Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)

Walking the Decidability Line for Rules with Existential Variables

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

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

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

Abstract

We consider positive rules in which the conclusion may contain existentially quantified variables, which makes reasoning tasks (such as Deduction) undecidable. These rules, called $\forall\exists$-rules, have the same logical form as TGD (tuple-generating dependencies) in databases and as conceptual graph rules. The aim of this paper is to provide a clearer picture of the frontier between decidability and non-decidability of reasoning with these rules. We show that Deduction remains undecidable with a single $\forall\exists$-rule; then we show that none of the known abstract decidable classes is recognizable. Turning our attention to concrete decidable classes, we provide new classes and classify all known classes by inclusion. Finally, we study, in a systematic way, the question "given two decidable sets of $\forall\exists$-rules, is their union decidable?", and provide an answer for all known decidable cases except one.

Introduction

Rules are fundamental constructs in knowledge-based systems and databases. Here we consider positive rules in first-order logic without functions, of 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 *existentially* quantified variables in the conclusion. *E.g.* the rule $R = \mathit{Human}(x) \rightarrow \mathit{Parent}(y, x) \wedge \mathit{Human}(y)$ stands for the formula $\forall x(\mathit{Human}(x) \rightarrow \exists y(\mathit{Parent}(y, x) \wedge \mathit{Human}(y)))$.

These rules, that we call $\forall\exists$-rules (BLMS09), correspond to a very general kind of integrity constraints in databases, i.e. so-called tuple generating dependencies (TGD) (AHV95). They are also equivalent to conceptual graph rules (Sow84; BM02). They generalize various kinds of constructs used to represent implicit knowledge, such as rules in deductive databases, or ontological knowledge such as rules expressing RDFS semantics (Hay04), constraints in F-logic-Lite (CK06; CGK08) and some kinds of inclusions in description logics (DLs) (BCM$^+$03; BLMS08; CGL09).

Variables existentially quantified in the conclusion, associated with arbitrary complex conjunctions of atoms, make $\forall\exists$-rules very expressive but also lead to undecidability of reasoning. Several decidable classes have been exhibited, in both artificial intelligence and databases. The aim of this paper is to bring a clearer picture of the frontier between decidability and undecidability of reasoning.

An important task on knowledge bases (KB) consists of querying them with queries at least equivalent to the fundamental queries in databases, i.e. conjunctive queries. We consider knowledge bases composed of a set of facts, which are existentially closed conjunctions of atoms, and a set of $\forall\exists$-rules. The problem of deciding whether the KB provides an answer to a conjunctive query is the same as deciding whether a fact is deducible from the KB. This is the basic problem we consider, and we simply call it DEDUCTION. Equivalent problems include *rule deduction* (is a $\forall\exists$-rule deducible from a KB?) as well as several fundamental problems in databases: *conjunctive query containment* w.r.t. a set of TGD, *TGD implication* and *boolean conjunctive query answering* under constraints expressed by TGD.

Contributions. The main contributions of this paper are as follows.

1. After having recalled that very strong restrictions on the set of predicates or the structure of $\forall\exists$-rules still maintain undecidability, we complete this gloomy picture by showing that DEDUCTION remains undecidable even with a *single* rule (Th. 1). This result has an important immediate consequence: adding a single rule to *any* set belonging to a decidable class of $\forall\exists$-rules can make the problem undecidable.

2. Even if the deduction problem is quickly undecidable, decidable classes have long been defined and used. Decidable classes found in the literature are based on various syntactic properties of $\forall\exists$-rules. We begin our study of decidable cases by considering abstract characterizations of these classes based on the behavior of reasoning mechanisms instead of syntactic properties. This yields three *abstract classes*. Two of them are based on a forward chaining scheme: *finite expansion sets* (BM02), ensuring that a finite number of rule applications is sufficient to answer any query, and the more general *bounded treewidth sets* (*bts*), inspired by the work of (CGK08), that relies on the finite treewidth model property of (Cou90). The third class is based on a backward chaining scheme: a *finite unification set* (BLMS09) ensures that any query can be finitely rewritten. We prove another negative result: these abstract classes are not recognizable, i.e. checking whether a given set of

$\forall\exists$-rules belongs to one of these classes is undecidable (Th. 3).

3. Since abstract classes are not recognizable, we turn our attention to *concrete classes* implementing their abstract behavior. These classes are defined by syntactic properties of $\forall\exists$-rules. They are less expressive but recognizable. We present a state of the art of known concrete decidable classes, classified by inclusion, and introduce two new decidable classes implementing the *bts* behavior: *frontier-guarded* rules and their extension to *weakly frontier-guarded* sets of rules. These classes are generalizations of the classes defined in (CGK08) and have the advantage of unifying some other known classes (BLMS09). We point out that their expressive power allows us to represent a set of description logic statements that are particularly interesting in the context of the new DLs tailored for conjunctive query answering. To show that (weakly) frontier-guarded rules have the *bts* property, we introduce a simple tool, i.e. the *Derivation Graph* (DG), as well as reduction operations on this graph. The fundamental property of this graph is as follows: if every DG produced by a set of rules can be reduced to a tree, then this set of rules has the *bts* property, which is especially the case for (weakly) frontier-guarded rules (Th. 5).

4. Having a range of non-comparable concrete decidable classes at our disposal, an interesting question is whether the union of two decidable classes remains decidable. This question is of utmost importance if we want to merge two ontologies for which decidability of reasoning is ensured by different syntactic properties, or if, having implemented the semantics of two KR languages with sets belonging to decidable classes, we want to consider the language built from the union of both languages. We present a systematic study of this question for all decidable classes we are aware of. With the exceptions of *disconnected rules*, which are universally compatible, and of a still open case, we show that the union of two incomparable decidable classes is never decidable (Th. 8 and Th. 9). These rather negative results on the rough union of decidable cases highlight the interest of precisely studying interactions between rules. We outline existing and future works in this direction.

Preliminaries

Facts and rules. We consider first-order logical languages with constants but no other function symbols. A *vocabulary* $\mathcal{V}$ is composed of a set of predicates and a set of constants. Hence, an atom on $\mathcal{V}$ is of form $p(t_1 \dots t_k)$, where p is a predicate of arity k in $\mathcal{V}$ and the t_i are constants in $\mathcal{V}$ or variables. For a formula ϕ, we note $terms(\phi)$ and $vars(\phi)$ resp. the terms and variables occurring in ϕ. We use the classical notions of semantic consequence, noted $\models$, and equivalence, noted $\equiv$. A *conjunct* is a (possibly infinite) conjunction of atoms. A *fact* is the existential closure of a conjunct. W.l.o.g. we also see conjuncts and facts as *sets* of atoms. The *full fact* w.r.t. a vocabulary $\mathcal{V}$ contains all ground atoms that can be built on $\mathcal{V}$ (thus any fact on $\mathcal{V}$ is a semantic consequence of it). A $\forall\exists$*-rule* $R = (H, C)$ is a closed formula of form $\forall x_1 \dots \forall x_p(H \rightarrow (\exists z_1 \dots \exists z_q C))$ where H and C are two finite non empty conjuncts respectively called the *hypothesis* and the *conclusion* of R. The *frontier* of R (notation $fr(R)$) is the set of variables occurring in both H and C: $fr(R) = vars(H) \cap vars(C)$. In examples, we omit quantifiers and use the form $H \rightarrow C$.

