

## A Generic Querying Algorithm for Greedy Sets of Existential Rules

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

► **To cite this version:**

Michaël Thomazo, Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph. A Generic Querying Algorithm for Greedy Sets of Existential Rules. *KR: Principles of Knowledge Representation and Reasoning*, Jun 2012, Rome, Italy. 13th International Conference on Principles of Knowledge Representation and Reasoning pp.096-106, 2012, <<http://www.dis.uniroma1.it/kr12/>>. <lirmm-00763518>

**HAL Id: lirmm-00763518**

**<https://hal-lirmm.ccsd.cnrs.fr/lirmm-00763518>**

Submitted on 11 Dec 2012

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Generic Querying Algorithm for Greedy Sets of Existential Rules

**Michaël Thomazo**

Univ. Montpellier 2  
France  
thomazo@lirmm.fr

**Jean-François Baget**

INRIA  
France  
baget@lirmm.fr

**Marie-Laure Mugnier**

Univ. Montpellier 2  
France  
mugnier@lirmm.fr

**Sebastian Rudolph**

KIT  
Germany  
rudolph@kit.edu

## Abstract

Answering queries in information systems that allow for expressive inferencing is currently a field of intense research. This problem is often referred to as ontology-based data access (OBDA). We focus on conjunctive query entailment under logical rules known as tuple-generating dependencies, existential rules or Datalog+/. One of the most expressive decidable classes of existential rules known today is that of greedy bounded treewidth sets (*gbts*). We propose an algorithm for this class, which is worst-case optimal for data and combined complexities, with or without bound on the predicate arity. A beneficial feature of this algorithm is that it allows for separation between offline and online processing steps: the knowledge base can be compiled independently from queries, which are evaluated against the compiled form. Moreover, very simple adaptations of the algorithm lead to worst-case-optimal complexities for specific subclasses of *gbts* which have lower complexities, such as guarded rules.

## Introduction

Answering conjunctive queries (CQs) in information systems that allow for expressive inferencing is currently a field of intense research, receiving input from several other domains. Instances of this problem have been addressed in different research domains, most notably the field of Semantic Web technologies where the problem is referred to as ontology-based data access (OBDA), and the database area, where the interest focusses on CQ entailment under rule-based deduction formalisms such as tuple-generating dependencies (TGDs) (Beeri and Vardi 1984) or Datalog+/- (Calì, Gottlob, and Kifer 2008; Calì, Gottlob, and Lukasiewicz 2009), also referred to as *existential rules* (Baget et al. 2009; 2011). The body and the head of these rules are arbitrary conjunctions of atoms (without function symbols) and variables occurring in the head but not in the body are existentially quantified. While entailment with existential rules is undecidable in general, lately, a plethora of logical fragments has been identified for which CQ answering is decidable, alongside with tight complexity bounds for most of them. One of the most expressive decidable class of existential rules known today is that of *greedy*

*bounded treewidth sets (gbts)* (Baget et al. 2011), which subsumes well-known formalisms such as (plain) Datalog and guarded rules, as well as generalizations of these (Calì, Gottlob, and Kifer 2008; Baget, Leclère, and Mugnier 2010; Krötzsch and Rudolph 2011). These fragments cover the core of lightweight description logics dedicated to query answering, namely DL-Lite (Calvanese et al. 2007),  $\mathcal{EL}$  (Baader, Brandt, and Lutz 2005; Lutz, Toman, and Wolter 2009) – which are the basis of the tractable fragments of the OWL Web Ontology Language – and more broadly the family of Horn description logics (Krötzsch, Rudolph, and Hitzler 2007; Ortiz, Rudolph, and Šimkus 2011).

While the decidability and complexity landscape of these formalisms is clearing up, few attempts have been made to find algorithms for CQ answering that are of more than just theoretical interest – notable exceptions being OWL tractable fragments and very simple Datalog+/- classes. Beyond these lightweight formalisms, CQ answering is usually considered a problem too hard to be practically solvable.

We undertake a step in this direction by devising an algorithm that sharply improves over an earlier proposal (Baget et al. 2011) by (i) allowing a more direct computation without the use of oracles thus being conceptually simpler and much easier to understand, (ii) allowing beneficial separation between offline and online processing steps, as the knowledge base can be compiled independently from queries, which are evaluated against the compiled form, and (iii) exhibiting worst-case-optimal complexity for *gbts*, as well as for specific subclasses of *gbts* which have lower complexities, by very simple adaptations of the algorithm.

Moreover, our endeavor is not without theoretical merit. First, our algorithm improves over the earlier one in terms of combined complexity from 3EXPTIME to 2EXPTIME, thereby establishing a novel upper bound and yielding that deciding CQ entailment under *gbts* rules is in fact 2EXPTIME-complete. Second, we establish a novel tight bound for query complexity since we prove that CQ entailment under *gbts* rules is NP-complete for query complexity.

Please see the accompanying research report (Thomazo et al. 2012) for more details.

## Outline

We give here an informal high-level description of the algorithm. Due to the existential variables in rule heads, a

forward chaining mechanism (like for instance the so-called chase in databases) does not halt in general. However, for *gbts* rules, each sequence of rule applications gives rise to a so-called *derivation tree*, which is a decomposition tree of the derived set of facts; moreover, this tree can be built in a *greedy* way: each rule application produces a new tree node (called a *bag*), which contains the atoms created by the rule application, such that the derived set of facts is the union of all bag atoms from this tree. The set of derived facts is potentially infinite, but thanks to its tree-like structure, the forward chaining process can be stopped after a finite number of rule applications.

The algorithm proceeds in two steps: first, it computes a finite tree, called a (*full*) *blocked tree*, which finitely represents all possible derivation trees; second, it evaluates a query against this blocked tree. Building a blocked tree relies on two notions:

- *bag patterns*: each bag is associated with a *pattern*, which encodes all ways of mapping a (subset of a) rule body to the current facts, while using some terms from this bag. It follows that a rule is applicable to the current facts iff one of the bag patterns contains its rule body. Then, the forward chaining can be performed “on the bag-level” by forgetting about the current facts and considering solely the derivation tree decorated with patterns. At each step, patterns are maintained and kept up-to-date by a propagation procedure based on a *join* operation between the patterns of adjacent bags.
- an *equivalence* relation on bags: thanks to patterns, an equivalence relation can be defined on bags, so that two bags are equivalent if and only if the “same” derivation subtrees can be built under them. The algorithm develops only one node per equivalence class, the other being *blocked* (note however that equivalence classes evolve during the computation, thus a blocked node can later become unblocked, and vice-versa). This tree grows until no new rule application can be performed to unblocked bags: the *full blocked tree* is then obtained.

