Complexity Boundaries for Generalized Guarded Existential Rules

Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, Michaël Thomazo

Complexity Boundaries for Generalized Guarded Existential Rules

Jean-François Baget
INRIA, France
baget@lirmm.fr

Marie-Laure Mugnier
University Montpellier II, France
mugnier@lirmm.fr

Sebastian Rudolph
KIT, Germany
sebastian.rudolph@kit.edu

Michaël Thomazo
University Montpellier II, France
thomazo@lirmm.fr

Research Report LIRMM (RR-11006), February 2011

Abstract

In this report, we establish complexities of the conjunctive query entailment problem for classes of existential rules (also called Tuple-Generating Dependencies or Datalog+/- rules). Our contribution is twofold. First, we introduce the class of *greedy bounded treewidth sets (gbts)* of rules, which covers guarded rules, and their known generalizations, namely (weakly) frontier-guarded rules. We provide a generic algorithm for query entailment with *gbts*, which is worst-case optimal for combined complexity with bounded predicate arity, as well as for data complexity. Secondly, we classify several *gbts* classes, whose complexity was unknown, namely frontier-one, frontier-guarded and weakly frontier-guarded rules, with respect to combined complexity (with both unbounded and bounded predicate arity) and data complexity.

1 Introduction

First-order Horn rules (without function symbols except constants) have long been used in artificial intelligence, as well as in databases under name Datalog. We consider here an extension of these rules that allows to create existentially quantified variables (ability called value invention in databases [AHV94]). More precisely, these extended rules are of the form $Body \rightarrow Head$, where $Body$ and $Head$ are conjunctions of atoms, and variables occurring only in the $Head$ are existentially quantified. E.g., $\forall x(\mathit{Human}(x) \rightarrow \exists y(\mathit{HasParent}(x, y) \wedge \mathit{Human}(x)))$. Such rules

are known in databases as Tuple-Generating Dependencies (TGDs) [BV84] and have been extensively used, e.g. for data exchange [FKMP05]. Recently, the corresponding logical fragment has gained new interest in the context of ontological knowledge representation. It has been introduced as the Datalog+/- framework in [CGK08, CGL09, CGL$^+$10], and independently, stemming from graph-based knowledge representation formalisms [CM09], as $\forall\exists$-rules [BLMS09, BLM10]. This rule-based framework is particularly well-suited to the topical ontological query answering problem, which consists of querying data while taking ontological knowledge into account. In particular, it generalizes the core of new description logics (DL) tailored for conjunctive query answering [CGL$^+$07, LTW09, BLMS08, CGL09, BLM10][1]. Moreover, in the case the DL-Lite family [CGL$^+$07], it has been shown that this covering by a Datalog+/- fragment is done without increasing complexity [CGL09].

The ability to generate existential variables, associated with arbitrarily complex conjunctions of atoms, makes entailment with these rules undecidable [BV81, CLM81]. Since the birth of TGD, and recently within the Datalog+/- and $\forall\exists$-rule frameworks, various conditions of decidability have been exhibited. Three "abstract" classes have been introduced in [BLM10] to describe known decidable behaviours: an obvious condition of decidability is the finiteness of the forward chaining (known as the *chase* in the TGD framework [JK84]); sets of rules ensuring this condition are called *finite expansion sets* (*fes*); a more general condition introduced in [CGK08] accepts infinite forward chaining provided that the facts generated have a bounded treewidth (when seen as graphs); such sets of rules are called bounded treewidth sets (*bts*); then decidability follows from the decidability of first-order logic (FOL) classes with the bounded treewidth model property [Cou90]. The third condition, giving rise to *finite unification sets*, relies on the finiteness of (a kind of) backward chaining mechanism. None of these three abstract classes is recognizable [BLM10].

In this paper, we focus on the *bts* paradigm and its main "concrete" classes. (Pure) *Datalog* rules (*i.e.* without existential variables) are *fes* (thus *bts*). *Guarded* rules [CGK08] are inspired by the guarded fragment of FOL. Their body has an atom (the guard) that contains all variables from the body. Guarded rules are *bts* (and not *fes*), they are generalized by *weakly guarded rules* (wg), in which the guarding condition is relaxed: only "affected" variables need to be guarded; intuitively, affected variables are variables that are possibly mapped, during the forward chaining process, to newly created variables [CGK08]. *wg*-rules include Datalog

[1] The DL constructor called existential restriction ($\exists R.C$) is fundamental in these DL. The logical encoding of an membership that contains it in its right-part requires an existentially quantified variable in the corresponding rule head. For instance, the above rule example can be seen as the logical translation of the DL inclusion $Human \sqsubseteq HasParent.Human$.

rules (in which there are no affected variables). Other decidable classes rely on the notion of the *frontier* of a rule (the set of variables shared between the body and the head of a rule). In a *frontier-one* rule *(fr1)*, the frontier is restricted to a single variable [BLMS09]. In a frontier-guarded rule *(fg)*, an atom in the body guards the frontier [BLM10]. Hence, *fg*-rules generalize both *guarded* rules and *fr1*-rules. When only affected variables from the frontier need to be guarded, we obtain the still decidable class of *weakly frontier guarded rules (wfg)*, which generalizes both *fg* and *wg* classes [BLM10]. Of all known recognizable *bts* classes, *wfg* is the class subsuming the most of the others.

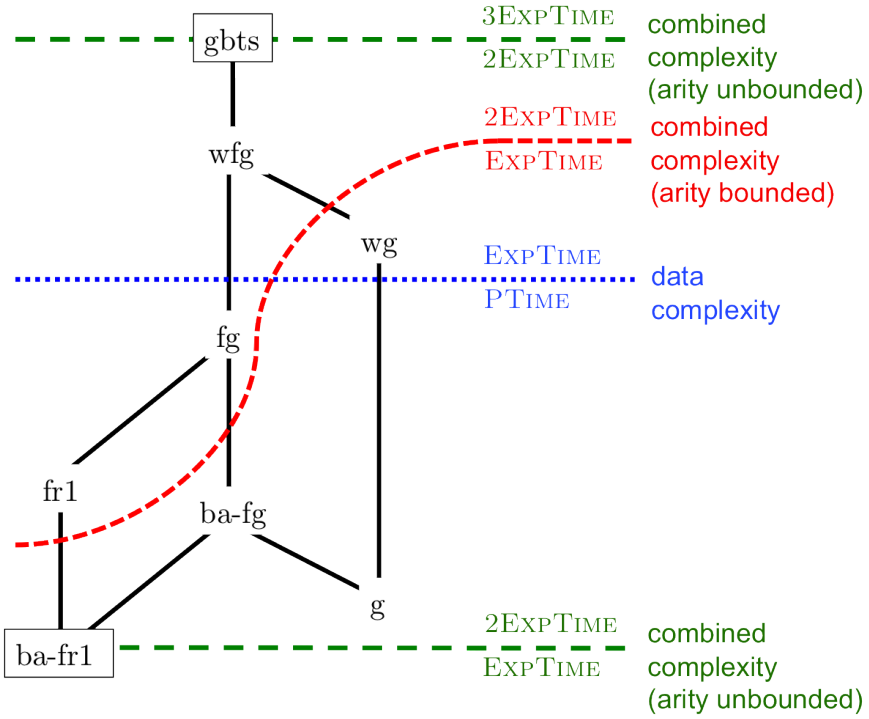

Figure 1: Complexity Boundaries. Tight bounds for gbts and ba-fr1 are yet unknown, we conjecture 3ExpTime-completeness and ExpTime-completeness, respectively.

Example 1 (with the usual simplified syntax for rules)
$\mathit{HasParent}(x, y), \mathit{HasParent}(y, z) \rightarrow \mathit{isGdParent}(z)$
is Datalog and *fr1* but not guarded;
$\mathit{WorksOn}(x, z), \mathit{WorksOn}(y, z), \mathit{StudentTandem}(x, y) \rightarrow \mathit{Grade}(x, t), \mathit{Grade}(y, t)$
is *fg*, but is neither *fr1*, nor guarded, nor Datalog.

Contrarily to *fes* and *fus*, the definition of *bts* does not provide a constructive entailment procedure. Some of its subclasses, namely guarded and *wg*, are provided with an algorithm and their complexity is known [CGK08, CGL09]. However, this is not the case for the *fr1*, *fg* and *wfg* classes. The aim of this paper is to solve these algorithmic and complexity issues.

Our contribution is twofold. First, by imposing a restriction on the allowed derivation sequences, we define a subclass of *bts*, namely greedy *bts (gbts)*, which has the nice property of covering the *wfg* class (thus all *bts* classes cited above). *gbts* are defined by a very simple condition: when such a set is processed in forward chaining, any application of a rule R from this set maps the whole frontier of R to terms belonging to the initial facts or to the atoms added by a *single* previous rule application. The fundamental property fulfilled by *gbts* is that to any derivation is naturally associated a bounded treewidth decomposition of the derived facts, which can be built in a greedy manner. We provide a generic algorithm for this class, which is worst-case optimal for data complexity, as well as for combined complexity in the case where predicate arity is bounded. Secondly, we classify the *wfg*, *fg* and *fr1* classes with respect to both combined (with and without bound on the predicate arity) and data complexities. We also consider the case of rules with an acyclic (more precisely, hypergraph-acyclic) body and point that body-acyclic fg-rules coincide with guarded rules from an expressivity and complexity perspective.

FIG. 1 shows the complexity lines for these classes of rules with three complexity measures, i.e., combined complexity without or with bound on the predicate arity, and data complexity. Notice that data complexity and bounded-arity combined complexity are not strictly layered. While *fg*-rules are much easier for data complexity (PTIME) than for bounded-arity combined complexity (2EXPTIME), wg-rules are in EXPTIME for both. Precise complexity results obtained are given in TAB. 1. New results are indicated by a star.

2 Preliminaries

As usual, an *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate with arity k, and the t_i are terms, *i.e.* variables or constants. A *conjunct* $C[\mathbf{x}]$ is a finite conjunction of atoms, where $\mathbf{x}$ is the set of variables occurring in C. A *fact* is the existential closure of a conjunct. A (boolean) *conjunctive query* (CQ) has the same form as a fact, thus we identify both notions. We also see conjuncts, facts and CQ as sets of atoms. Given an atom or a set of atoms A, we denote by $vars(A)$ and $terms(A)$ its set of variables and of terms, respectively. Given conjuncts F and Q, a *homomorphism* π from Q to F is a substitution of $vars(Q)$ by terms of F such

Class	arity unbounded	arity bounded	Data Complexity
gbts	in 3EXPTIME *	2EXPTIME-c *	EXPTIME-c*
wfg	2EXPTIME-c *	2EXPTIME-c *	EXPTIME-c*
fg	2EXPTIME-c *	2EXPTIME-c *	PTIME-c *
fr1	2EXPTIME-c *	2EXPTIME-c *	PTIME-c *
wg	2EXPTIME-c	EXPTIME-c	EXPTIME-c
guarded	2EXPTIME-c	EXPTIME-c	PTIME-c
ba-fg	2EXPTIME-c *	EXPTIME-c *	PTIME-c *
ba-fr1	EXPTIME-hard $^{\star(1)}$	EXPTIME-c *	PTIME-c *

(1) EXPTIME-c if no constants in rules

Table 1: Combined and Data Complexities

that $\pi(Q) \subseteq F$ (we say that π maps Q to F). It is well-known that, given two facts F and Q, $F \models Q$ iff there is a homomorphism from Q to F.

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

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

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

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

A *knowledge base* (KB) $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts (seen as a single fact) F and a finite set of rules $\mathcal{R}$. We denote by $\mathcal{C}$ the set of constants occurring in $(F, \mathcal{R})$ and by $\mathcal{P}$ the set of predicates occurring in $\mathcal{R}$. The *(boolean) CQ entailment* problem is the following: given a KB $\mathcal{K} = (F, \mathcal{R})$ and a (boolean) CQ Q, does $F, \mathcal{R} \models Q$ hold ?

We now specify some already introduced notions. 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 and there is an edge between two nodes if they belong to the same hyperedge. The treewidth of a fact is defined as the treewidth[2] of its associated primal graph. 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*$(F') \leq b$. A rule R is guarded if there is an atom $a \in body(R)$ (called a guard) with $vars(body(R)) \subseteq vars(a)$. R is weakly guarded *(wg)* if there is $a \in body(R)$ (called a weak guard) that contains all affected variables from $body(R)$. The notion of affected variable is relative to the rule set: a variable is affected if it occurs only in affected predicate positions, which are positions that may contain a new variable generated by forward chaining (see [FKMP05] for a precise definition). The important point is that a rule application necessarily maps non-affected variables to terms from the initial fact. R is frontier-one *(fr1)* if $|fr(R)| = 1$. R is frontier-guarded *(fg)* if there is $a \in body(R)$ with $vars(fr(R)) \subseteq vars(a)$. R is weakly-frontier guarded *(wfg)* if there is $a \in body(R)$ that contains all affected variables from $fr(R)$.

