algorithm is sound and complete and halts on positive instances of the problem. Whereas finite expansion sets ensure that all information deducible from a fact in forward chaining can be encoded in a finite fact, the *finite unification sets* presented hereafter ensure that the above set $\mathcal{Q}$ of rewritings is finite.

Definition 6 (Finite unification set) *A set of rules $\mathcal{R}$ is called a finite unification set (f.u.s.) if for every fact Q, there is a finite set $\mathcal{Q}$ of $\mathcal{R}$-rewritings of Q such that for any $Q' \in \mathcal{Q}$, for any rule $R \in \mathcal{R}$, for any unifier μ of Q' with R, there is a fact in $\mathcal{Q}$ that maps to some $\beta(Q', R, \mu)$. We say that $\mathcal{Q}$ is a full $\mathcal{R}$-rewriting set of Q.*

The above backward chaining mechanism halts in finite time if $\mathcal{R}$ is a f.u.s., hence:

Theorem 2 *$\mathcal{FR}$-*DEDUCTION *is decidable if $\mathcal{R}$ is a f.u.s.*

Similarly to f.e.s., f.u.s. yield an abstract characterization, that should be instantiated with concrete examples. We provide two concrete cases of f.u.s. rules hereafter.

Definition 7 (Atomic hypothesis rule) *A rule $R = (H, C)$ is called an* atomic hypothesis rule *(a.h.) if H contains a single atom.*

Property 2 *A set of a.h. rules is a f.u.s.*

A.h. rules are particularly well adapted to express necessary properties of concepts or relations in ontological languages, without any restriction on the form of the conclusion (*i.e.* rules of form $C(x) \rightarrow P$ or $r(x_1 \dots x_k) \rightarrow P$, where C is a concept, r a k-ary relation and P any set of atoms). The second kind of rules does not put any restriction on the form of the hypothesis but constrains the form of the conclusion:

Definition 8 (Domain restricted rule) *A rule $R = (H, C)$ is called a domain restricted rule (d.r.) if each atom of C contains all or none of the variables in H.*

Property 3 *A set of d.r. rules is a f.u.s.*

E.g. the set $\{R\}$ with $R = \mathit{Human}(x) \rightarrow \mathit{Parent}(y, x) \wedge \mathit{Human}(y)$ is not a f.e.s. but it is a f.u.s. since R is a.h. (and d.r.). It is easy to check that the union of two f.e.s. (resp. f.u.s.) is not a f.e.s. (resp. f.u.s.). Moreover, combining a f.e.s. and a f.u.s. may lead to non-decidability. In the next section, we study conditions on the interaction between rules that keep decidability.

4 Compilation of the rule base

Generally speaking, compiling a knowledge base involves preprocessing it off-line, so that the compiled form obtained can be used on-line to accelerate reasoning tasks (*e.g.* query answering). Concerning rules, a classical compilation technique consists of precomputing a graph encoding dependencies between rules. Due to space limitations, we will not detail how this technique allows one to improve the efficiency of forward and backward chainings (as done for example in [Bag04]), but rather use it to extend decidable classes of rules.

4.1 The Optimal Graph of Rule Dependencies

A rule R' is said to *depend* on a rule R if the application of R on a fact may trigger a *new* application of R':

Definition 9 (Dependency) *A rule $R' = (H', C')$* depends *on a rule $R = (H, C)$ if there is a fact F, a homomorphism $\sigma : H \rightarrow F$ and a homomorphism $\sigma' : H' \rightarrow \alpha(F, R, \sigma)$, such that $\sigma'(H') \not\subseteq F$.*

It is easy to define necessary conditions for a rule to depend on another: *e.g.* if R' depends on R then there is an atom in H' that can be unified (classical meaning) with an atom in C. Characterizing dependency by actually computable necessary and sufficient conditions is less obvious. In the next definition of the graph of rule dependencies, we thus distinguish between graphs that are "complete" w.r.t. to dependency (*i.e.* all dependencies are encoded by their edges), and the only sound and complete graph, whose edges encode all dependencies and only dependencies.