The Deduction problem. The conjunction of two facts F_1 and F_2 is equivalent to a fact, say F; in the set-representation of facts, F is obtained by making the union of F_1 and F_2 after renaming variables. In the following, we identify a set of facts with a single fact. A *knowledge base* $K = (F, \mathcal{R})$ is composed of a fact F and a set of $\forall\exists$-rules $\mathcal{R}$. We address the following consequence/deduction problem (noted DEDUCTION in the following): "given a KB $K = (F, \mathcal{R})$ and a fact Q (built on the same vocabulary), is Q deducible from K (notation $K \models Q$) , *i.e.* does $\{F\} \cup \mathcal{R} \models Q$ hold?". An instance of this problem is denoted by $(F, \mathcal{R}, Q)$. As explained in the introduction, this problem is a representative of several other problems. It is undecidable (more precisely, it is semi-decidable).

Homomorphisms. Given a set of variables X and a set of terms T, a *substitution* of X by T is a mapping from X to T. Let $\sigma : X \rightarrow T$ be a substitution, and F be a fact. $\sigma(F)$ denotes the fact obtained from F by replacing each occurrence of $x \in X \cap terms(F)$ by $\sigma(x)$. A *safe substitution* is a bijection from X 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 $vars(Q)$ to $terms(F)$ such that $\sigma(Q) \subseteq F$. If there is a homomorphism from Q to F, we say that Q *maps to* F. It is well-known that homomorphism checking is sound and complete w.r.t. logical deduction in the fragment of facts: given two facts F and Q, $F \models Q$ iff Q maps to F.

Forward Chaining. We assume in this paper that the reader is familiar with forward and backward chaining paradigms. A $\forall\exists$-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 $var(C) \setminus fr(R)$. 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 a *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 $F_i = \alpha(F_{i-1}, R, \sigma)$. This notion yields a sound and complete forward mechanism: given a KB $K = (F, \mathcal{R})$ and a fact Q, $K \models Q$ iff Q maps to an $\mathcal{R}$-derivation of F (SM96).

Backward Chaining. The key operation in backward chaining is the unification between a fact Q, classically called a goal, and a rule conclusion, which produces a new goal. Since a precise definition of a unifier is not needed in this paper, we refer for it to (BLMS09). Let μ be a unifier of a fact Q and the conclusion of a $\forall\exists$-rule R. The rewriting of Q w.r.t. μ and R is a fact noted $\beta(Q, R, \mu)$. A fact Q' is called an $\mathcal{R}$*-rewriting* of Q if there is a finite sequence (called a *rewriting sequence*) $Q = Q_0, Q_1, \dots, Q_k = Q'$ such that for all $1 \leq i \leq k$, there is a rule $R = (H, C) \in \mathcal{R}$ and a unifier μ of Q_{i-1} and C with $Q_i = \beta(Q_{i-1}, R, \mu)$. This

notion yields a sound and complete backward mechanism: given a KB $K = (F, \mathcal{R})$ and a fact Q, $K \models Q$ iff there is an $\mathcal{R}$-rewriting of Q that maps to F (SM96)(BLMS09).

In the sequel of the paper, we simply write rules instead of $\forall\exists$-rules.

A single rule encodes all rules

Even very strong restrictions are not sufficient to make DEDUCTION decidable. In (Bag01), it is shown that DEDUCTION can be reduced to its restriction where the vocabulary is limited to a single binary predicate name. In (BM02), it is proven that DEDUCTION remains undecidable with rules with a frontier of size 2, and where the hypothesis and conclusion are paths. In this section, we present another very strong restriction (when the KB is restricted to a single rule) that is not sufficient to make the problem decidable.

Theorem 1 DEDUCTION *remains undecidable when the set of rules is restricted to a single rule.*

Proof: Let $I = (F, \mathcal{R}, Q)$ be an instance of DEDUCTION. By a transformation τ, we build another instance $\tau(I) = (\tau(F), \tau(\mathcal{R}), \tau(Q))$ with $|\tau(\mathcal{R})| = 1$, such that I is a positive instance if and only if $\tau(I)$ is. τ is defined as follows:

- Let $\mathcal{V}$ be the vocabulary obtained by considering the constants and the predicates occurring in I. We consider a vocabulary $\mathcal{V}_\tau$ obtained by replacing each predicate name of arity k in $\mathcal{V}$ by a predicate (of same name) of arity $k+1$ and by adding to this vocabulary two new constants f and g (f for "fact" and g for "garbage"). Each atom $p(t_1, \ldots, t_k)$ defined on $\mathcal{V}$ is translated into an atom $p(t_1, \ldots, t_k, t)$ on $\mathcal{V}_\tau$ where t is either f (stating that this atom corresponds to an atom in F or deduced from F), or g. Given a fact F over $\mathcal{V}$, such a translation is denoted $\tau(F, t)$.
- $\tau(F)$ is the disjoint union of two sets: a set $C_f = \tau(F, f)$ (the "fact component"); and a set $C_g = \tau(U, g)$ (the "garbage component") where U is the full fact on $\mathcal{V}$ (since any fact on $\mathcal{V}$ can be deduced from U, this latter fact encodes that everything is true, but is garbage).
- $\tau(Q) = \tau(Q, f)$ (we want to deduce it from the part of $\tau(F)$ that corresponds to F).
- Let $\mathcal{R} = \{R_1 \ldots R_p\}$. W.l.o.g., assume that the sets of variables occurring in each rule are pairwise disjoint. Let $x_1 \ldots x_p$ be new variables, i.e. not occurring in $\mathcal{R}$. Then $\tau(R_i) = (\tau(H_i, x_i), \tau(C_i, x_i))$. $\tau(\mathcal{R}) = \{R = (H = \cup_i \tau(H_i, x_i), C = \cup_i \tau(C_i, x_i))\}$ is composed of a single rule that encodes all the previous ones.

Let us outline the main ideas of this transformation. Every rule in $\mathcal{R}$ is applicable to the garbage component C_g, thus R is applicable to C_g, with variables $x_1 \ldots x_p$ being necessarily mapped to the constant g. When a rule R_i is applicable to F by a homomorphism h, then R is applicable to $\tau(F)$ with $\tau(H_i, x_i)$ being mapped to C_f by $h \cup \{(x_i, f)\}$, and the remaining H_j in H being mapped either to C_f or C_g. Conversely, assume that R is applicable to $\tau(F)$: each $\tau(H_i, x_i)$ is necessarily mapped to C_f or to C_g; if $\tau(H_i, x_i)$ is mapped to C_f, this corresponds to an application of R_i to F. If all $\tau(H_i, x_i)$ are mapped to C_g then the corresponding application of R is redundant (by definition of the full fact). It follows that every derivation from F with the rules in $\mathcal{R}$ can be translated into a derivation from $\tau(F)$ with R (with a natural extension of the homomorphisms involved in the first derivation) and reciprocally (with a natural decomposition of the homomorphisms involved in the second derivation). Finally, h is a homomorphism from Q to a fact F' defined on $\mathcal{V}$ iff it is a homomorphism from $\tau(Q)$ to $\tau(F')$ (and we have $h(\tau(Q)) \subseteq C'_f$ with $C'_f = \tau(F', f)$).

Note that $k+1$-ary predicates are not required for such a result: with a similar encoding, the same result can be obtained using only unary and binary predicates. □

Known abstract classes are not recognizable

We distinguish between several kinds of known decidable cases according to the properties defining them:

- *abstract classes* are defined by abstract properties that ensure decidability but for which the existence of a procedure for deciding whether a given set of rules fulfills the property is not obvious; in fact, we show in this section that none of the three known abstract classes is recognizable;
- *concrete classes* are defined by syntactic properties of a given set of rules. These properties are said to be *individual* whenever they can be checked independently on each rule in the set, and they are said to be *global* otherwise.