3 Greedy Bounded-Treewidth Sets of Rules

In a greedy derivation, every rule application maps the frontier of the rule into terms added by a *single* previous rule application or occurring in the initial fact:

Definition 4 (Greedy Derivation) *An $\mathcal{R}$-derivation $(F_0 = F), \ldots, F_k$ is said to be* greedy *if, for all i with $0 \leq i < k$, there is $j < i$ such that $\pi_i(\mathsf{fr}(R_i)) \subseteq \mathsf{vars}(A_j) \cup \mathsf{vars}(F_0) \cup \mathcal{C}$, where $A_j = \pi_j^{safe}(\mathsf{head}(R_j))$.*

Example 2. Let $\mathcal{R} = \{R_0, R_1\}$ where:
$R_0 = r_1(x, y) \rightarrow r_2(y, z)$ and
$R_1 = r_1(x, y), r_2(x, z), r_2(y, t) \rightarrow r_2(z, t)$.
Let $F_0 = \{r_1(a, b) \wedge r_1(b, c)\}$ and $S = F_0, \ldots, F_3$ with:
$F_1 = \alpha(F_0, R_0, \{(y, b)\})$, $A_0 = \{r_2(b, x_1)\}$,
$F_2 = \alpha(F_1, R_0, \{(y, c)\})$, $A_1 = \{r_2(c, x_2)\}$,
$F_3 = \alpha(F_2, R_1, \pi_2)$, with $\pi_2 = \{(z, x_1), (t, x_2)\}$;
there is no A_j s.t. $\{\pi_2(z), \pi_2(t)\} \subseteq terms(A_j)$, thus S is not greedy.

To any greedy derivation S of F can be assigned a unique *derivation tree* $DT(S)$ built iteratively as follows: the root is a node x_0 with $\mathsf{terms}(x_0) = \mathsf{vars}(F) \cup$

[2]We assume that the reader is familiar with this notion.

$\mathcal{C}$ and $\mathit{atoms}(x_0) = \mathit{atoms}(F)$, and $\forall 0 \leq i < k$, we add a node x_{i+1} with $\mathit{terms}(x_{i+1}) = \mathit{vars}(A_{i+1}) \cup \mathit{vars}(F) \cup \mathcal{C}$ and $\mathit{atoms}(x_{i+1}) = \mathit{atoms}(A_{i+1})$. Since S is greedy, there is at least a j such that $\pi_i(\mathit{fr}(R_i)) \subseteq \mathit{terms}(x_j)$. We add an edge between x_{i+1} and $x_{j'}$, j' being the smallest integer such that $x_{j'}$ has this property. The nodes of $\mathit{DT}(S)$ are also called "bags".

Property 2 *Let $S = F_0 \ldots, F_k$ be a greedy derivation. Then $\mathit{DT}(S)$ is a tree decomposition of F_k of width bounded by $|vars(F)|+|\mathcal{C}|+max(|vars(head(R))|_{R\in\mathcal{R}})$.*

Definition 5 (greedy bounded-treewidth set of rules (gbts)) *$\mathcal{R}$ is said to be a greedy bounded-treewidth set (gbts) if (for any fact F) any $\mathcal{R}$-derivation (of F) is greedy.*

The class *gbts* is a strict subclass of *bts* (e.g. in Example 2: $\mathcal{R}$ is *fes* but not *gbts*). It is nevertheless an expressive subclass of *bts* since it contains *wfg*:

Property 3 *wfg-rules are gbts.*

Proof: Let R (from a *wfg* rule set $\mathcal{R}$) with weak frontier guard g. Assume R is applied by π and let $\pi(g) = a$. Either $a \in F$ or $a \in A_i$ for some i. Thus $\mathcal{R}$ is gbts. □

The following example shows that *gbts* strictly contains *wfg*:
$R = r_1(x, y), r_2(y, z) \rightarrow r(x, x'), r(y, y'), r(z, z'), r_1(x', y'), r_2(y', z')$;
$\{R\}$ is *gbts*, but not *wfg* (nor *fes*).

4 An Algorithm for gbts

The intuition underlying the decidability of gbts is that rule applications create a bounded number of "relevant patterns". When we have created all possible patterns, "large-enough" to map Q, we can halt the derivation process. In [CGL09], a specific notion of *type* is used for that purpose. However, to take non-guarded rules into account, we need to generalize it. We thus define the ultimate applicability of a rule, and the related notion of oracle. Sets of oracles generalize types. To simplify the presentation, we translate Q into a rule $R_Q = Q \rightarrow \mathit{match}$ where *match* is a fresh nullary predicate (note that $fr(R_Q)$ is empty). The question is now whether $\mathcal{R} \cup \{R_Q\} \models \mathit{match}$.

Definition 6 (Ultimate applicability, oracle) *Let F be a fact, $\mathcal{R}$ be a set of gbts rules, and $S = F_0(= F), \ldots, F_{i+1}$ be an $\mathcal{R}$-derivation of F (with $F_{j+1} = F_j \cup A_j, \forall 0 \leq j \leq i$), and let x_j be the bag of $DT(S)$ associated with A_j. We say that $R \in \mathcal{R}$ is* ultimately applicable *to x_j if there is an $\mathcal{R}$-derivation*

$F_0(=F), \ldots, F_j, \ldots, F_{i+1}, \ldots, F_l$ *with a homomorphism* π *from* $\mathsf{body}(R)$ *to* F_l *and* $\pi(\mathsf{fr}(R)) \subseteq \mathsf{terms}(x_j) = \mathsf{vars}(A_j) \cup \mathsf{vars}(F) \cup \mathcal{C}$*. We say that* $\pi_{|fr(R)}$ *is an* oracle *for the ultimate applicability of* R *on* x_j*. An* ultimate $\mathcal{R}$-derivation *is a sequence* $F_0, \ldots, F_k$ *where* $\forall 0 \leq i < k$*, there is* $R \in \mathcal{R}$ *and an oracle* $\pi_{|fr(R)}$ *for the ultimate applicability of* R *on some* x_j *with* $j < i$ *such that* $F_{i+1} = \alpha(F_i, R, \pi_{|fr(R)})$*.*

Any derivation is an ultimate derivation and an ultimate derivation $F_0, F_1, \ldots, F_k$ can always be extended to a derivation $F_0, F_0^1, \ldots, F_0^{i_1}, F'_1, \ldots, F'_{k-1}, F_{k-1}^1, \ldots, F_{k-1}^{i_k}, F'_k$, where F'_i contains F_i.

We now define an equivalence relation $\sim_Q$ on the bags of a derivation tree with the following informal meaning: $x_i \sim_Q x_j$ means that a rule body can be ultimately mapped on x_i iff if can be mapped similarly on x_j.

Definition 7 ($\sim_Q$) *Let* T *be a derivation tree,* x *and* y *two bags of* T*.* $x \sim_Q y$ *iff there is a bijective substitution* ψ *of* $\mathsf{terms}(x)$ *by* $\mathsf{terms}(y)$ *s.t.* $\forall R \in \mathcal{R}$*,* π *is an oracle for* R *on* x *iff* $\psi \circ \pi$ *is an oracle for* R *on* y*.*

Algorithm 1 behaves as a classical breadth-first forward chaining (also called "chase" in databases) with two main differences. First, instead of looking for homomorphisms to check the applicability of rules, it uses oracles for ultimate applicability, thus building an ultimate derivation tree. Secondly, $\sim_Q$ allows to prune the ultimate derivation tree.[3]

Theorem 4 $\sim_Q$ *Ultimate Saturation is sound and complete for CQ entailment with* $\mathcal{R}$ *gbts.*

Proof: Should we not take into account the $\sim_Q$ equivalence relation, soundness and completeness of the algorithm would immediately follow from the correspondence between rule application and ultimate rule application. However, $\sim_Q$ is used to cut whole subtrees of the derivation tree. To prove that this partial exploration does not prevent completeness, we have to prove the following property: "if $x \sim_Q y$ and x parent of x', then there is a child y' of y such that $x' \sim_Q y'$". Thus exploring only one subtree of two $\sim_Q$ equivalent bags will be sufficient to preserve the completeness of the algorithm.

We first build $y' = \mathsf{copy}_{[x \leftarrow y]}(x')$ the bag obtained from y "as x' is obtained from x". The bag x' was obtained by ultimately applying some rule R according to some oracle π, that mapped $\mathsf{fr}(R)$ to terms from $\mathsf{atoms}(x)$. Thus, since $x \sim_Q y$, there exists an oracle $\psi \circ \pi$ that maps $\mathsf{fr}(R)$ to terms from $\mathsf{atoms}(y)$. The bag y' thus

[3] This is necessary to guarantee termination, resembling the *blocking* techniques applied in DL tableaux algorithms.

Algorithm 1: $\sim_Q$ Ultimate Saturation

Data: Two facts F, Q, gbts rule set $\mathcal{R}$
Result: YES if $F, \mathcal{R} \models Q$, NO otherwise.
$\mathcal{R} \leftarrow \mathcal{R} \cup \{R_Q\}$;
$T \leftarrow$ newTree(x_0) ; // x_0 is the root
terms$(x_0) \leftarrow$ *vars*$(F) \cup \mathcal{C}$;
Continue $\leftarrow$ True; Depth $\leftarrow 0$; // depth of T
while *Continue* **do**
  Continue $\leftarrow$ False;
  for $x \in$ *leavesAtDepth(T, Depth)* **do**
    for $R \in \mathcal{R}$ **do**
      for $\pi \in$*oracles*$(R, x) \backslash$*oracles*$(R,$ *parent*$(x))$ **do**
        if $R = R_Q$ **then**
          return YES;
        $y \leftarrow$ newNode();
        terms$(y) \leftarrow$ *vars*$(\pi^{safe}($*head*$(R))) \cup$ *vars*$(F) \cup \mathcal{C}$;
        if $\neg \exists z \in$ *nodes*(T) *such that* $y \sim_Q z$ **then**
          addEdge$(T, (x, y))$;
          Continue $\leftarrow$ True;
  Depth++;
return NO;

corresponds to that ultimate rule application, and its atoms are built accordingly (precisely, *atoms*$(y') = (\psi \circ \pi)^{safe}($*head*$(R))$).

It remains now to prove that $x' \sim_Q y'$ *i.e.,* to prove the following property P: "there exists a bijection ψ' from *terms*(x') to *terms*(y') such that π is an oracle for R in x' iff $\psi' \circ \pi$ is an oracle for R in y'". The bijective substitution ψ' is built as follows: for any term t of x' that is already in x, $\psi'(t) = \psi(t)$. Otherwise, we define $\psi'(t) = t'$ where t and t' have been safely rewritten from the same term in *head*(R).

We first point out that it is immediate to see that $x'' = \textit{copy}_{[y \leftarrow x]}(y') \sim_Q x'$. Indeed, x' and x'' are two bags created by the same ultimate rule application, the second one creating only redundancy. They have thus equivalent oracles (*i.e.,* their sets of oracles are equals up to the natural bijection from *terms*(x') to *terms*(x'')). Should we be able to prove the first direction of property P, the second would then immediately follow: if π is an oracle on y', then (first direction of property P), there is an oracle π' on x'' that is equivalent to an oracle π'' on x'.

It remains only to check that if π is an oracle for some R on x', then $\psi' \circ \pi$ is also an oracle on y'. In order to do that, we generalize the notion of *copy*: instead of copying the child of a bag x under a $\sim_Q$ equivalent node y, we *copy* a whole

subtree of x under y. The definition of the *copy* of a subtree immediately follows from the copy of a unique bag, by means of a trivial recursion.

Then we point out that if π is an oracle for R on x', then there is a subtree rooted in x' such that *body*(R) can be mapped (using a homomorphism π' that extends π) to the fact associated with a minimal derivation generating that subtree. Let us consider a minimal such subtree. Finally, we show that *body*(R) can also be mapped to the fact associated with the *copy* of that subtree under the equivalent node y, and that $\psi' \circ \pi$ is an oracle of R in y'. The proof is made by induction on the depth of that subtree. □

Let us know study the complexity of Algorithm 1. If we denote by T_{max} the maximum number of non-equivalent nodes that can be generated, the asymptotic complexity of Algorithm 1 is $T_{max} \times |\mathcal{R}| \times$ [*(cost of a call to oracles* (R, x)*)* + $|$*oracles*$(R, x)| \times$ *(cost of checking the existence of a* $\sim_Q$ *equivalent node*)].

Property 5 *Let $\mathcal{R}$ be gbts, Q a query, F a fact. Let $q = \max_i(|$terms(body$(R_i))|)$, $b = |$terms$(F)| + \max_i(|$terms(head$(R_i))|) + |\mathcal{C}|$ and w the maximum predicate arity. Let S be an ultimate derivation of F, x a bag of DT(S). The cost of a call to* oracles(R, x) *is in the order of* poly$(b^b, 2^{b^{q+1}2^{|\mathcal{P}|\cdot q^{w+1}}})$.