The evaluation of a conjunctive query against a blocked tree cannot be performed by a simple homomorphism test. Instead, we define the notion of a *\*-homomorphism*, which can be seen as a homomorphism to an “unfolding” or “development” of this blocked tree. As the length of the developed paths is bounded with an exponent that depends only on the rule set (more precisely, the exponent is the maximal number of variables shared by the body and the head of a rule), checking if there is a *\*-homomorphism* from a conjunctive query to a blocked tree is time polynomial in data complexity and nondeterministically time polynomial in query complexity.

## Preliminaries

An *atom* is of the form  $p(t_1, \dots, t_k)$  where  $p$  is a predicate with arity  $k$ , and the  $t_i$  are terms, *i.e.* variables or constants. A *conjunct*  $C[\mathbf{x}]$  is a finite conjunction of atoms, where  $\mathbf{x}$  is the set of variables occurring in  $C$ . A *fact* is the existential

closure of a conjunct.<sup>1</sup> A Boolean *conjunctive query* (CQ) has the same form as a fact, thus we identify both notions. We also represent conjuncts, facts and CQs as sets of atoms. Given an atom or a set of atoms  $A$ ,  $\text{vars}(A)$  and  $\text{terms}(A)$  denote its set of variables and of terms, respectively. Given conjuncts  $F$  and  $Q$ , a *homomorphism*  $\pi$  from  $Q$  to  $F$  is a substitution of  $\text{vars}(Q)$  by terms of  $F$  s.t.  $\pi(Q) \subseteq F$  (we say that  $\pi$  maps  $Q$  to  $F$ ). First-order semantic entailment is denoted by  $\models$ . 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 (Existential Rule)** An existential rule (or simply rule when not ambiguous) is a formula  $R = \forall \mathbf{x} \forall \mathbf{y} (B[\mathbf{x}, \mathbf{y}] \rightarrow (\exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]))$  where  $B = \text{body}(R)$  and  $H = \text{head}(R)$  are conjuncts, called the body and the head of  $R$ , respectively. The frontier of  $R$ , denoted  $\text{fr}(R)$ , is the set of variables  $\text{vars}(B) \cap \text{vars}(H) = \mathbf{y}$ .

We can now omit quantifiers since there is no ambiguity.

**Definition 2 (Application of a Rule)** A rule  $R$  is applicable to a fact  $F$  if there is a homomorphism  $\pi$  from  $\text{body}(R)$  to  $F$ ; the result of the application of  $R$  to  $F$  w.r.t.  $\pi$  is a fact  $\alpha(F, R, \pi) = F \cup \pi^{\text{safe}}(\text{head}(R))$  where  $\pi^{\text{safe}}$  is a substitution of  $\text{head}(R)$ , that replaces each  $x \in \text{fr}(R)$  with  $\pi(x)$ , and each other variable with a “fresh” variable, *i.e.*, not introduced before. As  $\alpha$  only depends on  $\pi|_{\text{fr}(R)}$  (the restriction of  $\pi$  to  $\text{fr}(R)$ ), we also write  $\alpha(F, R, \pi|_{\text{fr}(R)})$ .

**Example 1** Let  $F = \{r(a, b), r(c, d), p(d)\}$  and  $R_1 = r(x, y) \rightarrow r(y, z)$ . There are two applications of  $R_1$  to  $F$ , respectively by  $h_1 = \{(x, a), (y, b)\}$  and  $h_2 = \{(x, c), (y, d)\}$ . Let  $F_1 = \alpha(F, R_1, h_1) = F \cup \{r(b, z_1)\}$ . Let  $F_2 = \alpha(F, R_1, h_2) = F \cup \{r(d, z_2)\}$ .

**Definition 3 ( $\mathcal{R}$ -derivation)** Let  $F$  be a fact, and  $\mathcal{R}$  be a set of rules. An  $\mathcal{R}$ -derivation (from  $F$  to  $F_k$ ) is a finite sequence  $(F_0 = F), (R_0, \pi_0, F_1), \dots, (R_{k-1}, \pi_{k-1}, F_k)$  s.t. for all  $0 \leq i < k$ ,  $R_i \in \mathcal{R}$  and  $\pi_i$  is a homomorphism from  $\text{body}(R_i)$  to  $F_i$  s.t.  $F_{i+1} = \alpha(F_i, R_i, \pi_i)$ . When only the successive facts are needed, we note  $(F_0 = F), F_1, \dots, F_k$ .

**Theorem 1 (Soundness and Completeness)** 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 from  $F$  to  $F_k$  s.t.  $F_k \models Q$ .

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  $\mathcal{R}$ . W.l.o.g. we assume that the rules have pairwise disjoint sets of variables. We denote by  $\mathcal{C}$  the set of constants occurring in  $(F, \mathcal{R})$  and by  $T_0$  (called the “initial terms”) the set  $\text{vars}(F) \cup \mathcal{C}$ , *i.e.*,  $T_0$  includes not only the terms from  $F$  but also the constants occurring in rules. 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?

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 (also called Gaifman graph) of this hypergraph has the same set of nodes and there is an

<sup>1</sup>Note that hereby we generalize the classical notion of a fact in order to take existential variables into account.

edge between two nodes if they belong to the same hyper-edge. The treewidth of a fact is defined as the treewidth<sup>2</sup> of its associated primal graph. Given a fact  $F_i$ , a derivation  $S$  to  $F_i$  or a tree decomposition  $\mathcal{T}$  of  $F_i$ , we note  $\text{atoms}(S) = \text{atoms}(\mathcal{T}) = F_i$ .

A set of rules  $\mathcal{R}$  is called a *bounded treewidth set (bts)* if all  $\mathcal{R}$ -derived facts have bounded treewidth, i.e., for any fact  $F$  there exists an integer  $b$  such that, for any fact  $F'$  that can be  $\mathcal{R}$ -derived from  $F$ ,  $\text{treewidth}(F') \leq b$ .

The proof of decidability of CQ entailment with *bts* relies on a result by Courcelle (Courcelle 1990), that states that classes of first-order logic having the bounded treewidth model property are decidable. It does not (at least not directly) provide a halting algorithm. Very recently, a subclass of *bts* has been defined, namely *greedy bts (gbts)*, which is equipped with a halting algorithm (Baget et al. 2011). A derivation is said to be *greedy* if, for every rule application in this derivation, all the frontier variables not being mapped to the initial terms  $T_0$  are jointly mapped to terms added by a single previous rule application. This allows to build a tree decomposition of a derived fact in a greedy way.