The decidable concrete classes found in the literature can be grouped into three abstract classes according to the properties that underlie their decidability: *finite expansion sets* are based on the finiteness of forward chaining; in their extension to *bounded-treewidth sets*, the produced facts have a tree-like structure, which allows to stop the forward chaining when the size of the facts is sufficient to conclude w.r.t. a given query; *finite unification sets* are based on the finiteness of backward chaining. We prove that none of them is recognizable.

A set of rules is called a finite expansion set (*fes*) 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.

Definition 1 (finite expansion set) *(BM02) $\mathcal{R}$ is called a* fes *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'.*

If $\mathcal{R}$ is a *fes*, 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 following definition of a bounded treewidth set of rules (Def. 3) basically follows from (CGK08). This abstract class translates the fundamental property underlying the concrete decidable classes in this latter paper.

A fact can naturally be seen as a hypergraph whose nodes are the terms in the fact and whose hyperedges encode atoms. The primal graph of this hypergraph has the same set of nodes (terms) and there is an edge between two nodes

is they belong to the same hyperedge (atom). The following treewidth definition for a fact corresponds to the usual definition for the associated primal graph.

Definition 2 (Treewidth of a fact) *Let F be a (possibly infinite) fact. A* tree decomposition *of F is a (possibly infinite) tree $T = (\mathcal{X} = \{X_1, \ldots, X_k, \ldots\}, U)$ where:*

1. *the X_i are sets of terms of F with $\bigcup_i X_i = terms(F)$;*
2. *For each atom a in F, there is $X_i \in \mathcal{X}$ s.t. $terms(a) \subseteq X_i$;*
3. *For each term e in F, the subgraph of T induced by the nodes X_i such that $e \in X_i$ is connected.*

The width *of a tree decomposition T is the size of the largest node of T, minus 1. The* treewidth *of a fact F is the minimal width among all its possible tree decompositions.*

Definition 3 (Bounded treewidth set) *(basically (CGK08)) A set of rules $\mathcal{R}$ is called a* bounded treewidth set (*bts*) *if for any fact F there exists an integer b such that, for any fact F' that can be $\mathcal{R}$-derived from F, treewidth$(F') \leq b$.*

Theorem 2 (Decidability of *bts*) [1] *The restriction of* DEDUCTION *to bounded treewidth sets of rules is decidable.*

Proof: Let $\mathcal{R}$ be a *bts*. Then for any fact F, there exists a bound b such that any fact $\mathcal{R}$-derivable from F has treewidth at most b. Consider the infinite fact F^* defined as the union of all facts $\mathcal{R}$-derivable from F. F^* is universal for F and $\mathcal{R}$, i.e. any fact consequence of $\{F\} \cup \mathcal{R}$ is consequence of F^*. Thanks to the treewidth compactness theorem (Tho88), F^* has bounded treewidth. Since F^* is universal, it follows that for any fact Q, both $F^* \wedge Q$ and $F^* \wedge \neg Q$ have a model of bounded treewidth when they are satisfiable. We conclude with (Cou90), that states that classes of first-order logic having the bounded treewidth model property are decidable. □

A *fes* is a *bts*, since the finite saturated graph generated by a *fes* from an initial fact F has treewidth bounded by its own size.

With finite unification sets, the finiteness of backward chaining is based on the finiteness of the set of most general rewritings of Q.

Definition 4 (Finite unification set) *(BLMS09) A set of rules $\mathcal{R}$ is called a finite unification set (*fus*) if for every fact Q, there is a finite set $\mathcal{Q}$ of $\mathcal{R}$-rewritings of Q such that, for any $\mathcal{R}$-rewriting Q' of Q, there is an $\mathcal{R}$-rewriting Q'' in $\mathcal{Q}$ that maps to Q'.*

We show now that *fes*, *bts* and *fus* yield abstract characterizations that are undecidable, with a proof applying to the three abstract classes.

Theorem 3 *Deciding if a set $\mathcal{R}$ is a finite expansion (resp. finite unification, resp. bounded treewidth) set is not decidable.*

The proof of this theorem relies on the following lemmas.

[1] This theorem is an immediate generalization of Th. 23 in (CGK08), that applies to the concrete *bts* class called "weakly guarded TGD".

Lemma 1 *Let $(F, \mathcal{R}, Q)$ be an instance of* DEDUCTION. *Let $\mathcal{V}$ be the vocabulary obtained by considering predicates and constants occurring in F, $\mathcal{R}$ and Q. We note $\mathcal{R}' = allrules(F, \mathcal{R}, Q) = \mathcal{R} \cup \{\emptyset \rightarrow F; Q \rightarrow U\}$ a new set of rules, where U is the full fact.*

Then $F, \mathcal{R} \models Q$ iff $\emptyset, \mathcal{R}' \models U$.

Proof: Since the fact F and the rule $\emptyset \rightarrow F$ are equivalent, we will prove successively both directions of the equivalence $F, \mathcal{R} \models Q$ iff $F, \mathcal{R} \cup \{Q \rightarrow U\} \models U$.
($\Rightarrow$) Immediate, since $F, \mathcal{R} \models Q$ and $Q, \{Q \rightarrow U\} \models U$ $\Rightarrow F, \mathcal{R}' \models U$.
($\Leftarrow$) If $F, \mathcal{R}' \models U$, then there exists a derivation $F = F_0, F_1, ..., F_k$ such that $F_k \models U$. Suppose that the rule $R_U = Q \rightarrow U$ is not used in this derivation. Then $F, \mathcal{R} \models U$ and since any fact can be deduced from U, we have $F, \mathcal{R} \models Q$. Otherwise, let us consider the smallest i such that F_i is obtained from F_{i-1} by the application of rule R_U. It means that there is a homomorphism from Q to F_{i-1} (applicability of the rule) and that $F, \mathcal{R} \models F_{i-1}$ (R_U was not needed). Then $F, \mathcal{R} \models Q$. □

Lemma 2 *Let $(F, \mathcal{R}, Q)$ be an instance of* DEDUCTION. *Let $\mathcal{R}' = allrules(F, \mathcal{R}, Q)$ be defined as in lemma 1. If $\emptyset, \mathcal{R}' \models U$, then $\mathcal{R}'$ is a* fes*, a* fus*, and a* bts.

Proof: Assume $\emptyset, \mathcal{R}' \models U$. We successively prove all three implications:
1) It follows that, for any fact H on $\mathcal{V}$, we have $H, \mathcal{R}' \models U$ and then the forward chaining algorithm produces in finite time a fact F' such that $F' \models U$ (from semi-decidability of DEDUCTION proven with forward chaining). Thus $F' \equiv U$ and any fact that can be derived from F' is also equivalent to U: it means that $\mathcal{R}'$ is a *fes*.
2) Since all *fes* are also *bts*, $\mathcal{R}'$ is also a *bts*.
3) It follows that, for any fact Q', we have $\emptyset, \mathcal{R}' \models Q'$ and then a breadth-first exploration of all possible rewritings of Q' will produce $\emptyset$ in finite time (from semi-decidability of DEDUCTION proven with backward chaining). Since $\emptyset$ is more general than any other rewriting of Q', $\mathcal{R}'$ is a *fus*. □
Proof: [Th. 3] (By absurd). Assume there exists a halting, sound and complete algorithm that determines whether a set of rules is a *fes* (resp. a *bts*, resp. a *fus*). Then we exhibit the following halting, sound and complete algorithm for DEDUCTION.