Proof: See Section 4.1. □

Checking $y \sim_Q z$ has a cost $\mathcal{O}(b^b \times (|\mathcal{R}| \times q^b)^2)$ and T_{max} is upper-bounded by $2^{|\mathcal{R}| \times b^q}$, hence the following theorem (with hardness results stemming from *wfg* subclass):

Theorem 6 *CQ entailment is in* 3EXPTIME *for combined complexity,* 2EXPTIME*-complete when the arity of the predicates is bounded and* EXPTIME*-complete for data complexity.*

4.1 Computing the set of oracles of a bag

In this section, we show how to compute the set of oracles associated with a bag of $DT(S)$, where S is an ultimate derivation, and evaluate the cost of this computation.

Notations: Given a derivation (or ultimate derivation) S of F, we note $S(F)$ the last fact of S. By an *extension* S' of $S = (F_0, \ldots, F_k)$, we mean a derivation with *prefix* S, i.e., $S' = (F_0, \ldots, F_k, \ldots, F')$. The extension of $S = (F_0, \ldots, F_k)$ with $(R, \pi|_{fr(R)})$, where R is a rule and $\pi|_{fr(R)}$ maps $fr(R)$ to F_k is $S' = (F_0, \ldots, F_k, \alpha(F_k, R, \pi|_{fr(R)}))$. In the following, F denotes the initial fact and $\mathcal{R}$ is a gbts set of rules.

4.1.1 Alternative derivation tree, alternative (ultimate) patterns and oracles

We define an alternative derivation tree in order to ensure that patterns (which are defined below) are in finite number. Instead of encoding $A_i = \pi_i^{safe}(head(R_i))$ in a bag x_{i+1}, we encode $head(R_i)$ in x_{i+1}, i.e., $terms(x_{i+1}) = \textsf{terms}(\textsf{head}(R_i)) \cup \textsf{vars}(F) \cup \mathcal{C}$. Then, to keep track of $\pi|_{fr(R_i)}$, we label the edge between x_{i+1} and its parent. For convenience, we also define a mapping $\Psi_{x_{i+1}}$, which assigns to each term in $\textsf{terms}(x_{i+1})$ its corresponding term in F_{i+1}.

Definition 8 (Alternative derivation tree) *Let $S = (F_0, \ldots, F_k)$ be an $\mathcal{R}$-derivation of F. We build the alternative derivation tree of S (notation $ADT(S)$) in the following way. The root of the tree is a node x_0 such that $\textsf{terms}(x_0) = \textsf{vars}(F) \cup \mathcal{C}$. We define Ψ_{x_0} as the identity on $terms(x_0)$. For all $i \in \{0, \ldots, k-1\}$, we add a node x_{i+1} such that $\textsf{terms}(x_{i+1}) = \textsf{terms}(\textsf{head}(R_i)) \cup \textsf{vars}(F) \cup \mathcal{C}$. Since S is greedy, there is at least one node x_j such that $\Psi_{x_j}(\textsf{terms}(x_j))$ contains $\pi_i(\textsf{fr}(R_i))$. There is even a smallest j' such that $x_{j'}$ has that property. We add an edge between x_{i+1} and $x_{j'}$, which is labeled by a substitution j_{i+1} of $\textsf{fr}(R_i) \cup \textsf{vars}(F) \cup \mathcal{C}$ by $\textsf{terms}(x_j)$ such that, for all $x \in fr(R_i)$, $\Psi_{x_{j'}} \circ j_{i+1}(x) = \pi_i(x)$ and j_{i+1} is the identity on the other terms. Finally, we define $\Psi_{x_{i+1}} : terms(x_{i+1}) \rightarrow terms(F_{i+1})$, with $\Psi_{x_{i+1}}(t) = \Psi_{x_{j'}}(j_{i+1}(t))$ if $t \in \textsf{domain}(j_{i+1})$ and the safe substitution operated to add A_i to F_i otherwise.*

The alternative pattern of a bag describes the different ways facts with at most n terms can be mapped using some terms of the bag.

Definition 9 (Alternative pattern of a bag) *Let $S = (F = F_0, \ldots, F_k)$ be an $\mathcal{R}$-derivation of F. Let B be a bag of $ADT(S)$. The* alternative[4] n-pattern *of B in S is the set of all pairs (G_j, π_j) such that:*

- *G_j is a conjunction with at most n terms;*
- *π_j is a substitution of some terms from G_j by terms from B such that $\Psi_B \circ \pi_j$ is extendable to a homomorphism from G_j to F_k.*

We are moreover interested in what happens later on in the forward chaining process, and that is why we define ultimate patterns.

Definition 10 (Alternative ultimate pattern of a bag) *Let $S = (F = F_0, \ldots, F_k)$ be an $\mathcal{R}$-derivation of F. Let B be a bag of $ADT(S)$. The* alternative ultimate n-pattern *of B in S in the union of all the alternative n-patterns of B in S', for all extensions S' of S.*

[4] We call them alternative patterns since we could similarly define patterns on the bags of a derivation tree.

From now, we consider that $n = max_i(|body(R_i)|)$ (i.e., $n = q$ in Property 5).

If we are able to compute ultimate patterns, we can compute the set of oracles for a given bag:

Property 7 (Correspondence between oracles and alternative patterns) *Let S be an $\mathcal{R}$-derivation of F. Let B a bag in $DT(S)$, and B' the corresponding bag in $ADT(S)$. Given $R \in \mathcal{R}$, $\pi : \mathsf{fr}(R) \to \mathsf{terms}(B)$ is an oracle for the ultimate applicability of R on B iff there exists $\pi' : \mathsf{fr}(R) \to \mathsf{terms}(B')$ such that $(\mathsf{body}(R), \pi')$ belongs to the alternative ultimate pattern of B' and $\pi = \Psi_{B'} \circ \pi'$.*

Proof: Direct consequences from definitions. □

In the following, we omit the qualifier "alternative" for patterns since all patterns considered are alternative patterns. We keep the distinction between a derivation tree and an alternative derivation tree.

4.1.2 Computation of (alternative) ultimate patterns

First, we show how to compute the ultimate pattern of the root of the alternative derivation tree. Then, we show that the ultimate pattern of any bag can be computed in polynomial time (with respect to the number of patterns and the number of junctions, defined hereafter) as long as the ultimate pattern of its parent is known. In both computations, we rely on a translation of our problem into a rewriting system on terms, which are trees (those terms have thus nothing to do with terms in Q or in the KB). Those trees represent derivation trees: nodes are intuitively the same thing as bags. They are labeled by patterns. Edges are labeled by "junctions" which are equal to the j mappings in an ADT, i.e., represent how terms are shared between the bags which are abstracted into nodes. The rules of the original problem are translated into creation rules, and we add update rules in order to maintain the property that the label of a node is the pattern of the corresponding bag in the derivation tree.

This translation being done, determining the ultimate pattern of a bag with pattern p boils down to the following question: starting from a tree t consisting of a single node labeled by p, what is the maximal label (given a natural order on patterns, i.e., inclusion) that can label the root of a term rewritten from t? Put this way, it does not seem easier to determine the ultimate pattern of a bag. To overcome this difficulty, we close our set of rewriting rules in such a way that if p can evolve into p' after an arbitrary number of rule applications with the initial set of rules, then there is a rule $p \to p'$ in the closed set. Then, the problem of determining the ultimate pattern of a bag of pattern p is reduced to the problem of searching the maximal p' such that there exists a rule $p \to p'$ in the closed set of rules.

Definition 11 ((J, P)-rewriting system) *A (J, P)-rewriting system (or, when not ambiguous, a rewriting system) is a set of rules of the following forms:*

- $p_i \rightarrow j(p_i, p_j)$
- $p_i \rightarrow p'_i$
- $j(p_i, p_j) \rightarrow j(p'_i, p_j)$
- $j(p_i, p_j) \rightarrow j(p_i, p'_j)$

with $p_i, p_j, p'_i, p'_j \in P$ and $j \in J$. Rules of the first kind are called creation rules*. The other rules are* update rules*.*

A creation rule creates a new node labeled by p_j, and linked by an edge labeled j to an existing node labeled by p_i. The other rules update the label of an existing node.

We now define a rewriting system that meets our needs. Let us first notice that in the alternative derivation tree, all terms are included in $\mathcal{B} = \mathit{vars}(F) \cup \mathcal{C} \cup \cup_i(\mathit{vars}(\mathit{head}(R_i)))$. Thus, we can define every pattern as a set of pairs (G, π), with π being a substitution of a subset of $\mathit{terms}(G)$ by $\mathit{terms}(\mathcal{B})$. We denote by P the set of all such pairs. We also define the notion of junction:

Definition 12 (Junction) *A junction j is a mapping from a subset of $\mathcal{B}$ containing $\mathit{vars}(F) \cup \mathcal{C}$ to $\mathcal{B}$ such that $j(c) = c$ if $c \in \mathit{vars}(F) \cup \mathcal{C}$.*

Rewriting systems act upon terms, whose definition is below.

Definition 13 (Term) *A term is a tree whose nodes are labeled by a pattern and edges by junctions.*

We aim at defining a rewriting system that builds the *tree of patterns* of an alternative derivation tree. Intuitively, this tree of patterns is a term having the same structure as the ADT, with its nodes being labeled by the pattern of the corresponding bag and its edges by the label of the corresponding edge.

Definition 14 (Tree of patterns of an alternative derivation tree) *Let S be a derivation of F, and $\mathit{ADT}(S)$ be its alternative derivation tree. The tree of patterns of $\mathit{ADT}(S)$ is a term t built recursively as follows:*

- *if S is the trivial derivation (restricted to F), then t is a node, labeled by p_F, the pattern (in S) of the bag containing $\mathit{terms}(F) \cup \mathcal{C}$ (i.e., the set of pairs (G, σ) with G being a conjunction over at most n terms and σ being a homomorphism from G to F). This node is said to be associated with the unique bag of $ADT(S)$.*

- *if S is a derivation with m rule applications, let t' be the tree of patterns associated with the $m-1$ first rule applications. Let v be the node associated with the bag B to which we add a child B': we add a child v' to v and associate v' with B'. We label each node of t by the pattern in S of the bag associated with this node, and we label the edge between v and v' by j, where j is the label of the edge between B and B' in $\mathbf{ADT}(S)$.*

Let us first have a look at the result in the alternative derivation tree of the application of (R, π) to $S(F)$. B being the closest bag to the root containing $\pi(fr(R))$, we add a child B' to B. What is the pattern of B'? We split the answer to this question into two parts.

First part. We can first give a lower bound on the pattern of B': it contains at least all pairs (G, σ) where σ is a substitution of some terms from G by terms from B' such that $\Psi_{B'} \circ \sigma$ can be extended to a homomorphism from G to *the atoms that were added at the last step*.

Second part. However, there are possibly homomorphisms that use also atoms and/or terms that were already in $S(F)$. Moreover, the last rule application may have allowed new homomorphisms in other bags. We thus have to update patterns, by using information of the following kind: if B' is a child of B and the pattern of B (resp. B') contains (G, σ) (resp. (G', σ')), then B contains (G'', σ''), whose construction depends on how terms are shared by B and B'; similarly, information can be propagated from B to B'.

Rules for the first part. Each bag should have a corresponding node in the mimicking term. We first define the "minimum" pattern of the newly created bag.

Definition 15 (n-pattern associated with a (C, π)) *Let C be a conjunction. Let π be a substitution of terms from C by terms from C. The n-pattern associated with (C, π) is the set of pairs (G, σ) with G being a conjunction over at most n terms and σ being a homomorphism from G to $\pi(C)$.*

Using this definition, we propose a set of rules allowing to mimic the creation of new bags.

Definition 16 (Creation rules associated with (R, π)) *Let p be a pattern and $R \in \mathcal{R}$. For every $\pi : \mathsf{fr}(R) \to \mathcal{B}$ such that $(\mathsf{body}(R), \pi) \in p$, we define a creation rule: $p \to j(p, p_{\mathsf{head}(R),\pi})$, where $p_{\mathsf{head}(R),\pi}$ is the pattern associated with $(\mathsf{head}(R), \pi')$, where π' is such that $\pi'(x) = \pi'(y)$ if and only if $\pi(x) = \pi(y)$, and j is a mapping with domain $\mathsf{fr}(R) \cup \mathsf{vars}(F) \cup \mathcal{C}$, equal to π on $\mathsf{fr}(R)$ and to the identity otherwise.*

Rules for the second part. However, we are not only interested in the minimal pattern of the generated bag, but in having the exact pattern for every bag in the derivation. Thus, we add now update rules, which will ensure that a node in the term is labeled by the pattern of the corresponding bag in the derivation tree.

We first define a union operation on conjunctions.

Definition 17 ($G(G_1, G_2, \pi_1, \pi_2, j)$) *Let G_1 and G_2 be two conjunctions of atoms, B_1 and B_2 two bags, π_1 a substitution of some terms from G_1 by terms of B_1, π_2 a substitution of some terms from G_2 by terms of B_2, and j a substitution of some terms from B_2 by terms of B_1. We assume w.l.o.g. that terms in G_1 and G_2 are disjoint. We define $G(G_1, G_2, \pi_1, \pi_2, j)$ as the conjunct obtained from the atoms in G_1 and G_2 by merging terms having the same image by π_1 or by $j \circ \pi_2$.*

Notation. Let T be a set of terms, π_1 be a mapping from B_1 to T, and π_2 be a mapping from B_2 to T. If for every $x \in B_1 \cap B_2, \pi_1(x) = \pi_2(x)$, we define $\pi_1 \cup \pi_2(x)$ from $B_1 \cup B_2$ to T such that $\pi_1 \cup \pi_2(x) = \pi_1(x)$ if $x \in B_1$, and $\pi_1 \cup \pi_2(x) = \pi_2(x)$ otherwise.