**Definition 4 (Greedy Derivation)** An  $\mathcal{R}$ -derivation  $F_0, \dots, F_k$  is said to be *greedy* if, for all  $i$  with  $0 < i < k$ , there is  $j \leq i$  s.t.  $\pi_i(\text{fr}(R_i)) \subseteq \text{vars}(A_j) \cup T_0$ , where  $A_j = \pi_j^{\text{safe}}(\text{head}(R_j))$  (any  $j \leq i$  can be chosen if  $\text{fr}(R_i)$  is mapped to  $T_0$ ).

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

From now on, we restrict our focus to *gbts* and we assume that  $\mathcal{R}$  denotes a *gbts* rule set.

Any greedy derivation gives rise to a derivation tree, whose root corresponds to the initial fact, and each other node corresponds to a rule application of the derivation. To each node is assigned a set of terms and a set of atoms. Note that the set of terms assigned to the root is  $T_0$ , i.e., it includes the constants that may be brought by rule applications. Moreover,  $T_0$  is included in the set of terms of all nodes. This ensures that the derivation tree is a decomposition tree of the associated derived fact.

**Definition 6 (Derivation Tree)** Let  $S = (F_0 = F), \dots, F_k$  be a greedy derivation. The derivation tree assigned to  $S$ , notation  $DT(S)$ , is a rooted tree  $\mathcal{T} = (\mathcal{B}, \text{terms}, \text{atoms}, U, \lambda)$ , where  $\mathcal{B} = \{B_0, \dots, B_k\}$  is a set of nodes, also called bags,  $U$  is the set of edges, terms and atoms are bag labeling mappings, and  $\lambda$  is an edge labeling mapping, such that:

1. the root of  $\mathcal{T}$  is  $B_0$  with  $\text{terms}(B_0) = T_0$  and  $\text{atoms}(B_0) = \text{atoms}(F)$ .
2. For  $0 < i \leq k$ , let  $R_{i-1}$  be the rule applied according to homomorphism  $\pi_{i-1}$  to produce  $F_i$ ; then  $\text{terms}(B_i) = \text{vars}(A_{i-1}) \cup T_0$  and  $\text{atoms}(B_i) = \text{atoms}(A_{i-1})$ , where  $A_{i-1} = \pi_{i-1}^{\text{safe}}(\text{head}(R_{i-1}))$ . There is an edge between  $B_i$

<sup>2</sup>We assume that the reader is familiar with this notion, see e.g. (Robertson and Seymour 1984).

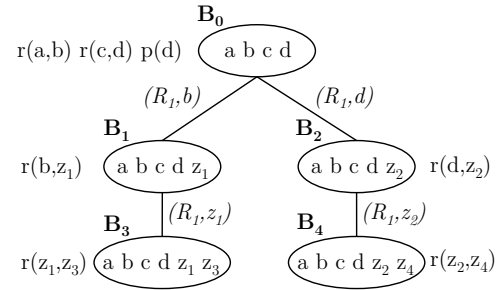


Figure 1: Derivation tree of Example 1. Only the image of the single frontier variable from  $R_1$  is mentioned in edge labels.

and the node  $B_j$  s.t.  $j$  is the smallest integer for which  $\pi_{i-1}(\text{fr}(R_{i-1})) \subseteq \text{terms}(B_j)$  (since the derivation is greedy, such a  $B_j$  always exists); this edge is labeled by  $(R_{i-1}, \pi_{i-1}|_{\text{fr}(R_{i-1})})$ .

The derivation tree is a decomposition tree of  $F_k$ , whose width is bounded by  $|T_0| + \max_{R \in \mathcal{R}}(|\text{vars}(\text{head}(R))|)$ .

**Example 1 (contd.)** See also Figure 1. We build  $DT(S)$  for  $S = (F_0 = F), (R_1, h_1, F_1), (R_1, h_2, F_2)$ . Let  $B_0$  be the root of the  $DT(S)$ .  $(R_1, h_1)$  yields a bag  $B_1$  child of  $B_0$ , with  $\text{atoms}(B_1) = \{a, b, c, d, z_1\}$  and  $\text{terms}(B_1) = \{r(b, z_1)\}$ .  $(R_1, h_2)$  yields a bag  $B_2$  with  $\text{atoms}(B_2) = \{a, b, c, d, z_2\}$  and  $\text{terms}(B_2) = \{r(d, z_2)\}$ .  $\text{fr}(R_1) = \{y\}$  and  $h_2(y) = d$ , which is both in  $\text{terms}(B_0)$  and  $\text{terms}(B_1)$ ,  $B_2$  is thus added as a child of the highest bag, i.e.,  $B_0$ .  $R_1$  can be applied again, with homomorphisms  $h_3 = \{(x, b), (y, z_1)\}$  and  $h_4 = \{(x, d), (y, z_2)\}$ , which leads to create two bags,  $B_3$  and  $B_4$ , under  $B_1$  and  $B_2$  respectively. Clearly, this can be repeated indefinitely.

Given a rooted tree  $\mathcal{T}$  and a node  $B$  in  $\mathcal{T}$ , the *subtree rooted in  $B$*  contains all descendants of  $B$  in  $\mathcal{T}$ , including  $B$ . A *prefix subtree* of a rooted tree  $\mathcal{T}$  is obtained from  $\mathcal{T}$  by deleting some of its subtrees (i.e., turning some nodes of  $\mathcal{T}$  into leaves). Given derivations  $S$  and  $S'$ , if  $S' = S.S''$  (i.e., the sequence  $S$  is a prefix of the sequence  $S'$ ) then  $DT(S)$  is a prefix subtree of  $DT(S')$ , but the converse is false.

It is not known whether *gbts* is recognizable,<sup>3</sup> however large and easily recognizable subsets of *gbts* are known. These subsets are based on the guardedness notion, inspired from guarded logic (Andréka, Németi, and van Benthem 1998), and/or on properties of rule frontiers. A rule  $R$  is said to be *guarded* if there is an atom  $a$  in its body, called a guard, that contains all the variables from the body, i.e.,  $\text{vars}(a) = \text{vars}(\text{body}(R))$  (Cali, Gottlob, and Kifer 2008). Since any rule application necessarily maps a guard to the atom of a bag from the derivation tree, it follows that all derivations with guarded rules are greedy. The guardedness constraint can be relaxed in two ways: first, by noticing that variables necessarily mapped to initial terms do not need to be guarded, we obtain *weakly guarded rules* (Cali, Gottlob, and Kifer 2008); second, by noticing that only frontier

<sup>3</sup>We conjecture that it is.