Data: $(F, \mathcal{R}, Q)$ an instance of DEDUCTION
Result: YES iff $F, \mathcal{R} \models Q$, NO otherwise.
1 **if** $\mathcal{R}' = allrules(F, \mathcal{R}, Q)$ *is a* fes *(resp.* fus*, resp.* bts*)* **then**
2 | **return** YES *iff* $\emptyset, \mathcal{R}' \models U$*, and* NO *otherwise*;
3 **else return** NO;

This algorithm halts: the condition in line 1 is checked in finite time (our hypothesis), and if this condition is fulfilled then the semantic consequence (line 2) can also be checked in finite time. This algorithm is also sound and complete: line 2 returns the correct answer (lemma 1) and, assuming the condition is not verified, line 3 also returns the correct answer (from lemma 1 and contrapositive of lemma 2, we have "if $\mathcal{R}'$ is not a *fes* then $F, \mathcal{R} \not\models Q$"). □

Extending concrete classes

Since abstract classes are not recognizable, it is important to have as large as possible concrete subclasses of these abstract classes. In this section, we first review known concrete cases. Then we introduce new concrete classes that generalize all known concrete *bts* classes based on individual criteria.

Known concrete cases

Let us begin with the list of concrete classes based on *individual* criteria (we mention after each case the abstract argument that was initially used to prove decidability).

- *range restricted rules*[2] (*rr*), which do not have existentially quantified variables [*fes*]. They correspond to rules in positive Datalog;
- *disconnected rules* (*disc*), whose frontier is empty (BM02) [*fes*]; note that the hypothesis and conclusion may share constants (but not variables);
- *guarded rules* (*g*), such that an atom of the hypothesis contains ("guards") all variables of the hypothesis (CGK08) [*bts*];
- *atomic hypothesis rules* (*ah*), whose hypothesis is restricted to a single atom (BLMS09) [*fus*]. They are special guarded rules, thus are also *bts*;
- *domain-restricted rules* (*dr*), in which each atom in the conclusion contains all or none of the variables in the hypothesis (BLMS09) [*fus*] ; they include *disc* rules, which are thus also *fus*.

Example 1
$R_1 = r(x,y) \wedge r(y,z) \rightarrow r(x,z)$ is only *rr*
$R_2 = r(x,y) \wedge r(y,z) \rightarrow r(u,v)$ is only *disc* (and *dr* since $disc \subseteq dr$)
$R_3 = t(x,y,z) \wedge q(x) \rightarrow t(y,z,u)$ is only *g*
$R_4 = t(x,y,z) \rightarrow t(y,z,u) \wedge q(y)$ is only *ah* (and *g*)
$R_5 = r(x,y) \wedge r(y,z) \rightarrow o(x,y,z,t) \wedge r(t,u)$ is only *dr*
$R_6 = r(x,y) \wedge r(y,z) \rightarrow r(z,u)$ does not belong to any of the above classes.

Let us also mention the database *inclusion dependencies* (ID) in which the hypothesis and conclusion are restricted to a single atom. The original decidability proof for IDs was complex (JK84). Since IDs are special *ah*, both *bts* and *fus* argument yield simple new proofs. In (BLMS09) the case of rules with a *frontier of size one* ($fr1$) is mentioned (unpublished proof from Baget). In previous example, R_6 is the only *fr1*-rule. Note that *fr1*-rules, *g*-rules and *disc*-rules are incomparable classes. The three classes will be seen as subclasses of the more general class of *frontier-guarded rules* introduced hereafter.

Let us now turn our attention to concrete classes defined by global properties. The *g*-rule class is generalized by the class of *weakly guarded rules* (*wg*), in which only some variables of the hypothesis need to be guarded (CGK08). Given a set of rules $\mathcal{R}$, a position i in a predicate p (notation (i,p)) is said to be *affected* if it may contain a new variable generated by forward chaining, i.e.: (1) if there is a rule conclusion containing an atom with predicate p and an existentially quantified variable at position i, then position (i,p) is affected; (2) if a rule hypothesis contains a variable x appearing at an affected position (i,p) and x appears in the conclusion of this rule in position (j,q) then (j,q) is affected. Given $\mathcal{R}$, a weak guard in a rule $(H,C) \in \mathcal{R}$ is an atom in H that contains all variables in H that occur only in affected positions (i.e. do not occur in a non-affected position); these variables are said to be *affected*. $\mathcal{R}$ is said to be *weakly guarded* if each rule in $\mathcal{R}$ has a weak guard. In previous example, $\{R_2\}$, $\{R_5\}$ and $\{R_6\}$ are not weakly guarded. *wg* are *bts* (CGK08). Special cases of *wg*-rules are *g*-rules (a guard is a weak guard) and *rr*-rules (no position is affected), both based on individual properties.

Two other concrete classes defined by global properties found in the literature are *weakly acyclic rules* (wa) (FKMP03)(DT03) and sets of rules with an *acyclic graph of rule dependencies* ($aGRD$) (Bag04).

The first graph, introduced for TGD and called *dependency graph*[3], encodes variable sharing between positions in predicates. The nodes represent the positions in predicates (cf. the notation *(p,i)* introduced for *wg* rules). For each rule $R = (H,C)$ and each variable x in H occurring in position (p,i): if $x \in fr(R)$, there is an edge from (p,i) to each position of x in C; furthermore, for each existential variable y in C (i.e. $y \in var(C) \setminus fr(R)$) occurring in position (q,j), there is a special edge from (p,i) to (q,j). The set of rules is weakly acyclic if this graph has no circuit passing through a special edge.

The *graph of rule dependencies* encodes possible interactions between rules: the nodes represent the rules and there is an edge from R_1 to R_2 iff an application of the rule R_1 may create a new application of the rule R_2 (with this abstract condition being effectively implemented by a unification operation). $aGRD$ is the case where this graph is without circuit[4].

New concrete cases

Definition 5 ((Weakly) Frontier-guarded rules) *Given a set of variables S, a rule is S-*guarded *if an atom of its hypothesis contains (at least) all variables in S. A rule is* frontier-guarded *(*fg*) if it is S-guarded with S being its frontier. A set of rules is* weakly frontier-guarded *(*wfg*) if each rule is S-guarded with S being the set of affected variables in its frontier.*

[2]These rules have long been used in logic programming and deductive databases. The name "range-restricted" is from (AHV95). Other names found in databases are *full* implicational dependencies (CLM81) and *total* tuple-generating dependencies (BV84).

[3]We use here the terminology of (FKMP03), developed in (FKMP05).

[4]Unfortunately, the term "acyclic" is ambiguous when used on directed graphs. In this paper, by acyclic we mean without any undirected cycle (i.e. the underlying undirected graph is a forest). We keep the expressions "acyclic GRD" and "weakly acyclic" that come from other papers, but precise that they refer to circuits.

The class of frontier-guarded rules includes *g*-rules, *fr1*-rules and *disc*-rules. The class of weakly frontier-guarded rules generalizes it as well as the class of weakly guarded rules, which itself generalizes range-restricted rules. In particular, it covers all known decidable classes (to the best of our knowledge) having the *bounded treewidth set* property and based on individual criteria.