Definition 18 (Upward / Downward update rule) *To every triple $(p_1, j, p_2) \in P \times J \times P$, we assign an upward update rule:*

$$j(p_1, p_2) \rightarrow j(p_1', p_2)$$

where p_1' is built in the following way:

- $p_1 \subseteq p_1'$
- *for all $(G_1, \pi_1) \in p_1, (G_2, \pi_2) \in p_2$, we add every (G', π'), where G' is a conjunction over at most n terms such that there exists φ homomorphism from G' to $G(G_1, G_2, \pi_1, \pi_2, j)$, and π' is a restriction of $(\pi_1 \cup (j \circ \pi_2)) \circ \varphi$ having a range included in $\mathsf{range}(\pi_1) \cup \mathsf{range}(j)$.*

We also assign to (p_1, j, p_2) a downward update rule:

$$j(p_1, p_2) \rightarrow j(p_1, p_2')$$

where p_2' is defined in a similar way: $p_2 \subseteq p_2'$ and, for all $(G_1, \pi_1) \in p_1, (G_2, \pi_2) \in p_2$, we add every (G', π'), where G' is a conjunction over at most n terms such that there exists φ homomorphism from G' to $G(G_1, G_2, \pi_1, \pi_2, j)$, and π' is a restriction of $(\pi_1' \cup \pi_2) \circ \varphi$ with $j \circ \pi_1' = \pi_1$ and $\mathsf{range}(\pi') \subseteq \mathsf{range}(\pi_1') \cup \mathsf{range}(\pi_2)$.

In the following, we show that $\mathcal{R}$-derivations of a fact F are "equivalent" to some rewritings (canonical rewritings, defined hereafter) of p_F.

Property 8 *Let S be a derivation of F, and $R \in \mathcal{R}$. Let B be a bag of $ADT(S)$, and v_B the corresponding node in the tree of patterns t of $ADT(S)$. R is applicable by some π : body$(R) \rightarrow S(F)$ with range$(\pi_{|fr}) \subseteq \Psi_B($terms$(B))$ iff there exists π' such that $\Psi_B \circ \pi' = \pi_{|\mathbf{fr}(R)}$ and $($body$(R), \pi'_{|fr(R)}) \in$ pattern$(B) =$ label(v_B).*

Moreover, t', obtained from t by applying the creation rule associated with (R, π) as high as possible in t, has the same structure as the tree of patterns of $ADT(S')$, where S' is the extension of S with (R, π).

Proof: By definition of a pattern, $($body$(R), \pi_{|fr(R)})$ belongs to the pattern of B iff $\Psi_B \circ \pi'$ can be extended to a homomorphism from body(R) to F. From the definition of a tree of patterns, label$(v_B) =$ pattern(B). Since in the derivation tree, the new bag is added as a child as close to the root as possible and t has the same structure as $ADT(S)$, we conclude that t' has the same structure as $ADT(S')$. □

In the following, t' built as in the previous property is called the *primitive tree of patterns* of $ADT(S')$.

We now show that, after a rule application leading to add a new bag B' as the child of a bag B in the alternative derivation tree, we can compute the pattern of B' by updating the tree of patterns: after having applied the creation rule to create the node assigned to B' and compute its "primitive pattern", it is sufficient to apply a downward update rule (from the node assigned to B to the node assigned to B').

Property 9 *Let S be a derivation of F and S' be the extension of S with (R, π). Let t be the tree of patterns of $ADT(S)$ and t' be the primitive tree of patterns of $ADT(S')$. Let B' be the bag created by (R, π) and B its parent in $ADT(S')$, let v' and v be the nodes respectively assigned to B' and B in t', and let j be the label of the edge between v and v'. Let p and p' be the respective labels of v and v' in t'. The label of v' in the tree of patterns of $ADT(S')$, i.e., the pattern of B' in S', is p'' obtained by applying the downward update rule $j(p, p') \rightarrow j(p, p'')$.*

Proof: $\Rightarrow$ We first show that the pattern of B' in S' is included in p''. Let (G, ψ) be in the pattern of B'. By definition of a pattern, there is a homomorphism ψ' from G to $S'(F)$ that extends $\Psi_{B'} \circ \psi$. We define a partition $\{G_o, G_n\}$ of G, where G_o contains the atoms that are mapped to $S(F)$. Hence, G_n is the inverse image by ψ' of atoms created at the last step. Thus, $(G_n, \psi_{|G_n})$ belongs to p' and $(G_o, j \circ \psi_{|G_o})$ belongs to p. Thus, $G = G(G_o, G_n, j \circ \psi_{|G_o}, \psi_{|G_n}, j)$. Hence, w.r.t. notations in Definition 18, we have $p_1 = p$, $p_2 = p'$ and $p'_2 = p''$ and we choose $G' = G$,

$\varphi = id$, $\pi_1 = j \circ \psi_{|G_o}$ and $\pi_1 = \psi_{|G_n}$.
$\Leftarrow$ Conversely, let (G, ψ) be in p''. The first possible case is that (G, ψ) belongs to p', and thus it belongs to the pattern of B'. Otherwise, there exists $(G_1, G_2, \psi_1, \psi_2)$ such that G is a conjunction over at most n terms that maps by some φ to $G(G_1, G_2, \psi_1, \psi_2, j)$, with $(G_1, \psi_1) \in p, (G_2, \psi_2) \in p'$, and ψ being a restriction of $(\psi'_1 \cup \psi_2) \circ \varphi$ s.t. $\psi_1 = j \circ \psi'_1$, and having range included in the terms of B'. It remains to note that there is a homomorphism from $G(G_1, G_2, \psi_1, \psi_2, j)$ to $S'(F)$ that extends $\Psi_{B'} \circ (\psi'_1 \cup \psi_2)$, thus $\Psi_{B'} \circ \psi$ is extendable to a homomorphism from G to $S'(F)$. We conclude that (G, ψ) belongs to the pattern of B'.
□

Definition 19 *Let S be a derivation of F and S' be the extension of S with (R, π). Let t be the tree of patterns of $ADT(S)$, and t' be the primitive tree of patterns of $ADT(S')$. The* update *of t' is done in the following way:*

- *apply the downward update rule applicable to the bag newly created,*
- *traverse the tree t' starting from this node and applying all possible update rules for each node.*

Property 10 *Let S be a derivation of F and S' be the extension of S with (R, π). Let t be the tree of patterns of $ADT(S)$, and t' be the primitive tree of patterns of $ADT(S')$. The* update *of t' is the tree of patterns of $ADT(S')$.*

Proof: By induction. The base case is Property 9. The recursive step is done similarly to the proof of Property 9. □

Let us point out that we build the tree of patterns of an alternative derivation tree by using a *canonical* rewriting sequence, as defined below.

Definition 20 (Canonical rewriting sequence) *A rewriting sequence is said to be* canonical *if for all creation rule r associated with (R, π) and applied to a term t:*

- *either r is applied to the root of t,*
- *or, let v_1 be the node to which r is applied, v_2 be its parent and let j be the label of the edge between v_1 and v_2. r should fulfill that* range$(\pi) \not\subseteq$ *domain*(j).

Moreover, between two applications of creation rules, update rules are applied until no further modification is possible.

It is easy to associate with such a rewriting sequence leading to a term t a derivation whose t is the tree of patterns:

Property 11 *With any canonical rewriting sequence r of p_F (the pattern associated with F), we can associate a derivation S of F such that the term obtained from p_F by applying r is the tree of patterns of $\mathbf{ADT}(S)$.*

Proof: Built recursively on the length of the rewriting sequence (to each creation rule corresponds a rule application). □

We order patterns by inclusion.

Definition 21 *Let p be a pattern and t_p be the term restricted to a node labeled by p. There exists p_c^* such that p_c^* is the biggest pattern that can be at the root of a term rewritten from t_p with a canonical rewriting sequence. We call this pattern the* canonical limit *of p.*

Proof: Arguments: finite number of patterns + monotonicity of rewriting rules + confluence. □

From Property 10 and Property 11, we obtain the following property:

Property 12 *Let F be a fact. The ultimate pattern of F is the canonical limit of p_F.*

Proof: Comes directly from the equivalence between derivation sequence and rewriting sequence shown in previous properties. □

Then, if we denote by l the cost of computing the canonical limit of a pattern, the cost of the computation of the ultimate pattern of the root of $ADT(S)$ is $\mathcal{O}(l)$.

Assuming that we know the ultimate pattern of a bag, we can compute the ultimate pattern of its child in a similar fashion:

Property 13 *Let S be a derivation of F. Let B_1 be a bag and B_2 be a child of B_1, created by a rule application (R, π). If p_1 is the ultimate pattern of B_1, then one can compute the ultimate pattern of B_2 in the following way: start from p_2, where p_2 is the pattern associated with $(\mathsf{head}(R), \pi')$, such that $\pi'(x) = \pi'(y)$ iff $\pi(x) = \pi(y)$; apply downward update rules on p_1, getting p_2', then compute the canonical limit of p_2'. This limit is the ultimate pattern of B_2.*

Proof: Since p_1 is the ultimate pattern of B_1, it is useless to update it. The rest of the proof is the same as for Property 12. □

We have seen that we can compute the ultimate pattern of any bag if we are able to compute the canonical limit of any pattern in the rewriting system associated with our set of rules. We focus then on this question in the following. First, we relax the condition on the rewriting sequence, showing that the canonical limit is equal to the limit of a pattern.

Definition 22 *Let p be a pattern. There exists p^* such that p^* is the biggest pattern that can be the root of a term rewritten from p. We call this pattern the limit of p.*

Property 14 *Let $p \in P$. The canonical limit of p is equal to the limit of p.*

Proof: We canonize every rewriting sequence, by induction on the length of the rewriting. First, we associate with the term restricted to the node labeled by p (on which we apply non-canonized rules) another term restricted to a single node labeled by p. Each time we apply a rule in the non-canonized sequence, we apply it to the "correct" node, which is an ancestor of the associated node in the canonized term. We ensure that the label of the associated term is bigger than the label of the initial term.
• For the empty rewriting sequence, that is trivial;
• Let us assume that we have built a canonical rewriting sequence with the desired property. The interesting case is when we apply a creation rule. Let us assume that we apply it on v in the initial term. Let v' be the node associated with v. There are two possibilities: the range of π is not included in the domain of the junction of v' with its parent, we can freely add the new node below v'. Otherwise, we consider its first ancestor fulfilling this condition, and add the new node below it. We show then as in properties 9 and 10 that the conditions on patterns is still fulfilled. □

The only step left it to compute the limit of a pattern. Given a rewriting system $\mathcal{S}$, we define the closure of $\mathcal{S}$, by adding rules which allow to simulate a succession of rules in a single step. Closure operations are presented below:

- add $p_i \to p_i$ for any p_i
- if $p_i \to p_j \in \mathcal{S}$ and $p_j \to p_k \in \mathcal{S}$, then $p_i \to p_k \in \mathcal{S}$
- if $p_i \to \varphi(p_i, p_j) \in \mathcal{S}$ and $p_j \to p_k \in \mathcal{S}$, then $p_i \to \varphi(p_i, p_k) \in \mathcal{S}$
- if $p_i \to \varphi(p_i, p_j) \in \mathcal{S}$ and $p_j \to p'_j \in \mathcal{S}$ and $\varphi(p_i, p'_j) \to \varphi(p'_i, p'_j) \in \mathcal{S}$ then $p_i \to p'_i \in \mathcal{S}$

The closure of a rewriting system is the set of rules obtained by applying closure operations until a fixpoint is reached. It is well defined, since there is a finite number of rules. Moreover, we can compute it in polynomial time in P and J. The key property is the following:

Property 15 *Let $\mathcal{S}$ be a rewriting system. Let p be a pattern, and t_p be the term restricted to a root labeled by p. There exists a term t' derived from t_p with rules from $\mathcal{S}$ whose root is labeled by p' if and only if $p \to p'$ is in the closure of $\mathcal{S}$.*

Proof: $\Rightarrow$ We prove by induction on k that for any $\mathcal{S}$, if t can we derived with rules from $\mathcal{S}$ into a term with root labeled by p' in k steps, then $p \rightarrow p'$ belongs to the k-closure of $\mathcal{S}$.
$\Leftarrow$ We prove by induction on k that for any $\mathcal{S}$, if a rule is added at the k^{th} step of closure, the same local effect can be achieved in at most 3^k operations with the initial rule set. □

Property 16 *The closure of a rewriting system can be computed in polynomial time in the number of patterns and junctions.*

Proof: The cost comes from the computation of the closure. There is a number of rules which is polynomial in $|P|, |J|$. At each step of closure, at least one rule is added, then at most a polynomial number of closure steps are performed. Each step has a polynomial cost in the number of rules, thus the closure can be computed in polynomial time in $|P|, |J|$. □

Corollary 17 *The limit of a pattern can be computed in polynomial time in the number of patterns and junctions.*

Proof of property 5:
We can check that the cardinality of J is at most b^b and the cardinality of P is at most $2^{b^{q+1}2^{|\mathcal{P}|.q^{w+1}}}$. Then, from the previous property, the limit of a pattern, and thus the ultimate pattern of a bag, can be computed in time $poly(b^b, 2^{b^{q+1}2^{|\mathcal{P}|.q^{w+1}}})$. By Property 7, the set of oracles for a bag is thus computed with the same complexity.