Class	$w$ unbounded	$w$ bounded	Data Comp.
<b>gbts</b>	2EXPTIME *	2EXPTIME	EXPTIME
<b>wfg, jfg</b>	2EXPTIME	2EXPTIME	EXPTIME
<b>fr1, fg</b>	2EXPTIME	2EXPTIME	PTime
<b>wg</b>	2EXPTIME	EXPTIME	EXPTIME
<b>guarded</b>	2EXPTIME	EXPTIME	PTime

\*: 2EXPTIME membership proven in this paper

Table 1: Combined and Data complexity for *gbts* classes. Upper and lower bounds coincide.

variables need to be guarded, we obtain *frontier-guarded* rules (Baget, Leclère, and Mugnier 2010); finally, both relaxations can be combined, which yields *weakly frontier-guarded* rules (Baget, Leclère, and Mugnier 2010). More precisely, a rule  $R$  is *weakly guarded* (*wg*) if there is an  $a \in \text{body}(R)$  that contains all *affected* variables from  $\text{body}(R)$ ; a variable is said to be affected if it occurs only in affected predicate positions, which are positions that may contain a new variable generated by forward chaining (this notion requires to consider the whole set of rules). The reader is referred to (Fagin et al. 2005) for a syntactic characterization of affected variables. A rule  $R$  is *frontier-guarded* (*fg*) if there is an  $a \in \text{body}(R)$  with  $\text{vars}(fr(R)) \subseteq \text{vars}(a)$ . Note that frontier-guarded rules generalize another class based on a simple property of the frontier: a rule  $R$  is *frontier-one* (*fr1*) if  $|fr(R)| = 1$  (Baget et al. 2009). A rule  $R$  is *weakly-frontier guarded* (*wfg*) if there is an  $a \in \text{body}(R)$  that contains all affected variables from  $fr(R)$ . By refining the notion of an affected variable, *wfg* can be further generalized into jointly-(frontier)-guarded (*j-(fg)*) (Krötzsch and Rudolph 2011).

Table 1 summarizes the complexity results for the above rule classes, in terms of combined complexity (i.e., w.r.t. the joint size of  $F$ ,  $\mathcal{R}$  and  $Q$ ), with unbounded or bounded predicate arity (noted  $w$ ), and of data complexity (i.e., w.r.t. the size of  $F$  only, while  $\mathcal{R}$  and  $Q$  are assumed to be fixed). Note that frontier-guarded rules form the largest known subclass of *gbts* that enjoys polynomial data complexity. That the combined complexity for *gbts* with unbounded arity is in 2EXPTIME is a novel result.

## Patterned Forward Chaining

This section focusses on *bag patterns*. We first show that forward chaining can be performed by considering solely the derivation tree endowed with bag patterns. Then we define *joins* on patterns in order to update them incrementally after each rule application.

**Definition 7 (Pattern)** A pattern of a bag  $B$  is a set of pairs  $(G, \pi)$ , where  $G$  is subset of a rule body and  $\pi$  is a partial mapping from  $\text{terms}(G)$  to  $\text{terms}(B)$ .  $G$  and  $\pi$  are possibly empty.

For any derivation  $S$ , we obtain a *patterned derivation tree*, noted  $PDT(S)$ , by enriching the derivation tree  $DT(S)$  assigning a pattern  $P(B)$  to each bag  $B$  of  $DT(S)$ .

**Definition 8 (Pattern soundness and completeness)** Let  $F_k$  be a fact obtained via a derivation  $S$  and let  $B$  be a bag

in  $PDT(S)$ .  $P(B)$  is said to be sound w.r.t.  $F_k$  if for all  $(G, \pi) \in P(B)$ ,  $\pi$  is extendible to a homomorphism from  $G$  to  $F_k$ .  $P(B)$  is said to be complete w.r.t.  $F_k$  (and  $\mathcal{R}$ ), if for any  $R \in \mathcal{R}$ , any  $sb_R \subseteq \text{body}(R)$  and any homomorphism  $\pi$  from  $sb_R$  to  $F_k$ ,  $P(B)$  contains  $(sb_R, \pi')$ , where  $\pi'$  is the restriction of  $\pi$  to the inverse image of  $\text{terms}(B)$ .  $PDT(S)$  is said to be sound and complete w.r.t.  $F_k$  if for all its bags  $B$ ,  $P(B)$  is sound and complete w.r.t.  $F_k$ .

Provided that  $PDT(S)$  is sound and complete w.r.t.  $F_k$ , a rule  $R$  is applicable to  $F_k$  iff there is a bag in  $PDT(S)$  whose pattern contains a pair  $(\text{body}(R), -)$ ; then, the bag created by a rule application  $(R, \pi)$  on  $F_k$  has parent  $B_j$  in  $DT(S)$  iff  $B_j$  is the bag in  $PDT(S)$  at the smallest depth s.t.  $P(B_j)$  contains  $(\text{body}(R), \pi')$ , with the restrictions of  $\pi'$  and  $\pi$  to  $fr(R)$  being equal. Patterns are managed as follows: (1) The pattern of  $B_0$  is the minimal sound and complete pattern with respect to  $F$ ; (2) after each addition of a bag  $B_i$ , the patterns of all bags are updated to ensure the soundness and completeness with respect to  $F_i$ . It follows that we can define a *patterned* derivation, where rule applicability is checked on patterns, and the associated sound and complete patterned derivation tree, which is isomorphic to the derivation tree associated with the (regular) derivation.

Remember that our final goal is to avoid building the current derived fact. We will now incrementally maintain sound and complete patterns by a propagation mechanism on patterns. This is where we need to consider patterns with subsets of rule bodies and not just full rule bodies. We recall that the rules have pairwise disjoint sets of variables.

**Definition 9 (Elementary Join)** Let  $B_1$  and  $B_2$  be two bags,  $e_1 = (sb_R^1, \pi_1) \in P(B_1)$  and  $e_2 = (sb_R^2, \pi_2) \in P(B_2)$  where  $sb_R^1$  and  $sb_R^2$  are subsets of  $\text{body}(R)$  for some rule  $R$ . Let  $V = \text{vars}(sb_R^1) \cap \text{vars}(sb_R^2)$ . The (elementary) join of  $e_1$  with  $e_2$ , noted  $J(e_1, e_2)$ , is defined iff for all  $x \in V$ ,  $\pi_1(x)$  and  $\pi_2(x)$  are both defined and  $\pi_1(x) = \pi_2(x)$ . Then  $J(e_1, e_2) = (sb_R, \pi)$ , where  $sb_R = sb_R^1 \cup sb_R^2$  and  $\pi = \pi_1 \cup \pi_2$ , where  $\pi_2'$  is the restriction of  $\pi_2$  to the inverse image of  $\text{terms}(B_1)$  (i.e., the domain of  $\pi_2'$  is the set of terms with image in  $\text{terms}(B_1)$ ).