Example 2
$R_7 = r(x,y) \wedge C(y) \wedge r(x,z) \wedge D(z) \rightarrow s(x,u) \wedge E(u)$ is not *g* but *fr1* since the frontier is restricted to x, thus it is *fg*.
$R_8 = r(x,y) \wedge r(y,z) \rightarrow s(x,u) \wedge s(y,u)$ is not *g* nor *fr1* but it is *fg*.
$R_9 = r(x,z) \wedge s(y,z) \rightarrow s(y,u) \wedge r(u,x)$ is not *fg* (the frontier is $\{x,y\}$); taken as a singleton it is not *wg* either (the affected variables in H are x and z), but it is *wfg* (since $r(x,z)$ guards x, which is the only variable both affected and in the frontier).
In the first example, R_2, R_3, R_4, R_6 are all *fg*; R_5, which is *dr*, is neither *fg* nor *wfg*.

$\forall\exists$-rules are well-suited to represent ontological knowledge, especially the kind of axioms forming the core of recent description logics tailored for efficient query answering. Typically, $\forall\exists$-rules allow to express inclusions between concepts built with conjunction ($\sqcap$) and full existential restriction ($\exists r.C$), as well as role inclusions, domain and range restrictions, reflexivity and transitivity role properties... The first-order translation of these assertions yields rules that, besides the fact that they have an "acyclic" hypothesis and conclusion, are special cases of previous concrete classes. Disjointness axioms and functionality axioms are other widely used ontological assertions. To represent them, our framework has to be extended with negative constraints (which generalize disjointness axioms) and rules with equality (which generalize functionality axioms). A negative constraint can be seen as a forbidden fact (BM02) or as a rule with a conclusion restricted to the special symbol $\bot$ (always false) (CGL09). An equality rule is of form $H \rightarrow x = y$, where x and y are in H. Note that, in these recent DLs, as the both we cite below, disjointness and functionality axioms can be processed as constraints applied to the initial ABox (the initial set of facts) and do not interfere with query answering. For instance, (CGL09) shows that the major members of the DL-Lite family (CGL$^+$07) are covered by guarded rules (plus negative constraints and specific equality rules). Another example of DL covered by $\forall\exists$-rules is $\mathcal{ELH}^{dr}_{\bot}$ (LTW09): it can be easily checked that all inclusions in this DL are *fr1*-rules or *ID*, thus they are *fg*-rules (with $\bot$ being processed by a negative constraint). E.g. the rule R_7 in example 2 translates the following $\mathcal{ELH}^{dr}_{\bot}$ inclusion: $\exists r.C \sqcap \exists r.D \sqsubseteq \exists s.E$. Finally, note that rules expressing transitivity are not *fg*, but *rr*, thus both *wg* and *wfg*. The weakly frontier-guarded class thus seems particularly appropriate for studying ontological fragments.

In the following, we prove that (weakly) frontier-guarded rule sets are bounded treewidth sets. We introduce the notion of a derivation graph. This graph is of interest in itself because it allows us to explain properties of rules by structural properties of the facts they produce. We call *frontier atom* in a rule R an atom in the hypothesis of R that contains at least one frontier variable. Frontier atoms play an important role in the next definitions.

Definition 6 (Derivation Graph) *Let $D = (F = F_0, F_1, \ldots, F_n = F')$ be a derivation sequence. The Derivation Graph assigned to D is the directed graph $G_D = (\mathcal{X}, E, newAtoms, label)$, where $\mathcal{X}$ is the set of nodes, E is the set of edges, and $newAtoms$ and $label$ are functions respectively labeling nodes and edges, such that:*

- $\mathcal{X} = \{X_0 \ldots X_n\}$,
- *$newAtoms$ assigns to each $x_i \in \mathcal{X}$ the set of atoms created at step i, i.e. $newAtoms(X_0) = F$ and for $1 \leq i \leq n, newAtoms(X_i) = F_i \setminus F_{i-1}$. Furthermore, we note $terms(X_i) = terms(newAtoms(X_i))$.*
- *there is an edge (X_i, X_j) in E if: let $F_j = \alpha(F_{j-1}, R, \sigma)$; there are $a \in newAtoms(X_i)$ and b a frontier atom in R with $\sigma(b) = a$; $label(X_i, X_j) = \{e \in terms(X_i) | \exists a \in newAtoms(X_i)$ s.t. $e \in terms(a), \exists b$ frontier atom in R with $x \in terms(b) \cap fr(R)$, $\sigma(b) = a$ and $\sigma(x) = e\}$.*

Roughly speaking, nodes and their labeling encode atoms created at each derivation step; each edge (X_i, X_j) expresses that the homomorphism σ from a rule hypothesis H to F_{j-1}, that has led to F_j, has mapped at least one frontier atom in H to an atom (in F_{j-1}) created in F_i; the label of (X_i, X_j) indicates the terms in F_i that are used to produce the new atoms in F_j. By definition, a derivation graph has no circuit, but it is generally not acyclic (i.e. it is not a tree, or a forest if not connected). Every application of a disconnected rule leads to a node initially isolated, thus the graph may be not connected.

Property 1 (Decomposition properties) *Let $(F, \mathcal{R})$ be a KB such that no rule in $\mathcal{R}$ has a constant in its conclusion. Then, for any $\mathcal{R}$-derivation D from $F = F_0$ to $F_n = F'$, G_D satisfies the following properties, called the* decomposition properties w.r.t. F':

1. *$\bigcup_i X_i = terms(F')$;*
2. *For each atom a in F', there is $X_i \in \mathcal{X}$ s.t. $a \in newAtoms(X_i)$;*
3. *For each term e in F', the subgraph of G_D induced by the nodes X_i such that $e \in terms(X_i)$ is connected.*
4. *For each $X_i \in \mathcal{X}$, the size of $terms(X_i)$ is bounded by an integer that depends only on the size the KB (here $max(|terms(F)|, |terms(C_i)|_{R_i \in \mathcal{R}})$.*

Proof: The proof of conditions 1), 2) and 4) being immediate, we focus here on condition 3). Every edge labeled e links two nodes containing e. For each term e in F', there exists X_e a node corresponding to F_e, the first derived graph in which e appears (if e has been generated by a rule application then X_e identifies that rule application, otherwise e belongs to F and $X_e = X_0$). Moreover, if X_i contains a term e then F_e (the graph associated to X_e) has been generated before F_i in the derivation sequence. We can thus establish the following property: "for each node X_i such that $e \in terms(X_i)$, there exists a path from X_e to X_i in which

all nodes contain e and all edge labels contain e". This property can be easily proven by a recurrence on the length of the derivation from F_e to F_i. □

Note that the third decomposition property is not true for constants occurring in a rule conclusion. We will process these constants in a special way together with the notion of affected variable.

Property 1 expresses that D_G satisfies the properties of a tree decomposition of F' (seen as a graph) except that it is not —yet— acyclic. We now introduce operations that allow to build an acyclic graph from D_G for some classes of rules, while keeping these properties.

Definition 7 (Reduction operations on Derivation Graphs)

- ***Redundant edge removal.*** *Let (X_i, X_k) and (X_j, X_k) be two edges with the same endpoint. If a term e appears in $label(X_i, X_k)$ and $label(X_j, X_k)$, then e can be removed from one of the label sets. If the label of an edge becomes empty, then the edge is removed.*
- ***Edge contraction.*** *Let (X_i, X_j) be an edge. If $terms(X_j) \subseteq terms(X_i)$ then X_i and X_j can be merged into a node X such that $newAtoms(X) = newAtoms(X_i) \cup newAtoms(X_j)$. This merging involves the removal of (X_i, X_j) and, in all other edges incidental to X_i or X_j, X_i and X_j are replaced by X, with multiedges being replaced by a single edge labeled by the union of their labels.*

Property 2 *The above operations preserve the decomposition properties w.r.t. F'.*

Proof: Conditions 1), 2) and 4) are trivially respected by both operations. No atom (and thus no term) disappears in the derivation graph, and no node receives any additional atom (since the only merging of nodes happens when a set is included in the other).