5 Weakly Frontier-Guarded Rules

First, the EXPTIME-complete data complexity of wfg-rules directly follows from EXPTIME membership of gbts (SECT. 4) and EXPTIME-hardness of wg-rules [CGK08].

We now prove that w(f)g-rules can be polynomially translated into (f)g-rules. In particular, this allows us to exploit the 2EXPTIME membership result established in the next section for fg-rules. W.l.o.g. we assume here that the initial fact F does not contain any variable. Then, a homomorphism from a rule body to a derived fact necessarily maps non-affected variables to constants in $\mathcal{C}$. Thus, by replacing non-affected variables in rules with all possible constants, we obtain an equivalent set of rules (Section 5.1). However, this partial grounding produces a worst-case exponential blow-up in the number of non-affected variables per rule. We thus provide a way to simulate partial groundings with only polynomial blow-up (Section 5.2). Note however that this second translation does not preserve the predicate arity.

5.1 Translation by Partial Grounding

Given a rule R in a set of rules $\mathcal{R}$, let $\mathsf{nav}(R)$ denote the set of variables that are not affected in R. Let $ground(R)$ denote the set containing all rules obtained by uniformly substituting the non-affected variables of R by constants, i.e., $ground(R) = \{\sigma(R) \mid \sigma : \mathsf{nav}(R) \to \mathcal{C}\}$. Moreover let $ground(\mathcal{R}) = \bigcup_{R\in\mathcal{R}} ground(R)$. Let $s = \max_{R\in\mathcal{R}} |\mathsf{nav}(R)|$ the maximal number of non-affected variables per rule.

Lemma 1 *Let $\mathcal{R}$ be a set of rules, F a fact and Q a query. Then we have that $F, \mathcal{R} \models Q$ if and only if $F, ground(\mathcal{R}) \models Q$.*

Proof: For any $R \in \mathcal{R}$ and $R_i \in ground(R)$, let $h_{R_i} : body(R) \to body(R_i)$ denote the homomorphism naturally associated with the grounding. Given $R \in \mathcal{R}$, let us consider a homomorphism $h : body(R) \to F$. Then, there is $R_i \in ground(R)$ and $h' : body(R_i) \to F$, such that $h = h' \circ h_{R_i}$. Conversely, for any $R_i \in ground(R)$ and any $h' : body(R_i) \to F$, we have a homomorphism $h = h' \circ h_{R_i} : body(R) \to F$. Note that $h(head(R)) = h'(head(R_i))$. It follows that any derivation sequence with $\mathcal{R}$ can be expressed as a derivation sequence with $ground(\mathcal{R})$, and reciprocally. □

The following properties are immediate. Note that for a rule R, the number of rules in $ground(R)$ is equal to $|C|^{|\mathsf{nav}(R)|}$.

Property 18 (Properties of *ground*)

- *If $\mathcal{R}$ is weakly (frontier-) guarded, then $ground(\mathcal{R})$ is (frontier-) guarded.*
- $|ground(\mathcal{R})|$ *is exponential in the number of affected variables* per rule *and polynomial in the size of F (more precisely: the size of $\mathcal{C}$).*
- ground *is arity-preserving.*

5.2 A more clever partial grounding

For convenience, we fix bijections $\#_R : \mathsf{nav}(R) \to [1..|\mathsf{nav}(R)|]$ which for every rule, assign numbers to all the non-affected variables. Now let $v_1, \ldots, v_m$ and $v'_1, \ldots, v'_s$ be variable symbols not used in $\mathcal{R}$, where $m = |C|$ and $s = max(|nav(R)|)_{R\in\mathcal{R}}$.

We now define the function τ, mapping rules from $\mathcal{R}$ to frontier-guarded rules as follows:

$$\begin{aligned}
\tau(\forall x_1 ... x_k (body \rightarrow (\exists y_1 ... y_{k'} head))) &= \forall \tau_R(x_1) ... \tau_R(x_k)\ v_1 ... v_m \\
&\quad (\tau_R(body) \rightarrow (\exists y_1 ... y_{k'} \tau_R(C))) \\
\tau_R(\textstyle\bigwedge_{i \in I} a_i) &= \textstyle\bigwedge_{i \in I} \tau_R(a_i) \\
\tau_R(p(t_1, ..., t_l)) &= p(v_1, ..., v_m, v'_1, ..., v'_s, \tau_R(t_1), ..., \tau_R(t_l)) \\
\tau_R(t) &= \begin{cases} v'_{\#_R(t)} & \text{if } t \in \mathsf{nav}(R) \\ t & \text{otherwise.} \end{cases}
\end{aligned}$$

Note that thereby the arity of all predicates is increased by $m + s$. The first m positions will be used to permanently hold all constants $c_1, \ldots, c_m$ and the next s positions will serve as a pool for special "safe variables" (i.e., non-affected) which will be used for our implicit grounding. Now we let

$$\tau(\mathcal{R}) := \{\tau(R) \mid R \in \mathcal{R}\} \cup \mathcal{S},$$

where $\mathcal{S}$ contains for every predicate p (let its arity be l) from $\mathcal{R}$ the rules $R^p_{c_i \mapsto v'_j}$

$$\begin{aligned}
\forall v_1, \ldots, v_m, v'_1, \ldots, v'_s, x_1, \ldots, x_l \quad & (p(v_1, \ldots, v_m, v'_1, \ldots, v'_s, x_1, \ldots, x_l) \\
& \rightarrow p(v_1, \ldots, v_m, v^*_1, \ldots, v^*_n, x_1, \ldots, x_l))
\end{aligned}$$

where $v^*_j = v_i$ and $v^*_k = v'_k$ for all $k \neq j$. Thereby, the rule $R_{c_i \mapsto v'_j}$ is used to realize the bindings the constant c_i to the safe variable v'_j, which in turn will be used in the above translation to realize the implicit grounding. $\mathcal{S}$ contains $m \cdot s$ rules for every predicate p, i.e., the total size of $\tau(\mathcal{R})$ is polynomially bounded by the combined size of F and $\mathcal{R}$.

Furthermore,

let

$$\tau(F) := \{\tau'(atom) \mid atom \in F\}$$

with

$$\tau'(p(e_1, \ldots, e_l)) = p(c_1, \ldots, c_m, \underbrace{c_1, \ldots, c_1}_{s}, e_1, \ldots, e_l).$$

Note that the choice of c_1 at positions $m+1, \ldots, m+s$ is just an arbitrary one.

Similarly,

$$\tau(Q) := \{\tau'(atom) \mid atom \in Q\}.$$

Analog to the argumentation above, we see that the size of $\tau(F)$ is polynomially bounded by the combined size of F and $\mathcal{R}$.

Let us now sketch the ideas of the translation. Any atom having the form $p(c_1, \ldots, c_m, \ldots, e_1, \ldots, e_l)$ corresponds to a fact $p(e_1, \ldots, e_l)$ originally present

in $\mathcal{K}$ or derived from $\mathcal{K}$. From any such fact, the rules in $\mathcal{S}$ allow to generate all facts of form $p(c_1, \ldots, c_m, c_{i_1}, \ldots, c_{i_s}, e_1, \ldots, e_l)$ for any $i_1, \ldots, i_s$ in $1 \ldots m$. We cannot make τ' create all these facts directly as there would be exponentially many (and the translation would not be polynomial anymore). This will allow to map the non-affected variables added to each rule atom: if an atom $a = p(t_1, \ldots, x, \ldots, t_l) \in body(R)$, with x being the *ith* element in $\mathsf{nav}(R)$ ($\#_R(x) = i$), can be mapped to $b = p(e_1, \ldots, c_j, \ldots, e_l)$, with x being mapped to the constant c_j, then the $\mathcal{S}$ rules allow to generate from b an atom

$$p(c_1, \ldots, c_m, c_{i_1}, \ldots, c_j, \ldots, c_{i_s}, e_1, \ldots, c_j, \ldots, e_l)$$

to which

$$\tau_R(a) = p(v_1, \ldots, v_m, v'_1, \ldots, v'_i, \ldots, v'_s, t_1 \ldots, v'_i, \ldots, t_l)$$

can be mapped.

Lemma 2 *Let F be a fact,$\mathcal{R}$ a set of rules and Q a query. Then we have*

$$F, ground(\mathcal{R}) \models Q \quad \textit{iff} \quad \tau(F), \tau(\mathcal{R}) \models \tau(Q).$$

Proof:

Let us begin with some easily checked remarks:
(1) All atoms initially present in $\tau(F)$ or derived from it are of form $p(c_1, \ldots, c_m, c_{i_1}, \ldots, c_{i_s}, e_1, \ldots, e_l)$, where the c_{i_j} are constants from $\mathcal{C}$.
(2) The set of atoms $\mathcal{S}$-derivable from a fact $p(c_1, \ldots, c_m, \ldots, c_{i_1}, \ldots, c_{i_s}, e_1, \ldots, e_l)$ is exactly the set of atoms $p(c_1, \ldots, c_m, c_{i'_1}, \ldots, c_{i'_s}, e_1, \ldots, e_l)$, where the $c_{i'_j}$ are constants from $\mathcal{C}$, i.e., $(c_{i_1}, \ldots, c_{i_s})$ can be replaced by any s-tuple of constants.
(3) Let F' be a fact, with $F' = F$ or is derived from F. Any homomorphism $h : Q \to F'$ can be trivially extended to a homomorphism $h' : \tau(Q) \to \tau(F')$, with for all variable x in Q, $h'(x) = h(x)$. Reciprocally, any homomorphism $h' : \tau(Q) \to \tau(F')$ can be restricted to a homomorphism $h : Q \to F'$ keeping the images of variables.
(4) Let $R \in \mathcal{R}$ and F' be a fact, with $F' = F$ or is derived from F. Any homomorphism $h : body(R) \to F'$ can be extended to a homomorphism $h' : \tau_R(body(R)) \to F''$, where F'' is $\mathcal{S}$-derived from $\tau(F')$. More precisely, F'' is obtained from $\tau(F')$ by replacing each $m + j$-th constant ($1 \leq j \leq s$) in $\tau(F')$ atoms by $h(t)$, where $\#_R(t) = j$ (t is the j-th nav variable in R). According to remark (2), F'' can be obtained from $\tau(F')$ with $\mathcal{S}$ rules. h' is defined as follows: for each v_i, $h'(v_i) = c_i$; for each v'_j, with $j = \#_R(t)$, $h'(v'_j) = h(t)$; for each other variable x, $h'(x) = h(x)$. Reciprocally, any homomorphism $h' : \tau_R(body(R))$ to

F'', where F'' is $\mathcal{S}$-derived from $\tau(F')$, can be naturally restricted to a homomorphism $h : body(R) \to F'$.

We now prove the lemma.
($\Rightarrow$) Assume $F, ground(\mathcal{R}) \models Q$. Then, there is a derivation sequence from F to a fact F_k and a homomorphism from Q to F_k, which, by remark (3), can be extended to a homomorphism from $\tau(Q)$ to $\tau(F_k)$. Let us show that $\tau(F_k)$ can be indeed $\tau(\mathcal{R})$-derived from $\tau(F)$. From remark (4): if R_i is applicable to a fact F' by a homomorphism h, then $\tau(R)$ is applicable by an extension h' of h to a fact F'' $\mathcal{S}$-derivable from $\tau(F)$. Furthermore, from remark (2) $\tau(\alpha(F', R_i, h))$ is $\mathcal{S}$-derivable from $\alpha(F'', \tau(R), h')$. Thus, from any $ground(\mathcal{R})$-derivation sequence from F to F_k, one can build an $\tau(\mathcal{R})$-derivation from $\tau(F)$ to $\tau(F_k)$.
($\Leftarrow$) Assume $\tau(F), \tau(\mathcal{R}) \models \tau(Q)$. Then, there is a derivation sequence, say D, from $\tau(F)$ to a fact F'_k and a homomorphism from $\tau(Q)$ to F'_k. From D, we extract the sequence of applications of $\tau(R)$ rules for all $R \in \mathcal{R}$. This subsequence yields a $ground(\mathcal{R})$-derivation sequence from F to F_q, such that $F'_k = \tau(F_q)$. Indeed, assume, for a given rule $R \in \mathcal{R}$, $\tau(R)$ is applicable to a fact F' by homomorphism h'. h' maps $\mathsf{nav}(R)$ to a subset of $\mathcal{C}$. Let R_i be the rule corresponding to this partial grounding. Let F be the fact such that F' can be $\mathcal{S}$-derived from $\tau(F)$. Then R_i is applicable to F by the natural restriction h of h', and by remark (2), $\alpha(F', \tau(R), h')$ is $\mathcal{S}$-derivable from $\tau(\alpha(F, R_i, h))$. Thus, from any $\tau(\mathcal{R})$-derivation from $\tau(F)$ to F'_k, one can extract a $ground(\mathcal{R})$-derivation sequence from F to a fact F_q. Since there is a homomorphism from $\tau(Q)$ to F'_k, F'_k is not only $\mathcal{S}$-derivable from $\tau(F_q)$ but is equal to $\tau(F_q)$. By remark (3), there is a homomorphism from Q to F_q.
□

Theorem 19 *Given F, $\mathcal{R}$ (wfg) and Q, we have that*

$$F, \mathcal{R} \models Q \quad \textit{iff} \quad \tau(F), \tau(\mathcal{R}) \models \tau(Q).$$

Proof: Consequence of Lemmas 1 and 2. □

By adding the set of variables $nav(R)$ to each atom in R, τ transforms each weak guard (of $body(R)$ or $fr(R)$) into a guard. $\mathcal{S}$ is a set of rules with atomic head (furthermore Datalog) thus it is guarded. Hence, if $\mathcal{R}$ is a set of wg (resp. wfg) rules, then fg($\mathcal{R}$) is a set of g (resp. fg) rules.