Note that  $V$  may be empty. The elementary join is not a symmetrical operation since the range of the obtained mapping is included in  $\text{terms}(B_1)$ .

**Definition 10 (Join)** Let  $B_1$  and  $B_2$  be two bags with respective patterns  $P_1$  and  $P_2$ . The join of  $P_1$  with  $P_2$ , denoted  $J(P_1, P_2)$ , is the set of pairs  $J(e_1, e_2)$ , where  $e_1 = (sb_R^1, \pi_1) \in P_1$ ,  $e_2 = (sb_R^2, \pi_2) \in P_2$ ,  $sb_R^1$  and  $sb_R^2$  are subsets of  $\text{body}(R)$  for some rule  $R$ .

Note that  $P_1 \subseteq J(P_1, P_2)$  since each pair from  $P_1$  can be obtained by an elementary join with  $(\emptyset, \emptyset)$ . Similarly,  $J(P_1, P_2)$  contains all pairs  $(G, \pi)$  obtained from  $(G, \pi_2) \in P_2$  by restricting  $\pi_2$  to the inverse image of  $\text{terms}(B_1)$ .

If a pattern is sound w.r.t.  $F_{i-1}$  then it is sound w.r.t.  $F_i$ . The following property follows from the definitions:

**Property 2** If  $P_1$  and  $P_2$  are sound w.r.t.  $F_i$  then  $J(P_1, P_2)$  is sound w.r.t.  $F_i$ .

We consider now the step from  $F_{i-1}$  to  $F_i$  in a (patterned) derivation sequence: let  $B_c$  be the created bag and  $B_p$  be its parent in  $PDT(S)$ .

**Definition 11 (Initial pattern)** *The initial pattern of  $B_c$  is the set of pairs  $(G, \pi)$  s.t.  $G$  is a subset of a rule body and  $\pi$  is a homomorphism from  $G$  to  $\text{atoms}(B_c)$ .*

**Property 3 (Soundness of initial pattern of  $B_c$  w.r.t.  $F_i$ )**  
*The initial pattern of  $B_c$  is sound with respect to  $F_i$ .*

**Property 4 (Completeness of  $J(P(B_c), P(B_p))$  w.r.t.  $F_i$ )**  
*Let  $P(B_c)$  be the initial pattern of  $B_c$  and  $B_p$  be the parent of  $B_c$ . Assume that  $P(B_p)$  is complete w.r.t.  $F_{i-1}$  and  $\mathcal{R}$ . Then  $J(P(B_c), P(B_p))$  is complete w.r.t.  $F_i$ .*

*Proof:* Let  $\pi$  be a homomorphism from  $sb_R \subseteq \text{body}(R)$  to  $F_i$ , for some rule  $R$ . We show that  $J(P(B_c), P(B_p))$  contains  $(sb_R, \pi')$ , where  $\pi'$  is the restriction of  $\pi$  to the inverse image of  $\text{terms}(B_c)$ . Let us partition  $sb_R$  into  $b_{i-1}$ , the subset of atoms mapped by  $\pi$  to  $F_{i-1}$ , and  $b_i$  the other atoms from  $sb_R$ , which are necessarily mapped by  $\pi$  to  $F_i \setminus F_{i-1}$ , i.e.,  $\text{atoms}(B_c)$ . If  $b_i$  is not empty, by definition of the initial pattern,  $P(B_c)$  contains  $(b_i, \pi_c)$ , where  $\pi_c$  is the restriction of  $\pi$  to  $\text{terms}(b_i)$ . If  $b_{i-1}$  is not empty, by hypothesis (completeness of  $P(B_p)$  w.r.t.  $F_{i-1}$ ),  $P_p$  contains  $(b_{i-1}, \pi_p)$ , where  $\pi_p$  is the restriction of  $\pi|_{b_{i-1}}$  to the inverse image of  $\text{terms}(B_p)$ . If  $b_{i-1}$  or  $b_i$  is empty,  $(sb_R, \pi')$  belongs to  $J(P(B_c), P(B_p))$  (Points 1 and 2 in Def. 10). Otherwise, consider  $J((b_i, \pi_c), (b_{i-1}, \pi_p))$ : it is equal to  $(sb_R, \pi')$  (Point 3 in Def. 10).  $\square$

**Property 5 (Completeness of join-based propagation)**

*Assume that  $PDT(S)$  is complete w.r.t.  $F_{i-1}$ , and  $P(B_c)$  is computed by  $J(P_i(B_c), P(B_p))$ , where  $P_i(B_c)$  is the initial pattern of  $B_c$ . Let  $d(B)$  denote the distance of a bag  $B$  to  $B_c$  in  $PDT(S)$ . Updating a bag  $B$  consists in performing  $J(P(B), P(B'))$ , where  $B'$  is the neighbor of  $B$  s.t.  $d(B') < d(B)$ . Let  $\mathcal{T}'$  be obtained from  $PDT(S)$  by updating all bags by increasing value of  $d$ . Then  $\mathcal{T}'$  is complete w.r.t.  $F_i$ .*

*Proof:* Similar to the proof of Property 4. The crucial point is that if  $\pi$  maps an atom  $a$  of  $sb_R$  to an atom  $b$  of  $F_i \setminus F_{i-1}$ , and  $b$  shares a term  $e$  with  $B$ , then  $e \in \text{terms}(B_c)$ , hence, thanks to the running intersection property of a decomposition tree,  $e \in \text{terms}(B')$ , thus  $(e, \pi(e))$  will be propagated to  $P(B)$ .  $\square$

It follows that the following steps performed at each bag creation (where  $B_c$  is introduced as a child of  $B_p$ ) allow to maintain the soundness and completeness of the patterned DT: (1) initialize  $P(B_c)$  with its initial pattern; (2) update  $P(B_c)$  with  $J(P(B_c), P(B_p))$  (3) propagate: first, propagate from  $P(B_c)$  to  $P(B_p)$ , i.e., update  $P(B_p)$  by  $J(P(B_p), P(B_c))$ ; then, for each bag  $B$  updated from a bag  $B'$ , update its children  $B_i$  (for  $B_i \neq B'$ ) by  $J(P(B_i), P(B))$  and its parent  $B_j$  by  $J(P(B_j), P(B))$ .