Definition 10 (Graph of rule dependencies) *Let $\mathcal{R}$ be a set of rules. A* graph of rule dependencies *(GRD) of $\mathcal{R}$ is a directed graph $(\mathcal{R}, E)$, where $\mathcal{R}$ is the set of nodes and E is the set of edges, such that, if R' depends on R, then (R, R') is an edge of E. The* optimal *GRD of $\mathcal{R}$ (notation GRD($\mathcal{R}$)) is the (only) GRD of $\mathcal{R}$ with the minimal number of edges.*

Assertions (resp. queries) can be added to the GRD as rules with an empty hypothesis (resp. conclusion) and are thus sources (resp. sinks) in the GRD. The next theorem shows that our unifiers exactly capture the dependency notion:

Theorem 3 *Let $\mathcal{R}$ be a set of rules. Then the GRD $(\mathcal{R}, E)$, where $(R, R') \in E$ iff there is a unifier of the hypothesis of R' with R, is the optimal GRD of $\mathcal{R}$.*

Though all results presented hereafter in this paper remain valid when considering any GRD, building the optimal GRD is important both to optimize reasoning and provide wider decidable classes.

Example 4 (Dependency) Let $R_0 = p(x, y) \wedge p(y, x) \rightarrow p(x, z) \wedge p(z, t) \wedge p(t, x)$. With a weak criterium, R_0 could depend on itself. Using the criterium in Th. 3, it does not: C_0 is a single piece, and since R_0 has only one cutpoint (x), there should be a homomorphism from H_0 to C_0 (which would map x to x). Note that if z was replaced by y in C_0, the rule obtained would have 2 cutpoints and 2 pieces, and would depend on itself.

Example 5 (GRD) Let $\mathcal{R} = \{R_0, R_1, R_2, R_3\}$, where:
R_0 is the rule in Example 4
$R_1 = q(x) \wedge p(x, y) \rightarrow q(y)$
$R_2 = p(x, y) \rightarrow r(x, y, z) \wedge p(z, w)$
$R_3 = s(x) \wedge t(x, y) \rightarrow p(x, y)$
GRD($\mathcal{R}$) has two loops (R_1, R_1) and (R_2, R_2) plus the edges (R_0, R_1), (R_0, R_2), (R_3, R_0), (R_3, R_1) and (R_3, R_2).

Let us point out that decomposing rule conclusions into single atoms (see Sect. 3.1) would weaken the GRD notion. Indeed, "fake" dependencies could be introduced. For instance, in Example 3 and inserting Q (as a rule with an empty conclusion): there are edges (R_1^A, R_2^A) and (R_1^A, R_3^A), and, which is the bad point, an edge (R_3^A, Q), whereas there would be no edge (R, Q). Thus, even if the GRD obtained is

optimal for $\{R_1^A, R_2^A, R_3^A, Q\}$, it does not compute optimal dependencies w.r.t. $\{R, Q\}$.

4.2 Decidability results related to the GRD

Let us consider a basic forward chaining mechanism, that proceeds in a breadth-first way, *i.e.* at each step it computes all new (and non redundant) rule applications w.r.t. the current fact, then applies them to produce a new fact. If a subset of rules $X \subseteq \mathcal{R}$ has been non-redundantly applied at step i, then the only rules that have to be checked for applicability at step $i+1$ are in the set $\{R' \subseteq \mathcal{R} | \exists R \in X, (R, R') \in E\}$. Similar arguments apply for backward chaining. The next theorem follows:

Theorem 4 *Let $\mathcal{R}$ be a set of rules. If GRD($\mathcal{R}$) has no circuit, then $\mathcal{R}$ is both a f.e.s. and a f.u.s.*