Property 20 (Properties of τ)

- *If $\mathcal{R}$ is weakly (frontier-) guarded, then $\tau(\mathcal{R})$ is (frontier-) guarded.*
- *$|\tau(\mathcal{R})|$ is polynomial in the combined size if $\mathcal{R}$ and F (more precisely: the size of $\mathcal{C}$).*

- *τ is not arity-preserving: the arity of predicates grows with the size of $\mathcal{C}$.*

Corollary 21 *Any instance of CQ entailment with w(f)g-rules can be polynomially translated into an instance of the same problem with (f)g-rules.*

Proof: The size of $\tau(F)$ (resp. $\tau(\mathcal{R})$, $\tau(Q)$) is polynomially bounded by the combined size of F and $\mathcal{R}$. By adding $\mathsf{nav}(R)$ to each atom in R, τ transforms each weak guard into a guard. The rules from $\mathcal{S}$ are guarded. Hence, if $\mathcal{R}$ is w(f)g, then $\tau(\mathcal{R})$ is (f)g. We conclude with Th. 19. □

6 Frontier-Guarded and Frontier-One Rules

In this section, we show that fg- and fr1-rules are both PTIME-complete for data complexity and 2EXPTIME-complete for combined complexity no matter whether predicate arity is bounded or not.

6.1 Combined Complexity of fg- and fr1-rules

In [BGO10], the authors establish the result that deciding entailment of unions of boolean CQ in the guarded fragment (GF) of FOL is 2EXPTIME-complete. This result can be used to prove the following theorem.

Theorem 22 *CQ entailment for fg-rules is in* 2EXPTIME.

Proof: We first observe that every frontier-guarded $\forall\exists$-rule can be translated into two rules one of which is a guarded $\forall\exists$-ruleand the other is Datalog: take a frontier-guarded rule

$$R: \quad \forall x_1, \ldots, x_k(H \rightarrow (\exists y_1, \ldots, y_{k'}C))$$

with a frontier guard $p(x_{i_1}, \ldots, x_{i_{k''}}) \in H$. Then we introduce a new k''-ary predicate p_R and let $separate(R)$ be the set containing the following two rules.

$$\begin{array}{ll} S_R: & \forall x_1, \ldots, x_k(H \rightarrow p_R(x_{i_1}, \ldots, x_{i_{k''}})) \\ T_R: & \forall x_{i_1}, \ldots, x_{i_{k''}}(p_R(x_{i_1}, \ldots, x_{i_{k''}}) \rightarrow (\exists y_1, \ldots, y_{k'}C)) \end{array}$$

It is straightforward to see that for any frontier-guarded rule set $\mathcal{R}$, we have $F, \mathcal{R} \models Q$ exactly if $F, \bigcup_{R\in\mathcal{R}} separate(R) \models Q$.

Obviously, T_R is guarded (and hence also lies in GF) while S_R is range restricted and frontier-guarded (but not necessarily guarded). The latter are the problematic ones that we have to take care of. Considering a rule of this latter type, i.e.

$$S_R: \quad \forall x_1, \ldots, x_k(H \rightarrow p_R(x_{i_1}, \ldots, x_{i_{k''}}))$$

it is straightforward that it can be equivalently written as the first-order sentence

$$S'_R: \quad \neg\exists x_1, \ldots, x_k(H \wedge \neg p_R(x_{i_1}, \ldots, x_{i_{k''}}))$$

which in turn is obviously equivalent to the set $\{S^*, S^{**}\}$ of formulae consisting of

$$S^*_R: \quad \neg\exists x_1, \ldots, x_k(H \wedge p'_R(x_{i_1}, \ldots, x_{i_{k''}}))$$

and

$$S^{**}_R: \quad \forall x_{i_1}, \ldots, x_{i_{k''}}(p(x_{i_1}, \ldots, x_{i_{k''}}) \wedge \neg p_R(x_{i_1}, \ldots, x_{i_{k''}}) \rightarrow p'_R(x_{i_1}, \ldots, x_{i_{k''}}))$$

for a newly introduced predicate p'_R (remember that $p(x_{i_1}, \ldots, x_{i_{k''}}) \in H$). S^{**}_R can be equivalently transformed into

$$S^{***}_R: \quad \forall x_{i_1}, \ldots, x_{i_{k''}}(p(x_{i_1}, \ldots, x_{i_{k''}}) \rightarrow p'_R(x_{i_1}, \ldots, x_{i_{k''}}) \vee p_R(x_{i_1}, \ldots, x_{i_{k''}}))$$

which is in GF (but not a guarded $\forall\exists$-rule, as it contains a disjunction in the head). So finally, we get that $F, \mathcal{R} \models Q$ iff

$$F \cup \{T_R, S^{***}_R \mid R \in \mathcal{R}\} \cup \{S^*_R \mid R \in \mathcal{R}\} \models Q \qquad (\dagger)$$

where the first two sets are in GF and the third consists of negated existentially quantified conjunctions of atoms. Hence we can conceive every S^*_R as a negated conjunctive query $\neg Q_R$. Consequently we have

$$\{S^*_R \mid R \in \mathcal{R}\} \equiv \{\neg Q_R \mid R \in \mathcal{R}\} \equiv \bigwedge_{R \in \mathcal{R}} \neg Q_R \equiv \neg \bigvee_{R \in \mathcal{R}} Q_R$$

which allows to rephrase $(\dagger)$ as

$$F \cup \{T_R, S^{***}_R \mid R \in \mathcal{R}\} \models Q \vee \bigvee_{R \in \mathcal{R}} Q_R \qquad (\ddagger)$$

which leaves us with a GF theory on the left hand side and a union of boolean conjunctive queries on the right hand side. Since the presented translation from the original problem to UCQ entailment in GF is linear, we have thus established the theorem. □

To prove the 2EXPTIME-hardness of fr1-rules we adopt and adapt the construction used to show 2EXPTIME-hardness for CQ entailment in the DL $\mathcal{ALCI}$ from [Lut07].

Theorem 23 *CQ entailment for fr1-rules with bounded predicate arity is* 2EXPTIME*-hard.*

Proof: We show that Lutz' knowledge base representation of the halting problem for exponentially space-bounded alternating Turing machines can be expressed by frontier-guarded $\forall\exists$-rules by applying the following modifications:

1. The acceptance condition is encoded in a "backward-manner" as in the encoding used to prove EXPTIME-hardness for Horn-$\mathcal{FLE}$ in [KRH07].
2. Negated concepts $\neg A$ are replaced by a concept B defined to be disjoint with A.
3. The query is modified and turned into Horn rules. This ensures that information about configuration changes can be propagated forward making the use of disjunction obsolete.

Then we get a rule set $\mathcal{R}$ with at most binary predicates encoding the above mentioned problem and therefore showing the claim.

For our construction, we modify the encoding of computations of the Turing-Machine in the following way: when visiting an existential state, we enforce the presence of **all** successor states instead of nondeterministically picking just one:

$$R \sqcap \exists s^{m+2}.(q \sqcap a) \sqsubseteq \prod_{(q',a',M)\in\delta(q,a)} \exists s.(R \sqcap T_{q',a',M}) \text{ for all } q \in Q_\exists, a \in \Gamma$$

Obviously, this different encoding necessitates to revisit the way the acceptance condition is encoded. Rather than just forbidding that the rejecting state q_r occurs anywhere, we determine whether the initial configuration is accepting by propagating information of acceptance backwards the computation tree back to the initial configuration using the concept name T_{accept}:

$$\begin{aligned}
R \sqcap \exists s^{m+2}.q_a &\sqsubseteq T_{accept} \\
R \sqcap \exists s^{m+2}.(q \sqcap a) \sqcap \exists s.(R \sqcap T_{q',a',M} \sqcap T_{accept}) &\sqsubseteq T_{accept} \\
\text{for all } q \in Q_\exists, a \in \Gamma, (q',a',M) \in \delta(q,a) & \\
R \sqcap \exists s^{m+2}.(q \sqcap a) \sqcap \prod_{(q',a',M)\in\delta(q,a)} \exists s.(R \sqcap T_{q',a',M} \sqcap T_{accept}) &\sqsubseteq T_{accept} \\
\text{for all } q \in Q_\forall, a \in \Gamma &
\end{aligned}$$

We now further modify $\mathcal{T}_w$ in a semantics-preserving way in order to obtain a Horn-$\mathcal{ALC}^{\mathsf{rs}}$ TBox. First substitute every occurrence of $\neg A_i$ by a new atomic concept $\overline{A_i}$, likewise substitute $\neg H$ by $\overline{H}$ and $\bigsqcap_{q\in Q} \neg q$ by $\overline{Q}$, then add the following

set of axioms to $\mathcal{T}_w$: $A_i \sqcap \overline{A_i} \sqsubseteq \bot$ for $i < m$, $H \sqcap \overline{H} \sqsubseteq \bot$, and $\overline{Q} \sqcap \bigsqcap_{q \in Q} \sqsubseteq \bot$. Note that this encoding is possible as the tertium non datur part of the negation is not needed for the modeling.

As next observation we find that some of the conditions for configuration trees which are not in Horn-$\mathcal{ALC}^{\mathsf{rs}}$ can be discarded or rewritten into Horn form:

- $G \sqsubseteq \bigsqcup_{a \in \Gamma} a \sqcap \bigsqcap_{a,a' \in \Gamma, a \neq a'} \neg(a \sqcap a')$
  The disjunction on the right hand side can be safely dismissed as the subsequent axiomatization realizes that (1) every tape position of the initial configuration gets assigned one entry and (2) the interplay of axioms and the way we rewrite the query into a rule ensures that a complete assignment of tape entries in one configuration ensures a complete assignment in all successor configurations.

- $(\underline{L_i} \sqcap H) \sqsubseteq (\forall s.((L_{i+1} \sqcap A_i) \rightarrow H) \sqcap \forall s.((L_{i+1} \sqcap \overline{A_i}) \rightarrow \overline{H})) \sqcup (\forall s.((L_{i+1} \sqcap \overline{A_i}) \rightarrow H) \sqcap \forall s.((L_{i+1} \sqcap A_i) \rightarrow \overline{H}))$
  This axiom ensures that there is exactly one position on the tape where the head is located. By a similar argument as above, we know that the rest of the axiomatization already ensures that there is **at least** one head position indicated in every configuration. In order to guarantee that in every configuration there is **at most** one head position, we can alternatively use the following axioms: $L_i \sqcap \exists s.(L_{i+1} \sqcap H \sqcap A_{i+1}) \sqsubseteq H \sqcap \forall s.(L_{i+1} \sqcap \overline{A_{i+1}} \rightarrow \overline{H})$.

- $L_m \sqcap H \sqsubseteq \forall s^2.(G \rightarrow \bigsqcup_{q \in Q} q)$ This axiom can be discarded as well, as the rest of the axioms ensures a deterministic assignment of one q to the head position.

We now come to the core part of our modification. Recap that the purpose of the conjunctive query $q_w = q_Y \cup q_Z \cup \bigcup_{i<m} q_w^i$ is to detect cases where tape entries with the same binary addresses deviate in terms of their entries or the indicated state of the Turing machine. Thereby $q_Y \cup \bigcup_{i<m} q_w^i$ is used to identify pairs of corresponding cells (and bind them to variables v and v') whereas q_Z detects deviating assignments in order to rule them out. As we aim at enforcing synchronized entries rather than eliminating models where they are out of sync, we only reuse the identification part of the query in our rule bodies and define $\mathcal{R}_{sync} = \{R_C \mid C \in Q \cup \{\overline{Q}\} \cup \Gamma\}$ where R_C is defined as $q_Y \cup \bigcup_{i<m} q_w^i \cup \{C(v)\} \rightarrow C(v')$.