## Bag Equivalence

In this section, we define an adequate relation of *equivalence* on patterns, which will allow us to develop only one bag per

equivalence class. We begin with the immediate notion of structural equivalence, then show that it has to be refined.

**Definition 12 (Structural Equivalence)** *Let  $B$  and  $B'$  be two bags in the same (partial) DT, or in two (partial) DTs, respectively created by applications  $(R, \pi_i)$  and  $(R, \pi_j)$  of the same rule  $R$ .  $B$  and  $B'$  are structurally equivalent if:*

- $\forall f, f' \in \text{fr}(R), \pi_i(f) = \pi_i(f') \Leftrightarrow \pi_j(f) = \pi_j(f')$
- $\forall a \in T_0, \forall f \in \text{fr}(R), \pi_i(f) = a \Leftrightarrow \pi_j(f) = a$

**Example 1 (contd.)** *Consider the DT in Example 1, depicted in Figure 1. Although both  $B_1$  and  $B_2$  result from  $R_1$ , they are not structurally equivalent because  $\text{fr}(R_1) = y$ ,  $h_1(y) \in T_0$ , and  $h_1(y) \neq h_2(y)$ .  $B_3$  and  $B_4$  are structurally equivalent.*

Structural equivalence is not sufficient to ensure that the “same” derivations can be carried out under equivalent bags, as shown by the next example.

**Example 1 (contd.)** *Let us add the rule  $R_2 = r(x, y) \wedge r(y, z) \wedge p(x) \rightarrow f(z)$ .  $R_2$  is applicable to  $B_4$  (i.e., with its frontier mapped to  $\text{terms}(B_4)$ ) but not to  $B_3$ .*

When  $B$  and  $B'$  are structurally equivalent, a natural bijection can be built between  $B$  and  $B'$ , which maps each initial term to itself, and each variable introduced in  $B$  to the respective variable introduced in  $B'$ . We will use this natural bijection to compare patterns and refine bag equivalence.

**Definition 13 (Natural bijection)** *Let  $B$  and  $B'$  be two structurally equivalent bags in a (partial) DT. The natural bijection from  $\text{terms}(B)$  to  $\text{terms}(B')$  (in short from  $B$  to  $B'$ ), denoted  $\psi_{B \rightarrow B'}$ , is defined as follows:*

- if  $x \in T_0, \psi_{B \rightarrow B'}(x) = x$
- otherwise, let  $\text{orig}(x) = \{u \in \text{vars}(\text{head}(R)) \mid \pi_i^{\text{safe}}(u) = x\}$ . Since  $B$  and  $B'$  are structurally equivalent,  $\forall u, u' \in \text{orig}(x), \pi_j^{\text{safe}}(u) = \pi_j^{\text{safe}}(u')$ . We define  $\psi_{B \rightarrow B'}(x) = \pi_j^{\text{safe}}(u)$ .

**Definition 14 (Pattern inclusion / equivalence)** *Let  $B$  and  $B'$  be two bags in a (partial) DT with respective patterns  $P(B)$  and  $P(B')$ . We say that  $P(B)$  includes  $P(B')$ , denoted  $P(B') \sqsubseteq P(B)$ , if:*

- $B$  and  $B'$  are structurally equivalent,
- $P(B)$  contains all elements from  $P(B')$ , up to a variable renaming given by the natural bijection:  $(G, \pi') \in P(B') \Rightarrow (G, \psi_{B' \rightarrow B} \circ \pi') \in P(B)$ .

*We say that  $P(B)$  and  $P(B')$  are equivalent, denoted  $P(B) \sim P(B')$ , if  $P(B') \sqsubseteq P(B)$  and  $P(B) \sqsubseteq P(B')$ .*

By extension, two bags are said to be *equivalent* if their patterns are equivalent. Given a derivation  $S$ , if two bags  $B$  and  $B'$  in  $DT(S)$  are equivalent, then the “same” derivations can be made *under* them (i.e., with rule applications that map the rule frontier to the subtree rooted in  $B$ , resp.  $B'$ ).

## Full Blocked Tree

We now define the notion of a *full blocked tree*, which finitely represents all the  $\mathcal{R}$ -derivations that can be performed in the KB. Informally, for every derivation  $S$ ,  $DT(S)$  can be generated from this tree by copying the root, then repeatedly copying children of unblocked nodes, while respecting structural equivalence.

**Definition 15 (Blocked Tree)** A *blocked tree* is a structure  $(\mathcal{T}_b, \sim)$ , where  $\mathcal{T}_b$  is a prefix of a patterned derivation tree and  $\sim$  is the equivalence relation on the bags of  $\mathcal{T}_b$  s.t. for each  $\sim$ -class, all but one bag are said to be blocked; this bag is called the representative of its class and is the only one that may have children.

With a blocked tree  $\mathcal{T}_b$  is associated a possibly infinite set of decomposition trees obtained by copying its bags. More precisely, this set is composed of pairs  $(\mathcal{T}, f)$ , where  $\mathcal{T}$  is a decomposition tree obtained from  $\mathcal{T}_b$  and  $f$  is a mapping from the bags of  $\mathcal{T}$  to the bags of  $\mathcal{T}_b$  such that for any  $B \in \mathcal{T}$ ,  $B$  and  $f(B)$  are structurally equivalent. We first define the bag copy operation:

**Definition 16 (Bag Copy)** Let  $B_1$  and  $B_2$  be structurally equivalent bags with natural bijection  $\psi_{B_1 \rightarrow B_2}$ . Let  $B'_1$  be a child of  $B_1$ . Copying  $B'_1$  under  $B_2$  (according to  $\psi_{B_1 \rightarrow B_2}$ ) consists in adding a child  $B'_2$  to  $B_2$ , s.t.  $\text{terms}(B'_2)$  is obtained by the following bijection  $b$ , and  $\text{atoms}(B'_2) = b(\text{atoms}(B'_1))$ : for all  $x \in \text{terms}(B'_1)$ , if  $x \in \text{terms}(B_1)$  then  $b(x) = \psi_{B_1 \rightarrow B_2}(x)$ , otherwise  $b(x)$  is a fresh variable.

**Property 6** Let  $B'_2$  be obtained by copying  $B'_1$  under  $B_2$  as in the previous definition. Let  $(R, \pi)$  be the label of the edge  $(B_1, B'_1)$ . Then  $B'_2$  can be obtained by applying  $R$  to  $B_2$  w.r.t.  $\psi_{B_1 \rightarrow B_2} \circ \pi$  (up to fresh variable renaming). Moreover,  $B'_1$  and  $B'_2$  are structurally equivalent and  $\psi_{B'_1 \rightarrow B'_2} = b$ .