Note that a loop (a self-unifiable rule) in the GRD is sufficient to yield the non-decidability of the deduction problem.[1] One can however accept some kinds of circuits, as stated in the next theorem:

Theorem 5 *Let $\mathcal{R}$ be a set of rules. If all strongly connected components[2] of GRD($\mathcal{R}$) are f.e.s. (resp. f.u.s.), then $\mathcal{R}$ is a f.e.s. (resp. f.u.s.).*

Since forward and backward chainings do not stop on the same kinds of rules, the idea is to combine both mechanisms, not into a single mechanism (like some Datalog evaluation techniques [AHV95]), but rather use one after the other, provided that the interactions between rules obey some constraint preserving finiteness, thus decidability. This constraint is expressed on the GRD as follows:

Definition 11 (Finitely combined set) *Let $\mathcal{R}$ be a set of rules. A partition $(\mathcal{R}_1, \mathcal{R}_2)$ of $\mathcal{R}$ is said to be* finitely combined *if $\mathcal{R}_1$ is a f.e.s., $\mathcal{R}_2$ is a f.u.s., and there is no edge from a rule of $\mathcal{R}_2$ to a rule of $\mathcal{R}_1$ in GRD($\mathcal{R}$). $\mathcal{R}$ is a* finitely combined set *if it admits a finitely combined partition.*

Example 5 *(continued)* Each rule forms its own connected component. $\{R_1\}$ is a f.e.s. since R_1 is range restricted (r.r.), but it is not a f.u.s. $\{R_2\}$ is a f.u.s. since R_2 is a.h. (and d.r.), but it is not a f.e.s. $\{R_3\}$ is a f.e.s. and a f.u.s. because R_3 is r.r. and d.r. R_0 does not belong to one of our concrete classes, but $\{R_0\}$ is a f.e.s. and a f.u.s. since its *GRD* has no edge. From Th. 5, we conclude that $\{R_0, R_1, R_3\}$ is a f.e.s. and $\{R_0, R_2, R_3\}$ is a f.u.s. A finitely combined partition necessarily considers R_3 and R_0 as f.e.s. because of their edges to R_1. Thus, the only finitely combined partition is $\{\{R_0, R_1, R_3\}, \{R_2\}\}$.

The next theorem provides an effective decidable sound and complete mechanism for finitely combined sets of rules: given a finitely combined partition, first use forward chaining on the f.e.s. part to compute a full derivation of the facts, say F', then use backward chaining on the f.u.s part to check if there is a rewriting of Q that maps to F'.

[1] In Baget's PhD thesis [Bag01], it is proven that a universal Turing machine can be encoded by a single self-dependent rule.

[2] Two nodes x and y are in the same strongly connected component if there are directed paths from x to y and from y to x. Any isolated node forms its own strongly connected component.

Theorem 6 *Let $(\mathcal{R}_1, \mathcal{R}_2)$ be a finitely combined partition of a set of rules $\mathcal{R}$, and F and Q be two facts. Then $F, \mathcal{R} \models Q$ iff Q' maps to F', where F' is a full $\mathcal{R}_1$-derivation of F and Q' belongs to a full $\mathcal{R}_2$-rewriting set of Q.*

5 Novelty, Related works and Perspectives

Weak characterizations of rule "dependency" (also called "precedence") have been used in several algorithms involving $\forall\exists$-rules, yielding non-optimal GRDs (*e.g.* in the Sesame reasoner which processes RDFS rules [BK03]). A precursor of our GRD is introduced in [Bag04]: this paper defines a criterium of dependency for CG rules, which, translated into $\forall\exists$-rules, is optimal for rules without constants. Instead, we rely on unification, which has an additional advantage of connecting the notion of dependency to backward chaining. F.e.s. are introduced in [BM02] and a result equivalent to the f.e.s. part in Th. 5 is proven in [Bag04]. Our definition of unification is equivalent (but more compact and simpler) to that introduced in [SM96] for processing CG rules in backward chaining. The notions of f.u.s. and associated results (Th. 2, Prop. 2 and 3) as well as the combined use of the GRD structure with forward and backward chainings are totally new (Th. 4, 5, 6).