Condition 3) is satisfied by edge contraction, which does not change the connectivity of the graph. Let us consider redundant edge removal. For each node X that contains a term e there exists a path from X_e to X (see proof of prop. 1) in which all nodes and edges are labeled e. Moreover, nodes incident to an edge labeled e also contain e, thus if a node X_k has two parents X_i and X_j, these two latter nodes also contain e and then there is a second path from X_e to X_k. By removing one of these edges, it is impossible to disconnect the set of nodes containing e. □

Theorem 4 *Let $\mathcal{R}$ be a set of rules without constant in conclusion. If for all $\mathcal{R}$-derivation D, G_D can be reduced to an acyclic graph then $\mathcal{R}$ is a bounded treewidth set.*

Proof: Follows from Prop. 1 and 2. □

Property 3 *If all rules are range-restricted and without constant in their conclusion, then any derivation graph with these rules can be reduced to a single node by a sequence of edge contractions.*

Proof: If all rules are *rr*, all terms in generated atoms are contained in the root of the derivation graph. We can thus iteratively contract all edges of the derivation graph into the root. □

Property 4 *If all rules are frontier-guarded and without constant in their conclusion, then any derivation graph with these rules can be reduced to an acyclic graph.*

Proof: We show that if a node X of the derivation graph is the destination of $n \geq 2$ distinct edges, then $n - 1$ of them can be suppressed by redundant edge removal. We begin by pointing out that, to be the destination of an edge, X must have been obtained by applying some rule R that contains at least one frontier node (*i.e.* R is not *disc*). Moreover, by definition of a derivation graph, these edges' labels are necessarily a subset of the terms that were images of the frontier nodes of R. In frontier-guarded rules, the guard g of R (*i.e.* the atom containing the frontier) generates an edge (X_g, X) in the derivation graph, where X_g contains the image of g. This edge is labeled by all terms of the frontier of R, and thus any other edge whose destination is X is redundant with (X_g, X) and can be removed. □

Property 5 *Frontier-guarded rules without constant in their conclusion are* bts.

Proof: Immediate consequence of prop. 4 and theorem 4. □

To cover rules that introduce constants, as well as weakly frontier-guarded rules, we extend the notion of derivation graph.

Definition 8 (Extended Derivation Graph) *Given a set of terms T and a derivation graph G_D, the extension of G_D with T, notation $G_D[T]$, is obtained from G_D with the following sequence of operations:*

1. *the mapping $terms$ is modified: for each X_i, $terms(X_i) = terms(newAtoms(X_i)) \cup T$ (i.e. the terms of T are added everywhere);*
2. *all terms occurring in T are removed from the labels in edges; if a label becomes empty, then the edge is removed;*
3. *for each connected component in G_D that does not include X_0, a node X_i without incoming edge is chosen and the edge (X_0, X_i) is added with label T.*

Property 6 *$G_D[T]$ satisfies the decomposition properties, with the bound on $|terms(X_i)|$ being increased by $|T|$; furthermore $G_D[T]$ does not contain new cycles w.r.t. G_D.*

Proof: There is no suppression of atoms so the terms and atoms of F' remain covered. We add at most $|T|$ terms to each node thus the width (and consequently the treewidth) of the derivation graph is at most increased by $|T|$. Global connectivity is ensured since T is added to all nodes of the derivation graph. Since edges are added only to reconnect disconnected components, no circuit is created. □

Theorem 4 and properties 3, 4, 5 can be extended to rules with constants in their conclusion by considering the extended derivation graph with T being the set of constants occurring in rule conclusions.

Property 7 *Let $\mathcal{R}$ be a set of rules and let C be the set of constants occurring in the rule conclusions. If $\mathcal{R}$ is* frontier-guarded*, then, for any $\mathcal{R}$-derivation D, $G_D[C]$ can be reduced to a tree. If $\mathcal{R}$ is* weakly frontier-guarded*, then, for any $\mathcal{R}$-derivation D, $G_D[C \cup terms(F)]$ can be reduced to a tree.*

Proof: The proof is similar to the proofs of properties 4 and 6. A key property is that a non-affected variable in a rule hypothesis is never mapped to a new variable (i.e. not occurring in the initial fact) by an application of this rule. □

Theorem 5 *Weakly frontier-guarded rule sets are* bts.

Figure 1 summarizes inclusions between decidable cases. All inclusions are strict and no inclusion is omitted (i.e. classes not related in the schema are indeed incomparable). The examples provided before allow to check most cases and it is easy to build other examples proving this claim. We add below some examples showing the incomparability of classes based on global criteria, namely wa, $aGRD$ and wfg.

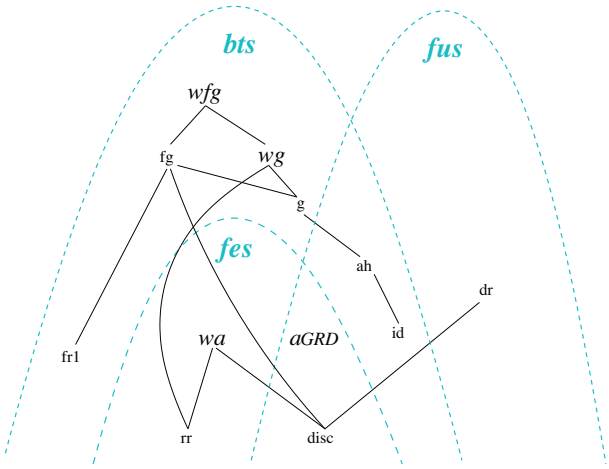

Figure 1: Inclusions between decidable cases

Example 3
$\{q(x) \rightarrow p(z,x),\ p(x,z) \wedge p(y,z) \rightarrow r(x,y)\}$ is *wa* and *aGRD* but is not *wfg* because the frontier variables x and y in the second rule are affected but not guarded
$p(x,y) \rightarrow p(y,z)$ is *wfg* (because it is *fr1*) but it is not *wa* neither *aGRD* (this rule depends on itself).
$p(x,y) \wedge q(y) \rightarrow p(y,z) \wedge s(z)$ is *aGRD* and *wfg* (because it is *fr1*) but it is not *wa*.
$q(x) \wedge p(x,y) \rightarrow q(y) \wedge r(y,z)$ is *wa* and *wfg* (because it is *fr1*) but it is not *aGRD* (this rule depends on itself).

Study of the union of decidable classes

Let us say that two decidable classes are *incompatible* if the union of two sets respectively belonging to these classes may be undecidable.

Universal compatibility of disconnected rules

We first prove that disconnected rules are compatible with any decidable set of rules.