**Definition 17 (Set of Trees generated by a Blocked Tree)** With a blocked tree  $\mathcal{T}_b$  is associated a set  $G(\mathcal{T}_b)$  inductively defined as follows:

- The pair  $(\mathcal{T}_b[\text{root}], \text{identity})$ , where  $\mathcal{T}_b[\text{root}]$  is the restriction of  $\mathcal{T}_b$  to its root, belongs to  $G(\mathcal{T}_b)$ .
- Given a pair  $(\mathcal{T}, f) \in G(\mathcal{T}_b)$ , let  $B$  be a bag in  $\mathcal{T}$ , and  $B' = f(B)$ ; let  $B'_r$  be the representative of  $B' \sim$ -class ( $B'_r \neq B'$  if  $B'$  is blocked) and  $B'_c$  be a child of  $B'_r$ . If  $B$  has no child structurally equivalent to  $B'_c$ , let  $\mathcal{T}_{\text{new}}$  be obtained from  $\mathcal{T}$  by copying  $B'_c$  under  $B$  (according to  $\psi_{B'_r \rightarrow B}$ ), which yields a new bag  $B_c$ . Then  $(\mathcal{T}_{\text{new}}, f \cup (B'_c, B_c))$  belongs to  $G(\mathcal{T}_b)$ .

For each pair  $(\mathcal{T}, f) \in G(\mathcal{T}_b)$ ,  $\mathcal{T}$  is said to be generated by  $\mathcal{T}_b$  via  $f$ .

Note that a generated decomposition tree is not necessarily a derivation tree, but it is a prefix of a derivation tree.

**Definition 18** A full blocked tree  $\mathcal{T}^*$  (of  $F$  and  $\mathcal{R}$ ) is a blocked tree satisfying the two following properties:

- (Soundness) If  $\mathcal{T}'$  is generated by  $\mathcal{T}^*$ , then there is  $\mathcal{T}''$  generated by  $\mathcal{T}^*$  and an  $\mathcal{R}$ -derivation  $S$  from  $F$  such that  $\text{atoms}(\mathcal{T}'') = \text{atoms}(S)$  (up to fresh variable renaming) and  $\mathcal{T}'$  is a prefix subtree of  $\mathcal{T}''$ .

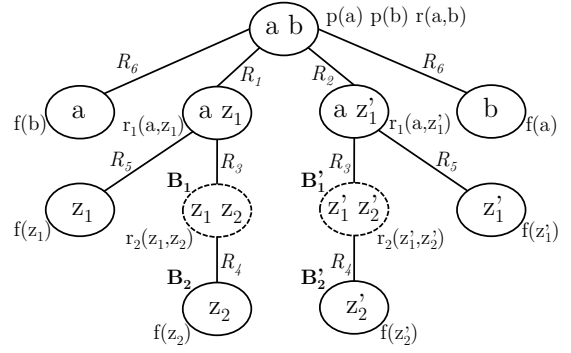


Figure 2: The yoyo example (Example 2) – Partial drawing:  $a, b$  omitted in some bags,  $\pi$  not drawn on edges.

- (Completeness) For all  $\mathcal{R}$ -derivations from  $F$ ,  $DT(S)$  is generated by  $\mathcal{T}^*$ .

## Building a Full Blocked Tree

To build a full blocked tree, the algorithm starts from a single bag corresponding to  $F$ . Rules are applied on the bag level, i.e., by considering only bag patterns. Patterns are updated by means of join propagation. For each bag equivalence class, all but one of the bags are blocked, which means that no existential rule can be applied to these bags. However, in order to obtain a full blocked tree, we cannot simply block bags, as shown by the next example.

**Example 2 (yoyo rules)** Let  $F = \{p(a), p(b), r(a, b)\}$ . Consider the following rules:

$$\begin{aligned} R_1: r(x, y) &\rightarrow r_1(x, z) & R_4: r_2(x, y) &\rightarrow f(y) \\ R_2: r(x, y) &\rightarrow r_1(y, z) & R_5: r_2(x, y) \wedge f(y) &\rightarrow f(x) \\ R_3: r_1(x, y) &\rightarrow r_2(y, z) & R_6: r_1(x, y) \wedge f(y) &\rightarrow f(x) \end{aligned}$$

Figure 2 shows the DT that should be obtained. In particular, the atoms  $f(a)$  and  $f(b)$  are produced. Assume now that patterned forward chaining is performed, and that  $B'_2$  is created just after  $B_2$ . Since these bags are equivalent,  $B'_2$  is blocked. However, to apply  $R_5$  to  $B'_1$  to produce  $f(z'_1)$  (i.e., mapping the frontier variable  $x$  to  $z'_1$ ) – which will then allow to produce  $f(b)$ , we need to have the pair  $(r_2(x, y) \wedge f(y), \{(x, z'_1)\})$  in  $P(B'_1)$ .  $(r_2(x, y), \{(x, z'_1)\})$  belongs to  $P(B'_1)$  but it cannot be expanded into  $(r_2(x, y) \wedge f(y), \{(x, z'_1)\})$  since  $R_4$  is not applied to  $z'_2$  ( $B'_2$  is blocked). This shows that the pattern of a blocked bag has to evolve “as if the derivation subtree rooted in this bag was built”, in order to correctly apply the rules on its ancestors.

Therefore, we introduce a set  $\Gamma$  of *pseudo-rules* to propagate pattern updates. A pseudo-rule has the form  $P \rightarrow P'$  (“ $P$  evolves to  $P'$ ”), where  $P$  and  $P'$  are patterns. It is applicable to any bag  $B$  with  $P(B) \sim P$  and its effect is to replace  $P(B)$  by  $P'$ . All the rules in  $\Gamma$  are “justified”, in the sense that for all bags  $B$ , if  $P(B)$  is equivalent to  $P$ , then there is a patterned derivation s.t. the pattern of  $B$  at the end of the derivation contains  $P'$ .

The sub-algorithms UPDATEPARENT and UPDATECHILD perform join propagation, respectively from a bag to its parent, and from a bag to one of its children. The reason for having two procedures instead of one is that only UPDATEPARENT can create new pseudo-rules.

In addition to the derivation tree  $\mathcal{T}$  and the set of pseudo-rules  $\Gamma$ , we use a structure  $\mathcal{B}$ , which allows to assign to each bag its pattern and to create pseudo-rules.  $\mathcal{B}$  is a set of pairs of lists  $(L_p, L_b)$  s.t.  $L_p$  is a list of pattern equivalence classes and  $L_b$  is a list of bags, whose patterns belong to  $L_p$ . We maintain the following properties:

- the head of  $L_p$  includes all other patterns of  $L_p$ ,
- the head of  $L_b$  is the representative of the bags with pattern the head of  $L_p$ ; it is unblocked, while all other bags in  $L_b$  are blocked;
- if  $B \in L_b$ , its pattern can evolve (by an appropriate sequence of pseudo-rule applications) to a pattern equivalent to the head of  $L_p$ .