Concerning TGD, let us first cite the pioneer paper of [JK84] that showed that query containment on inclusion dependencies (ID) is decidable even if the forward chaining (*i.e.* the chase) may not halt. Note that, as ID are rules with a single atom in the hypothesis and in the conclusion, they are particular a.h. rules, thus we provide another (and simpler) proof of decidability, based on the finiteness of backward chaining.

Let us now consider our results w.r.t. two very recent papers on TGD, [CGK08] and [DNR08]. [CGK08] presents decidability results going beyond f.e.s.: roughly summarized, it is shown that if a set of rules only generates facts having *bounded treewidth* when seen as graphs (we call such a set a bounded treewidth set, b.t.s. in short), the deduction problem becomes decidable. Intuitively, a b.t.s. generates a "tree-like" graph, and even though this graph can be infinite, to answer a query of finite size, it is not needed to generate more than a finite number of different "tree patterns". A f.e.s. is a b.t.s., since the finite saturated graph generated by a f.e.s. has bounded treewidth. On the other hand, a f.u.s. is not a b.t.s. Two concrete decidable fragments are defined in [CGK08]. *Guarded TGD* (gTGD) are rules such that an atom of the hypothesis contains ("guards") all variables of the hypothesis; in *weakly guarded TGD* (wgTGD), only some variables of the hypothesis need to be guarded[3]. gTGD are a subset of wgTGD, and both are b.t.s. The following examples illustrate that the decidable classes in both papers are incomparable.

1. $\{g(x,y,z,t) \wedge p(x,y) \wedge p(y,z) \wedge p(z,t) \rightarrow p(y,t)\}$ contains a single rule that is both guarded (gTGD) and r.r., but it is not a f.u.s.
2. $\{p(x,y) \rightarrow p(y,z)\}$ contains a single ID (thus both guarded and a.h.), but it is not a f.e.s.

[3] More precisely, these variables occur at an "affected" position in a predicate, *i.e.* a position in a predicate that may contain a new variable generated by forward chaining.

3. $\{t(x) \rightarrow s(x,y) \wedge t(y)\ ;\ t(x) \wedge t(y) \rightarrow r(x,y)\}$ is a f.u.s. (indeed, its GRD satisfies the conditions of Th. 5), but it is not a b.t.s.
4. $\{t(v,x) \rightarrow q(x,y) \wedge p(y,z) \wedge r(z,u)\ ;\ r(x,y) \wedge p(y,z) \rightarrow s(z,u) \wedge s(x,v)\}$ is both a f.e.s. and a f.u.s. (its GRD has no circuit), but it is not a set of wgTGD.

[DNR08] studies "sets of TGD with universal models", which are exactly f.e.s. The main decidable case is the "stratified chase graph", which can be seen as an instantiation of the f.e.s. part of Th. 5 with "weakly acyclic" (w.a.) TGD, this kind of TGD including the r.r. and disconnected rules of [BM02; Bag04]. The "chase graph" is equivalent to our GRD, but no constructive characterization of this graph is provided in [DNR08].

Since f.u.s. and b.t.s. form distinct decidable classes, one should explore how these sets interact with each other. The following theorems provide a first answer:

Theorem 7 *Let $\mathcal{R}$ be a set of rules. If all strongly connected components of GRD($\mathcal{R}$) are b.t.s., then $\mathcal{R}$ is a b.t.s.*

Since a single rule without self-dependency is a b.t.s., it follows that the latter example (point 4), though not a wgTGD, is a b.t.s.