Summing up, we have come up with a set of first-order Horn rules $\mathcal{R}_{sync}$ and rewritten Lutz' TBox $\mathcal{T}_w$ into a Horn-$\mathcal{ALC}^{\mathsf{rs}}$ TBox $\mathcal{T}'_w$, such that (using the first-order semantics) $\mathcal{A}_w \cup \mathcal{T}'_w \cup \mathcal{R}_{sync} \models T_{accept}(a)$ exactly if the Turing machine accepts w. We further find that $\mathcal{A}_w \cup \mathcal{T}'_w \cup \mathcal{R}_{sync}$ can be rewritten from Horn-$\mathcal{ALC}^{\mathsf{rs}}$ to Horn-$\mathcal{ALCI}$ such that ground consequences are preserved (see again

[Lut07]). Finally, the standard translation into first-order logic of any Horn-$\mathcal{ALCI}$ knowledge base yields a frontier-1 rule base. Likewise, the rewriting of $\mathcal{R}_{sync}$ yields only $\forall\exists$-rules with frontier size one (note however for Section 7 that these rules are not acyclic). Consequently we have shown that the word problem of an alternating ExpSpace-bounded Turing machine can be polynomially encoded into the entailment problem for frontier-one $\forall\exists$-rules with predicate arity ≤ 2. Hence we have shown 2EXPTIME-hardness for the latter.

□

6.2 Data Complexity of fg- and fr1-rules

The proof for PTIME membership for data complexity is based on a specific locality property of derivations for fg-rules which is established in the following lemma 4.

Let us first assume that no constant occurs in $\mathcal{R}$ and Q (we will introduce them later). As we are interested in data complexity, we consider that $\mathcal{R}$ and Q are fixed. Let n be the maximal number of frontier variables occurring in $\mathcal{R}$ and let m be the maximal number of variables occurring in the body of any rule or in Q. Now, let G denote all the semantically different formulae of the shape

$$\exists x_1, \ldots, x_m(\bigwedge_{atom \in Conj} atom)$$

where $Conj$ is a subset of the set of all atoms obtained from predicates $\mathcal{P}$ and terms from $\{x_1, \ldots, x_m, y_1, \ldots, y_n\}$. Given terms $t_1, \ldots, t_n$, it suffices to know for which $\varphi \in G$ holds $\varphi[y_1/t_1, \ldots, y_n/t_n]$ in order to determine what roles $t_1, \ldots, t_n$ can play in any rule application. Obviously, G is finite, so let $|G| = \nu$. For any specific φ and $t_1, \ldots, t_n$ we need at most m terms (i.e. assignments to the existentially quantified variables) to witness that $\varphi[y_1/t_1, \ldots, y_n/t_n]$ holds. Hence, an upper bound for witnesses for *all* respective φs is $w_{\max} := \nu \cdot m$.

We now proof two auxiliary lemmas.

Lemma 3 (Derivation Sequence Splitting) *Let $F, \mathcal{R} \models Q$ witnessed by an according derivation sequence $F = F_0, \ldots, F_k$. Let moreover $a \in F_{i+1} \setminus F_i$ be an atom with vars$(a) \in$ vars(F). Then there are derivation sequences for $F, \mathcal{R} \models a$ and $F \cup \{a\}\mathcal{R} \models Q$ both having a length strictly smaller than k.*

Lemma 4 *Let $\mathcal{R}$ be a fg rule set. Let $F, \mathcal{R} \models a$ for an atom $a = p(z_1, \ldots, z_l)$ with vars$(a) \subseteq$ vars(F) for which there is an according derivation sequence $F = F_0, \ldots, F_k$ such that all atoms from $F_k \setminus \{a\}$ that contain only variables occurring in F are already present in F. Then there is a set $V \subseteq$ vars(F) with $|V| \leq w_{\max}$ such that $F|_V, \mathcal{R} \models a$ where $F|_V := \{b \mid b \in F,$ vars$(b) \subseteq V\}$.*

Proof: We choose V such that it contains a witness for every satisfied φ w.r.t. $z_1, \ldots, z_l$ as defined above. Now we proof the statement by an induction on the derivation length k. Note that the case for $k = 0$ is trivial, as clearly $\{z_1, \ldots, z_l\} \in V$. For $k = 1$, we have that $a \in \alpha(F, R, \pi)$. In that case, we can be sure that V contains a witness for the rule body of R.

For greater k, consider the first step of the derivation sequence $F_1 = \alpha(F_0, R, \pi)$. Because of the assumption, *head*(R) must contain at least one existentially quantified variable per atom. Hence, let $\{y_1, \ldots, y_m\} :=$ *vars*$(F_1 \setminus F_0)$ the nonempty set of variables newly introduced by the first derivation step.

Now we split the derivation sequence starting from F_1 (having length $k - 1$) at every point where $F_{i+1} \setminus F_i = \{a'\}$ for an atom a' with *vars*$(a') \subseteq$ *vars*(F_1). We end up with separate derivation sequences $F_1 \dashrightarrow a'_1$, $F \cup \{a'_1\} \dashrightarrow a'_2$, ..., $F \cup \{a'_1, \ldots, a'_{l-1}\} \dashrightarrow a'_l = a$, each satisfying the condition that the finally derived fact is the only new fact which contains only variables from *vars*(F_1). Moreover, due to our assumption, none of the $a'_1, \ldots, a'_l$ contains variables only from *vars*(F). According to Lemma , each of these derivation sequences has length of at most $k - 1$. We now apply the induction hypothesis to identify the variables $V'_j \subseteq$ *vars*(F_j) with $|V'_j| \leq n$ such that $F_j|_{V'_j}, \mathcal{R} \models a'_j$. The variables from $V'_j \cap$ *vars*$(F) \setminus \{x_1, \ldots, x_l\}$ are exchanged by substitutes from V such that the set of witnessed φs (w.r.t. the variables contained in a'_j) does not change, hence $F_j|_{V''_j}, \mathcal{R} \models a'_j$ still holds for the modified V''_j. This is possible due to the former choice of V. Each of the modified V''_j that we thus obtain contains only variables from $V \cup \{y_1, \ldots, y_m\}$, therefore we can conclude $F_1|_{V \cup \{y_1, \ldots, y_m\}}, \mathcal{R} \models a$. As we already know that $F|_V, \mathcal{R} \models (F_1 \setminus F_0)$ and $\{y_1, \ldots, y_m\} \subseteq$ *vars*$(F_1 \setminus F_0)$ we also have $F|_V, \mathcal{R} \models a$. □

Relying on Lemma 4 we next provide a translation of constant-free $\mathcal{R}$ and Q into a Datalog program. The main idea is to "compile away" existential variables introduced in rule heads by "precomputing" deduction sequences that finally result in a query match.

Definition 23 *Given a constant-free fg-rule set $\mathcal{R}$, we define the Datalog-program $\mathbf{P}(\mathcal{R})$ as follows: Let $\{y_1, \ldots, y_{w_\mathcal{R}}\}$ be a set of variable symbols. Let $\mathbb{G}$ denote the finite set of all atoms with predicates from $\mathcal{R}$ and terms from $\{y_1, \ldots, y_{w_\mathcal{R}}\}$. Now let $\mathbf{P}(\mathcal{R})$ be the set of Datalog rules containing every $\forall y_1, \ldots, y_{w_\mathcal{R}}(B \to h)$ (with $B \subseteq \mathbb{G}$ and $h \in \mathbb{G}$) for which $B, \mathcal{R} \models h$.*

We define the Datalog-program $\mathbf{P}(\mathcal{R}, Q)$ as follows:

- Let $\{y_1, \ldots, y_{w_{\max}}\}$ be a set of variable symbols. Let $Ground$ denote the set of all "groundings" of $\mathcal{P}$-atoms with symbols from $\{y_1, \ldots, y_{w_{\max}}\}$ (note

that $Ground$ is finite). Now let $\mathbf{D}$ be the set of Datalog rules (i.e., such that $var(head) \subseteq var(body)$) that contains every $\forall y_1, \ldots, y_{w_{\max}}(body \rightarrow head)$ (with $body \subseteq Ground$ and $head \in Ground$) for which $body, \mathcal{R} \models head$.[5] Then, let $\mathbf{P}(\mathcal{R}, Q)$ contain $\mathbf{D}$.

- Likewise, for every grounding Q' of Q (i.e., obtained from Q by mapping its variables to $\{y_1, \ldots, y_{w_{\max}}\}$) and any $body \in Ground$ with $body, \mathcal{R} \models Q'$, let $\mathbf{P}(\mathcal{R}, Q)$ contain the rule $\forall y_1, \ldots, y_{w_{\max}}(body \rightarrow match)$, where $match$ is a new nullary predicate symbol.

Then we get the following lemma:

Lemma 5 *For $\mathcal{R}$ (fg) without constants, for every fact F, $F, \mathcal{R} \models Q$ iff $F, \mathbf{P}(\mathcal{R}, Q) \models match$.*

Proof: $\Leftarrow$: all rules of the first kind in $\mathbf{P}(\mathcal{R}, Q)$ are consequence of $\mathcal{R}$ and the body of the match rules encode facts built on variables to which Q can be mapped with a surjective homomorphism.
$\Rightarrow$: First, it is straightforward that $F, \mathcal{R} \models Q$ iff $F, \mathcal{R} \cup \{Q \rightarrow match\} \models match$. Let $F = F_0, \ldots, F_k$ be a derivation sequence for $F, \mathcal{R} \cup \{Q \rightarrow match\} \models match$. We split this derivation sequence starting from F_0 at every point where $F_{i+1} \setminus F_i = \{a\}$ for an atom a with $\mathit{vars}(a') \subseteq \mathit{vars}(F_0)$. We end up with separate derivation sequences $F \dashrightarrow a_1$, $F \cup \{a_1\} \dashrightarrow a_2$, ..., $F \cup \{a_1, \ldots, a_{l-1}\} \dashrightarrow a_l = match$, each satisfying the condition that the finally derived fact is the only new fact which contains only variables from $\mathit{vars}(F)$. For every of these derivation sequences $F_j \dashrightarrow a_j$ we can apply Lemma 4 to find sets of variables $V_j \subseteq \mathit{vars}(F_j)$ such that $F_j|_{V_j}, \mathcal{R} \models a_j$. Yet, then $\mathbf{P}(\mathcal{R}, Q)$ contains an according rule which allows to deduce a_j from F_j in one step. Put together this gives a deduction of $F, \mathbf{P}(\mathcal{R}, Q) \models match$. □

In order to leverage the above lemma for arbitrary fg-rule sets containing constants, we need to transform the task of deciding $F, \mathcal{R} \models Q$ into a setting where constants are excluded. The following definition and lemma provide for this by applying a partial grounding and subsequently shifting positions taken by constants into predicates.

Definition 24 *Let $\mathcal{R}$ be an arbitrary fg-rule set and let Q be a CQ. Let A be the set of constants occurring in $\mathcal{R}$ and Q. For every predicate p of arity k occurring in $\mathcal{R}$ and Q and every partial mapping $\gamma : \{1, \ldots, k\} \rightarrow A$, we let p_γ denote*

[5] Here we need entailment checks with any decision procedure. But this can be done independent of the underlying fact and the created Datalog program is fixed for given $\mathcal{R}$ and Q.

a new $(k - |dom(\gamma)|)$-ary predicate. Let ξ_A map atoms from $\mathcal{R}$ and Q to new atoms by projecting out positions filled by constants from A.[6] *We lift the function ξ_A to conjuncts and rules in the obvious way. Now, letting $PG_A(R)$ denote all* partial groundings *of R where some universally quantified variables are substituted by constants from A, we define the rule set* cfree$(\mathcal{R}, Q) = \{\xi_A(R') \mid R' \in PG_A(R), R \in \mathcal{R} \cup \{Q \rightarrow match\}\}$

Lemma 6 *For $\mathcal{R}$ (*fg*), cfree$(\mathcal{R}, Q)$ is* fg *and constant-free. Given a fact F and assuming fixed $\mathcal{R}$ and Q, the size of $\xi_A(F)$ and the time to compute it is polynomially bounded by $|F|$. Moreover $F, \mathcal{R} \models Q$ iff $\xi_A(F)$, cfree$(\mathcal{R}, Q) \models match$.*

The preceding lemma then allows to extend Lemma 5 to fg-rules with constants:

Lemma 7 *For $\mathcal{R}$ (*fg*) holds $F, \mathcal{R} \models Q$ exactly if $\xi_A(F), \mathbf{P}($cfree$(\mathcal{R}, Q), match) \models match$.*

We can now state the following complexity Theorem:

Theorem 24 *CQ entailment for fg- and fr1-rules is* PTIME*-complete for data complexity.*

Proof: Thanks to Lemma 7, we have reduced the problem to atom entailment in Datalog. Noting that $\mathbf{P}($cfree$(\mathcal{R}, Q), match)$ is independent from F and (w.l.o.g. assuming that F contains only constants) that $\xi_A(F)$ consists only of ground atoms, PTIME data complexity membership follows from the PTIME data complexity of entailment in Datalog [DEGV01]. PTIME-hardness for data complexity is a direct consequence of the same result for propositional Horn logic. □

7 Body-Acyclic fg- and fr1-Rules

Tree-like structures often lead to lower complexity. We focus here fg-rules with an acyclic body (in the sense of hypergraph-acyclicity). In this section, we study the complexity of frontier-guarded rules with an acyclic body. The acyclicity notion considered here is the *hypergraph acyclicity* stemming from database theory, which corresponds to the independently defined notion of an *acyclic guarded covering* [Ker01] (see [CM09] for details about this equivalence).