Theorem 6 *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_{disc}$ be a set of rules, where R_{disc} is a set of disconnected rules. If $\mathcal{R}_0$ is decidable, then $\mathcal{R}$ also is.*

Proof: The key property of a disconnected rule is that it needs to be applied only once: any further application of it is redundant. Assume we have an algorithm for DEDUCTION, say Ded, that decides in finite time if $F, \mathcal{R}_0 \models Q$ for any F and Q. We extend this algorithm to an algorithm that decides in finite time if $F, \mathcal{R} \models Q$ for any F and Q, as follows:

Data: $(F, \mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_{disc}, Q)$
Result: YES iff $F, \mathcal{R} \models Q$, NO otherwise.
$F' \leftarrow F$;
repeat
  forall $R_D = (H_D, C_D) \in \mathcal{R}_{disc}$ **do**
    if $Ded(F', \mathcal{R}_0, H_D)$ **then**
      $F' \leftarrow F' \cup \{C_D\}$ (with a safe substitution);
      Remove R_D from $\mathcal{R}_{disc}$;
until *stability of* $\mathcal{R}_{disc}$;
return $Ded(F', \mathcal{R}_0, Q)$;

□

Incompatibility results

We say that two sets of rules $\mathcal{R}_1$ and $\mathcal{R}_2$ are *equivalent* w.r.t. a vocabulary $\mathcal{V}$ if, for any fact F built on $\mathcal{V}$, the sets of facts on $\mathcal{V}$ deducible respectively from knowledge bases $(F, \mathcal{R}_1)$ and $(F, \mathcal{R}_2)$ are equals. Let us now consider two simple transformations from a rule into an equivalent pair of rules:

- τ_1 rewrites a rule $R = (H, C)$ into two rules:
  $R_h = H \rightarrow R(x_1 \dots x_p)$ and
  $R_c = R(x_1 \dots x_p) \rightarrow C$, where $\{x_1 \dots x_p\} = vars(H)$ and R is a new predicate (i.e. not belonging to the vocabulary) assigned to the rule. Note that R_h is both range-restricted and domain restricted, and R_c is atomic hypothesis.
- τ_2 is similar to τ_1, except that the atom $R(...)$ contains all variables in the rule: $R_h = H \rightarrow R(y_1 \dots y_k)$ and
  $R(y_1 \dots y_k) \rightarrow C$, where $\{y_1 \dots y_k\} = vars(R)$. Note that, among other properties, R_h is domain-restricted, while R_c is range-restricted.

Property 8 *Any set of rules can be split into an equivalent set of rules by τ_1 or τ_2.*

Proof: For τ_1, we prove that, given a set of rules $\mathcal{R}$ and a fact F, both on a vocabulary $\mathcal{V}$, there is an $\mathcal{R}$-derivation from F to a fact F' iff there is a $\tau_1(\mathcal{R})$-derivation from F to a fact F'' s.t. the restriction of F'' to the vocabulary $\mathcal{V}$ is isomorphic to F'. For each part of the equivalence, the proof can be done by recurrence on the length of a derivation. In the $\Rightarrow$ direction, it suffices to decompose each step of the $\mathcal{R}$-derivation according to τ. In the $\Leftarrow$ direction, we show that any $\tau(\mathcal{R})$-derivation can be reordered so that the rule applications corresponding to the application of a rule in $\mathcal{R}$ are consecutive in the derivation. The reason is that the atom $R(\dots)$ added by a rule application according to a given homomorphism keeps (at least) all information needed to apply R according to this homomorphism and cannot be used to apply another rule. For τ_2, the $\Rightarrow$ direction holds. The $\Leftarrow$ direction holds for a $\tau_2(\mathcal{R})$-derivation that is "com-

plete" w.r.t. the $R(\ldots)$ atoms, i.e. such that all rules applicable w.r.t. a homomorphism to an $R(\ldots)$ atom have been applied. Since any $\tau_2(\mathcal{R})$-derivation can be completed in a minimal way, we obtain the equivalence between $\mathcal{R}$ and $\tau_2(\mathcal{R})$. □

Theorem 7 *Any instance of* DEDUCTION *can be reduced to an instance of* DEDUCTION *with a set of rules restricted to two rules, such that each rule belongs to a decidable class.*

Proof: From Th.1, any instance of DEDUCTION can be encoded by an instance with a single rule, say R. By splitting R with τ_1 or τ_2, we obtain the wanted pair of rules. □

If we furthermore consider the concrete classes of the rules obtained by both transformations, we obtain the following result:

Theorem 8 DEDUCTION *remains undecidable if $\mathcal{R}$ is composed of*

- *a range-restricted rule and an atomic-hypothesis rule*
- *a range-restricted rule and a domain-restricted rule*
- *an atomic-hypothesis rule and a domain-restricted rule.*

Since *ah*-rules are also *g*-rules, this implies that *g*-rules are incompatible with *rr*-rules and *dr*-rules. The case of *fr1* is more tricky. We did not find any transformation from general rules into *fr1* rules (and other rules belonging to compatible decidable classes). To prove the incompatibility of *fr1* and *rr* (Th. 9), we use a reduction from the halting problem of a Turing Machine. This reduction transforms an instance of the halting problem into an instance of DEDUCTION, in which all rules are either *fr1* or *rr*. The compatibility of *fr1* and *dr* is an open question.

Theorem 9 DEDUCTION *remains undecidable if $\mathcal{R}$ is composed of* fr1*-rules and* rr*-rules.*

Proof: See Appendix. □

The following table synthesizes decidability results for the union of decidable classes based on individual criteria; ND means "not preserving decidability".

rr	fes *(wa)*					
id/ah	fg	ND				
g	fg	ND	g			
fr1	fg	ND	fg	fg		
fg	fg	ND	fg	fg	fg	
dr	dr	ND	ND	ND	Open	ND
	disc	**rr**	**id/ah**	**g**	**fr1**	**fg**

We can also conclude for concrete classes based on global criteria, i.e. *wg*, *wfg*, *wa* and *aGRD*: all of them are incompatible, which includes the incompatibility of each class with itself (indeed, the union of two sets satisfying a global property does generally not satisfy this property; only one added rule may lead to violate any of the above criteria).

Theorem 10 *The union of two sets belonging to classes* wg*,* wfg*,* wa *and* aGRD *does not preserve decidability.*

Proof: See that the transformation τ_1 decomposes a rule into two rules R_h and R_c s.t. $\{R_h\}$ and $\{R_c\}$ are each *wa*, *aGRD* and *wg*. Let I be any instance of DEDUCTION. I is transformed into an instance containing a single rule by the reduction in the proof of Th. 1. Let I' be the instance obtained by applying τ_1 to this rule. The set of rules in I' is the union of two (singleton) sets both wa, $aGRD$ and wg. Since I' is a positive instance iff I is, we have the result. □

It follows from previous results that abstract classes are incompatible:

Theorem 11 *The union of two sets belonging to classes* fes*,* bts *or* fus *does not preserve decidability.*

Proof: Follows from Th. 8 (for all possible pairs except *fes/fes*) and 10 (for the pair *fes/fes*). □

To conclude, the rough union of two sets of rules belonging to different decidable classes almost always leads to undecidability. The precise study of interactions between rules, as done in (BLMS09), is thus a promising approach. DEDUCTION is decidable when the graph of rule dependencies has no circuit (we have the $aGRD$ class). Even more interesting is the fact that when all connected components of this graph are *fes* (resp. *fus*, resp. *bts*), then the set of rules is a *fes* (resp. *fus*, resp. *bts*). By combining abstract classes we are able to effectively combine concrete classes implementing their behavior, and thus take all classes presented here into account, as well as those we are not yet aware of. With additional conditions on this graph, it is possible to combine a *bts* and a *fus* into a new decidable class (that strictly contains both *bts* and *fus*) using a mixed forward/backward chaining algorithm. This shows that using abstract classes is a powerful method for building generic decidability results.

Perspectives

We have pointed out the interest of precisely studying interactions between rules to enlarge decidable cases. Two techniques for encoding these interactions can be found in the literature and have been mentioned previously: one relies on the graph of rule dependencies and the other on a graph of position dependencies. Both graphs encode different kinds of interactions between rules. We are currently investigating a method combining these two techniques with the aim of gaining greater insight into interactions between rules.

Other future work includes precise studies of DEDUCTION complexity for all concrete decidable cases (pursuing the results in (BM02) and (CGK08)).