Theorem 8 *Let $\mathcal{R}$ be a set of rules admitting a partition $(\mathcal{R}_1, \mathcal{R}_2)$ such that $\mathcal{R}_1$ is a b.t.s., $\mathcal{R}_2$ is a f.u.s. and there is no edge in GRD($\mathcal{R}$) from a rule of $\mathcal{R}_2$ to a rule of $\mathcal{R}_1$. Then the deduction problem is decidable.*

Figure 1 synthesizes the inclusions between the decidable classes finally obtained. All these inclusions are *strict*. Decidable classes written in italics are abstract classes. Classes in boldface are generic classes: we provide an algorithm recognizing them, and the size of the recognized class is function of the number and the size of concrete classes (in standard font) considered as their input. Note that we have added a new concrete class: a set of rules with a frontier of size at most one (noted fr1), which is interesting because it is not a f.e.s., neither a f.u.s., but is a b.t.s. without being necessarily weakly guarded. We have highlighted (in gray) classes that form the contributions of this paper. Even if the notion of acyclic GRD is not new in itself, we have highlighted it because the optimal characterization of dependencies (Th.3) is a key contribution of this paper. As for further work, our study of decidable cases will be continued through a study of their complexity, pursuing the work in [BM02], [CGK08] and [DNR08].

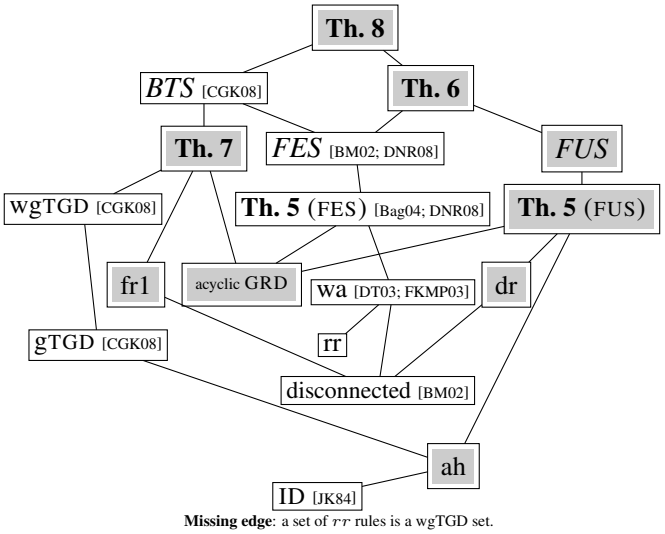

Figure 1: A map of decidable cases

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Bag01] J.-F. Baget. *Représenter des connaissances et raisonner avec des hypergraphes: de la projection à la dérivation sous contraintes*. PhD thesis, Université Montpellier II, Nov. 2001.

[Bag04] J.-F. Baget. Improving the forward chaining algorithm for conceptual graphs rules. In *KR*, pages 407–414. AAAI Press, 2004.

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[BK03] J. Broekstra and A. Kampman. Inferencing and truth maintenance in rdf schema: exploring a naive practical approach. In *PSSS*, 2003.

[BM02] J.-F. Baget and M.-L. Mugnier. The Complexity of Rules and Constraints. *JAIR*, 16:425–465, 2002.

[CGK08] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *KR*, 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.

[CK06] A. Calì and M. Kifer. Containment of conjunctive object meta-queries. In *VLDB*, pages 942–952, 2006.

[DNR08] A. Deutsch, A. Nash, and J.B. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.

[DT03] A. Deutsch and V. Tannen. Reformulation of xml queries and constraints. In *ICDT*, pages 225–241, 2003.

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[Hay04] P. Hayes, editor. *RDF Semantics*. W3C Recommendation. W3C, 2004.

[JK84] D.S. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28:167189, 1984.

[SM96] E. Salvat and M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *ICCS'96, LNAI 1115*, pages 248–262. Springer, 1996.

[Sow84] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Var84] M.Y. Vardi. The implication and finite implication problems for typed template dependencies. *JCSS*, 28(1):3–28, 1984.