To simplify the next notions, we first proceed with some normalization of a set of fg-rules, such that all rules are either disconnected (they have an empty frontier) or they have a non-empty frontier and a "variable-connected" body:

[6] For instance $\xi_{\{a,b\}}(p(x, a, b, c)) = p_{\{2 \mapsto a, 3 \mapsto b\}}(x, c)$.

1. Let R be a rule with a non-empty frontier and let $\mathcal{B}$ be the hypergraph assigned to $body(R)$. For each node in $\mathcal{B}$ assigned to a constant, split it in as many nodes as hyperedges it belongs to (thus each constant node obtained belongs to a single hyperedge); let $\mathcal{B}'$ be the hypergraph obtained; let C_f be the connected component of $\mathcal{B}'$ that contains the frontier guard(s) (if there are several frontier guards, they are all in the same connected component);

2. build a rule R_0 by gathering all connected components of $\mathcal{B}'$ except C_f; the body of R_0 is made of the conjunction of all these atoms and its head is restricted to a single nullary predicate p_0.

3. build the rule R_f with body made of the atoms from C_f and p_0, and head equal to $head(R)$.

Let us consider an instance $(F, \mathcal{R}, Q)$ of the entailment problem, where $\mathcal{R}$ is a set of fg-rules. We process all non-disconnected rules from $\mathcal{R}$ as described above, which yields an equivalent set of rules. Then, we eliminate disconnected rules (initial disconnected rules and obtained R_0 rules) by integrating them to F as in [BLM10]: this can be done with d calls to an oracle solving the entailment problem for fg-rules, where d is the number of disconnected rules.

From now on, we thus assume that all fg-rules have a non-empty frontier and their body on non-nullary predicates is "variable-connected", i.e., the associated hypergraph is connected and cannot be disconnected by the above step 1 ($\mathcal{B} = \mathcal{B}'$). We will also ignore nullary predicates, as they play absolutely no role.

7.1 Body-Acyclic fg-Rules (General case)

Let $\mathcal{H}$ be a hypergraph. The acyclicity of $\mathcal{H}$ is usually defined with respect to its so-called *dual* graph, whose nodes are the hyperedges of $\mathcal{H}$ and edges represent their intersection. We will use here a close notion, that we call *decomposition graph* of a set of atoms, which groups together a guard and the atoms it guards into a single node.

We assign the following decomposition graph $D(S)$ to a set of atoms S:

- Let $\{C_1, \dots C_p\}$ be a partition of S such that each C_i is guarded, with p minimal for this property.

- The set of nodes of $D(S)$ is $\{C_1, \dots C_p\}$ and there is an edge C_iC_j (with $i \neq j$) if C_i and C_j share (at least) a variable; each edge C_iC_j is labeled by the variables common to C_i and C_j (noted $vars(C_iC_j)$).

An edge C_iC_j of the decomposition graph is said to be *removable* if there is another path between C_i and C_j such that the labels of all edges in this path contain

the label of C_iC_j (this condition has to do with the so-called running decomposition property: the set of nodes in which a given variable occurs yields a connected subgraph of the decomposition graph; removing removable edges keeps this property). S is said to be (hypergraph-) acyclic (which corresponds to the fact that the associated hypergraph is acyclic) if each connected component of its decomposition graph can be turned into a tree by removing only removable edges (such a tree is usually called a *join tree* in databases). It is well known that when a hypergraph is acyclic, a join tree/forest can be built from it in polynomial time. An fg-rule R is said to be body-acyclic *(ba)* if its body is hypergraph-acyclic.

Note that our starting assumption (variable-connected body) ensures that the decomposition graph of the fg-rules we consider is connected. Let R be a ba-fg rule and let J be an join tree associated with *body(R)*. Let C_r be a J node that contains a frontier guard. J is considered as rooted in C_r, which yields a direction of its edges from children to parents: a directed edge (C_i, C_j) is from a child to its parent. R is translated into a set of guarded rules $\{R_1, \dots R_p\}$ as follows:

- To each edge (C_i, C_j) is assigned the atom $a_i = q_i(var(CiCj))$, where q_i is a new predicate
- To each $C_i \neq C_r$ is assigned the rule:
  $R_i = (conjunct(C_i) \wedge \bigcup_{(C_k,C_i)} a_k) \rightarrow a_i$
- To C_r is assigned the rule
  $R_r = (conjunct(C_r) \wedge \bigcup_{(C_k,C_r)} a_k) \rightarrow head(R)$

Note that the decomposition graph associated with the body of a guarded rule is restricted to a single node. Thus, guarded rules are trivially ba-fg rules. The above translation is the identity on guarded rules. The translation is linear in the size of the rule and arity-preserving (the arity of the new predicates is bounded by the arity of existing predicates). We conclude that the complexity of conjunctive query entailment with ba-fg-rules is the same as with guarded rules: for combined complexity, it is EXPTIME-complete with bounded arity and 2EXPTIME-complete with unbounded arity; it is in P for data complexity.

The above reduction is polynomial in the size of the rule and arity-preserving. Thus, previous complexity results on guarded rules apply to ba-fg-rules (in particular they are EXPTIME-complete for bounded-arity combined complexity, while fg-rules are 2EXPTIME-complete).

Finally, let us point out that hypergraph-acyclicity of rule bodies alone is not enough to obtain good properties: that the head of a rule is connected to only one node of the decomposition graph (thus, that the frontier is guarded) is crucial. Without this assumption, the entailment problem remains undecidable. This can

be checked for instance with the reduction from the word problem in a semi-Thue system, known to be undecidable, to the entailment problem (in a conceptual graph setting) built in [BM02]. This reduction yields rules with a body restricted to a path and frontier of size 2.

7.2 Body-Acyclic fg-Rules (Unary/Binary case)

We now consider the special case of predicates with arity bounded by 2, which is relevant w.r.t. description logics. Let us note ba-fg-2 this rule fragment.

To a set of binary atoms is naturally associated a (multi-)graph (unary atoms can be ignored for our purpose). The acyclicity notion can be simplified. Let us say that a cycle in such a graph is a *true* cycle if it contains only (distinct) variable nodes and is of length strictly greater than 2 (in other words, multi-edges are allowed).

Property 25 *A set of binary atoms has no true cycle if and only if it is hypergraph-acyclic.*

Proof: In the binary case, edges in the decomposition graph contain exactly one variable. Let G_S be the graph of the atom set S and D_S be its decomposition graph. If G_S has no true cycle, then any cycle in D_S is made of edges labeled by the same variable; thus, any of these edges is removable, which breaks the cycle. In the other direction: take a true cycle in G_S, which passes through k distinct variables, $k \geq 2$. It yields a cycle in D_S with all k edges labeled by a distinct variable. It can be checked that if D_S has a cycle whose edges do not have all the same label (and in this case this cycle has at least three edges with distinct labels), then it cannot be made acyclic (f.i., by induction on the number of removable edges in D_S). □

Let us now sketch links between the ba-fg-2 rule fragment and Horn description logics, firstly introduced in [HMS05]. Based on the fact that description logics [BCM$^+$07] are fragments of first-order logic, the logic Horn-$\mathcal{L}$ for some DL $\mathcal{L}$ is roughly defined as the axioms of $\mathcal{L}$ whose standard translation into FOL and subsequent skolemisation yields Horn clauses. The ba-fg-2 fragment includes the DL Horn-$\mathcal{ALCOI}^{\mathsf{Self}}(\sqcap)$ (see [KRH07] for a treatment of Horn DLs – $\mathcal{ALC}$ denotes the basic Boolean-closed description logic; $\mathcal{O}$ indicates the nominals and is needed if constants are involved; $\mathcal{I}$ stands for role inverse; Self allow concepts of the form $\exists R.\mathsf{Self}$ to indicate roles looping back to their source, $(\sqcap)$ indicates role conjunction). As a consequence of results from [KRH07], deduction of single atom facts (and hence CQ entailment) in Horn-$\mathcal{ALCOI}^{\mathsf{Self}}(\sqcap)$ is ExpTime-hard; CQ answering is known to be in 2ExpTime as a consequence of results in [CEO09]. However note that ba-fg-2 rules are more expressive than that: they allow for arbitrarily structured rule heads, whereas for Horn-$\mathcal{ALCOI}^{\mathsf{Self}}(\sqcap)$ rule heads are

required to be essentially tree-structured (a consequence implicitly imposed by the DL notation).

Still from the results on guarded rules, we get that conjunctive query entailment with ba-fg-2 rules is in EXPTIME for combined complexity and in P for data complexity. An EXPTIME lower bound easily follows from the fact that standard reasoning in the much weaker Horn-$\mathcal{FLE}$ is already EXPTIME-Hard, as shown in [KRH07][7]. The problem is thus EXPTIME-complete. Since rules translating Horn-$\mathcal{FLE}$ are both guarded and fr1[8], this lower bound holds for ba-guarded rules and ba-fr1-rules.

7.3 Body-Acyclic Frontier-1 Rules

PTIME-complete data complexity follows from the proof of Th. 24; about combined complexity, EXPTIME-hardness with bounded arity (thus with unbounded arity too) follows from the fact that standard reasoning in the weaker DL fragment Horn-$\mathcal{FLE}$ is EXPTIME-hard (see preceding subsection); from EXPTIME membership of guarded rules in the bounded arity case, we conclude that ba-fr1-rules are EXPTIME-complete with bounded-arity. The only remaining question is whether they are simpler than ba-fg-rules in the unbounded arity case. We established EXPTIME membership for the constant-free variant, and keep trying to prove it for the general case.

8 Conclusion

We have introduced the notion of greedy bts of existential rules that subsumes guarded rules as well as their known generalizations and gives rise to a generic algorithm for deciding CQ entailment. Moreover, we have classified known gbts subclasses w.r.t. their combined and data complexities. Some interesting open issues remain, e.g. the exact complexity of gbts in the unbounded predicate arity case and the recognizability of gbts. Future work will aim at the integration of rules expressing equality and other properties such as transitivity into this framework, preserving decidability, and trying to keep the desirable PTIME data complexity of fg-rules.

[7] It is known that conjunctive query entailment is at least as difficult as standard reasoning.

[8] Briefly said, in Horn-$\mathcal{FLE}$, we have $\sqcap$, $\exists R.C$, $\top$, $\bot$, and $\forall R.C$ (only in the right side of inclusions); no role inclusion, nor role composition.

References

[AHV94] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1994.

[BCM+07] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[BGO10] Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 1–10. IEEE Computer Society, 2010.

[BLM10] J.-F. Baget, M. Leclère, and M.-L. Mugnier. Walking the decidability line for rules with existential variables. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Canada*. AAAI Press, 2010.

[BLMS08] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. DL-$\mathcal{SR}$: a lite DL with expressive rules: Preliminary results. In *Proceedings of the Twenty-First International Workshop on Description Logics (DL2008), Dresden, Germany*, 2008.

[BLMS09] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, California, USA,*, pages 677–682, 2009.

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

[BV81] C. Beeri and M. Vardi. The implication problem for data dependencies. In *Proceedings of Automata, Languages and Programming, Eigth Colloquium (ICALP 1981), Acre (Akko), Israel*, volume 115 of *LNCS*, pages 73–85, 1981.

[BV84] C. Beeri and M.Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

[CEO09] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In Craig

Boutilier, editor, *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI'09)*, pages 714–720. IJCAI, 2009.

[CGK08] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia*, pages 70–80, 2008.

[CGL+07] Diego Calvanese, Guiseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[CGL09] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the 28th symposium on Principles of database systems (PODS'09)*, pages 77–86, New York, NY, USA, 2009. ACM.

[CGL+10] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242, 2010.

[CLM81] A. K. Chandra, H. R. Lewis, and J. A. Makowsky. Embedded implicational dependencies and their inference problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing (STOC 1981), Milwaukee, Wisconsin, USA*, pages 342–354, 1981.

[CM09] M. Chein and M.-L. Mugnier. *Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.

[Cou90] B. Courcelle. The monadic second-order logic of graphs: I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[DEGV01] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[FKMP05] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[HMS05] Ulrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 466–471. Professional Book Center, 2005.

[JK84] D.S. Johnson and A.C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[Ker01] G. Kerdiles. *Saying it with Pictures: a logical landscape of conceptual graphs*. PhD thesis, Univ. Montpellier II / Amsterdam, Nov. 2001.

[KRH07] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 452–457. AAAI Press, 2007.

[LTW09] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic el using a relational database system. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, California, USA,*, pages 2070–2075, 2009.

[Lut07] C. Lutz. Inverse roles make conjunctive queries hard. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, CEUR-WS, 2007.