In the section devoted to new concrete cases, we pointed out that frontier-guarded rules, and their generalization to weakly frontier-guarded rules, along with negative constraints and equality rules, allow to encode interesting ontological fragments, including some recent DLs tailored for query answering. A general framework dedicated to conjunctive query answering with ontologies is proposed in (CGL09). This Datalog-based framework is called Datalog$^{\pm}$: on one hand, this framework extends Datalog (i.e. range-restricted rules) with TGDs (i.e. $\forall\exists$-rules), equality generating dependencies (EGDs, i.e. equality rules) and negative constraints, on the other hand it restricts TGDs and EGDs to achieve decidability and tractability. Our decidable classes can be seen as new members of this family, which

generalize the members studied in (CGL09). Note however that the complexity of query answering (i.e. DEDUCTION) with these new classes of rules remains to be studied.

Acknowledgements

We thank Georg Gottlob for useful references.

References

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

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.

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

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

J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. DL-$\mathcal{SR}$: a lite DL with expressive rules: Preliminary results. In *Description Logics*, 2008.

J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *IJCAI*, pages 677–682, 2009.

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

C. Beeri and M.Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *KR*, pages 70–80, 2008.

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.

A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS*, pages 77–86, 2009.

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

A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *STOC*, pages 342–354. ACM, 1981.

B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

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

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

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.

P. Hayes, editor. *RDF Semantics*. W3C Recommendation. W3C, 2004.

D.S. Johnson and A.C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28(1):167–189, 1984.

C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic el using a relational database system. In *IJCAI*, pages 2070–2075, 2009.

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.

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

R. Thomas. The tree-width compactness theorem for hypergraphs. http://people.math.gatech.edu/ thomas/PAP/twcpt.pdf, 1988.

Appendix

Proof of Th. 9: DEDUCTION *remains undecidable if* $\mathcal{R}$ *is composed of* fr1*-rules and* rr*-rules.*

Proof: We consider the halting problem of a Turing machine: given a Turing machine $\mathcal{M}$ (with an infinite tape and a single final state) and a word m, s.t. the head of $\mathcal{M}$ initially points to the first symbol of m, does $\mathcal{M}$ accept m, i.e. is there a sequence of transitions leading $\mathcal{M}$ to the final state ? We build a reduction from this problem to DEDUCTION, such that each rule obtained is *fr1* or *rr*. Let us call *configuration* a global state of the Turing machine, i.e. its current control state, the content of the tape and the position of the head. The basic idea of the translation is that each transition is translated into a logical rule. However, whereas transitions can be seen as *rewriting* rules, logical rules are only able to *add* atoms. To simulate the rewriting of a configuration, we add a library of eight rules, called hereafter the *copy rules*. The rule assigned to a transition creates three new cells (a copy of the current cell, that contains the new symbol, and neighboring cells with the new position of the head), and the copy rules build the other relevant cells at the right and at the left of these new cells.

Let $(\mathcal{M}, m)$ be an instance of the halting problem. We build an instance $(F, \mathcal{R}, Q)$ of DEDUCTION as follows.

The vocabulary is composed of:

- binary predicates: $Succ$ to encode the succession of cells ($Succ(x, y)$ means that the cell x is followed by the cell y); $Value$ to indicate the content of a given cell ($Value(x, y)$ means that the cell x contains the symbol y); $Head$ to indicate the current position of the head and the current control state ($Head(x, y)$ means that the head points to cell x and the current state is y); $Next$ to encode the rewriting of a cell ($Next(x, y)$ means that cell x is rewritten as cell y); $Copy_r$ (resp. $Copy_l$) to rebuild the right (resp. the left) part of the word after a transition: $Copy_r(x, y)$ and $Copy_l(x, y)$ both mean that cell y is a copy of cell x in the next configuration;
- constants: each state T_i and each symbol v_i are translated into constants with the same name. Furthermore, there are three special constants, noted $\Box$ (the value of an empty cell), B (for Begin) and E (for End).

Let $m = m_1 \dots m_k$ and let T_0 be the initial state. F is obtained from this initial configuration. m is translated into a path of atoms with predicate $Succ$ (a "*Succ*-path") on variables $x_1 \dots x_k$, as well as atoms with predicate $Value$ that relate each x_i with the symbol m_i; for the needs of the copy mechanism, we actually translate the following word: "$\Box\ m_1 \dots m_k\ \Box$", and add special markers B and E at the extremities of this word. More precisely:
$F = \{Succ(B, x_0),$
$Succ(x_0, x_1), \dots Succ(x_k, x_{k+1}),$
$Succ(x_{k+1}, E),$
$Value(x_0, \Box), Value(x_{k+1}, \Box),$
$Value(x_1, m_1), \dots Value(x_k, m_k),$
$Head(x_1, T_0)\}$.

Note that there are no atoms $Value(B, \dots)$ and $Value(E, \dots)$.

Let $\delta = (T_i, v_p) \rightarrow (T_j, v_q, d)$ be a transition, with $d \in \{r, l\}$ indicating a move to the right (r) or to the left (l): δ can be read as "if the current state is T_i and the head points to the symbol v_p, then take state T_j, replace v_p by v_q and move to the right/left". Let $R(\delta)$ be the logical rule assigned to δ. If $d = r$, we have:

$R(\delta) = Head(x, T_i) \wedge Value(x, v_p) \rightarrow Next(x, y) \wedge Succ(z, y) \wedge Succ(y, t) \wedge Value(y, v_q) \wedge Head(t, T_j)$. This rule is *fr1*. The case $d = l$ is symmetrical: the head moves to the left.

To implement the copy mechanism, we have four rules to copy the right part of the word, and four symmetrical rules to copy its left part. Here are the four "right-copy" rules:
$R_{r1} = Succ(x, y) \wedge Next(x, z) \wedge Succ(z, u) \wedge Value(y, v) \rightarrow Copy_r(y, u) \wedge Value(u, v)$
$R_{r2} = Copy_r(x, y) \rightarrow Succ(y, z)$
$R_{r3} = Succ(x, y) \wedge Copy_r(x, z) \wedge Succ(z, u) \wedge Value(y, v) \rightarrow Copy_r(y, u) \wedge Value(u, v)$
$R_{r4} = Succ(x, E) \wedge Copy_r(x, y) \wedge Succ(y, z) \rightarrow Value(z, \Box) \wedge Succ(z, E)$.

R_{r2} is *fr1* and the other rules are *rr* (with R_{r4} begin also *fr1*). In the "left-copy" rules, say $R_{l1} \dots R_{l4}$, $Copy_l$ is used in an obvious way instead of $Copy_r$, with B replacing E. $\mathcal{R}$ contains these eight copy rules and one rule $R(\delta)$ per transition δ. Finally, Q encodes the fact that the head is in the final state: $Q = \{Head(x, T_f)\}$, where T_f is the final state.

The proof relies on the following equivalence: there is a derivation of F that contains a "*Succ*-path" from B to E encoding a word $\Box^* m' \Box^*$, with $Head(x, T)$ and $Value(x, m'_i)$, where x is a variable at a "position" k (with 0 being the position of the variable containing the beginning of m') *iff* there is a sequence of transitions of $\mathcal{M}$ from the initial configuration to a configuration with m' on the tape, the head pointing to a cell containing m'_i at a position k (with 0 being the position of the cell containing the beginning of m') and with control state T. The $\Rightarrow$ direction of this equivalence is proven by induction on the length of a derivation sequence. The $\Leftarrow$ direction is proven by induction on the number of transition applications.

□