---

### Algorithm 1: FULLBLOCKEDTREE

---

**Data:** A fact  $F$ , a gpbs rule set  $\mathcal{R}$   
**Result:** A full blocked tree of  $(F, \mathcal{R})$   
Let  $\mathcal{T}$  be the decomposition tree initialized with a single bag  $B_0$ , s.t.  $terms(B_0) = T_0$ ,  $atoms(B_0) = F$  and  $P(B_0)$  is its initial pattern  
 $\mathcal{B} = \{([P(B_0)], [B_0])\}$ ;  
 $\Gamma = \emptyset$ ;  
**while** There is a new rule application  $(R, \pi)$  on an unblocked node  $B_p$  **do**  
    Apply  $(R, \pi)$ , creating  $B_c$  (with  $P(B_c)$  its initial pattern);  
    UPDATECHILD( $\mathcal{T}, \mathcal{B}, \Gamma, B_c, B_p$ );  
    UPDATEPARENT( $\mathcal{T}, \mathcal{B}, \Gamma, B_p, B_c$ ); //  $\Gamma$  may grow  
    **while** There is a pseudo-rule  $P(B) \rightarrow P'$  applicable on a bag  $B$  **do**  
        Perform this application  
        UPDATEPARENT( $\mathcal{T}, \mathcal{B}, \Gamma, B, Parent(B)$ )  
**return**  $\mathcal{T}, \mathcal{B}$

---

We use below the following notations:

- for any bag  $B$ ,  $L_b(B)$  denotes the unique list containing  $B$ .  $L_p(B)$  denotes the list of patterns associated with  $L_b(B)$ ,
- the dual notation is used for  $L_p(P)$  and  $L_b(P)$ .

When we remove a bag  $B$  from the tree, we remove it also from  $L_b(B)$ . If it becomes empty,  $(L_p, L_b)$  is deleted.

Let  $s$  be the maximum size of a pattern,  $p$  the number of patterns,  $f$  the maximum size of a rule frontier,  $b$  the maximum size of a bag.

**Property 7** Algorithm 1 terminates in polynomial time in the number of patterns.

*Proof:* The maximum number of bags  $n_b$  in a blocked tree is upper-bounded by  $p(1 + |\mathcal{R}|b^f)$ . The maximum size of all patterns in a blocked tree is upper-bounded by  $n_b \times s$ . As long as no bag is removed from  $\mathcal{T}$ , at least one of the following parameters increases at each loop iteration:

---

### Algorithm 2: UPDATECHILD

---

**Data:**  $\mathcal{T}, \mathcal{B}, \Gamma, B_c$  and  $B_p$ , two bags of  $\mathcal{T}$ , where  $B_c$  is a child of  $B_p$   
**Result:**  $\mathcal{T}$  and  $\mathcal{B}$  updated  
Let  $P = JOIN(P(B_c), P(B_p))$ ;  
**if**  $P \neq P(B_c)$  **then**  
    Remove  $B_c$  from  $L_b(B_c)$ ;  
    **if**  $P$  appears in  $\mathcal{B}$  **then**  
        Remove the descendants of  $B_c$ ;  
        Add  $B_c$  in  $L_b(P)$  (at any but first position);  
        //  $B_c$  is blocked  
    **else**  
        Add  $([P], [B_c])$  to  $\mathcal{B}$ ; //  $B_c$  is unblocked  
    **for** any child  $B'_c$  of  $B_c$  **do**  
        UPDATECHILD( $\mathcal{T}, \mathcal{B}, \Gamma, B'_c, B_c$ )

---



---

### Algorithm 3: UPDATEPARENT

---

**Data:**  $\mathcal{T}, \mathcal{B}, \Gamma, B_p$  and  $B_c$ , two bags of  $\mathcal{T}$ , where  $B_c$  is a child of  $B_p$   
**Result:**  $\mathcal{T}, \mathcal{B}$  and  $\Gamma$  updated  
Let  $P = JOIN(P(B_p), P(B_c))$ ;  
**if**  $P \neq P(B_p)$  **then**  
    **for** any  $P' \in L_p(B_p)$  **do**  
         $\Gamma = \Gamma \cup \{P' \rightarrow P\}$   
    **if**  $P$  appears in  $\mathcal{B}$  **then**  
        Remove the descendants of  $B_p$ ;  
        Append  $L_p(B_p)$  at the end of  $L_p(P)$ ;  
        Append  $L_b(B_p)$  at the end of  $L_b(P)$ ;  
        //  $B_p$  is blocked  
    **else**  
        Add  $P$  as head of  $L_p(B_p)$ ;  
        //  $B_p$  remains unblocked  
    UPDATEPARENT( $\mathcal{T}, \mathcal{B}, \Gamma, Parent(B_p), B_p$ )  
    **for** any child  $B'_c \neq B_c$  of  $B_p$  **do**  
        UPDATECHILD( $\mathcal{T}, \mathcal{B}, \Gamma, B'_c, B_p$ )

---

either the number of bags, or the sum of the sizes of the patterns. However, during calls to either UpdateParent or UpdateChild, these parameters might decrease (due to descendants removal). We refine then the study by noticing that UPDATECHILD can only delete a bag if it has been called by UPDATEPARENT. Moreover, UPDATEPARENT has an effect only if it creates a new pseudo-rule, and there are at most  $p^2$  of them, where  $p$  is the number of patterns. Thus, at most  $(p^2 + 1)(n_b + n_b s)$  loop iterations are performed. The cost of an inner loop iteration is linear in the size of the decomposition tree, i.e., polynomial in  $p$ . In the end, the algorithm runs in polynomial time in the number of patterns.  $\square$

**Theorem 8** Algorithm 1 outputs a full blocked tree of  $(F, \mathcal{R})$ .

*Proof:* (sketch) Let  $\mathcal{T}^*$  be the blocked tree produced by the algorithm. *Completeness:* We prove by induction on the length of a derivation  $S$  that  $DT(S) \in G(\mathcal{T}^*)$ . *Correctness:*



We prove by induction on the number of bags in  $T$  that for any  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$  there is  $S$  s.t.  $\mathcal{T}$  is isomorphic (by  $\Psi$ ) to a prefix of  $DT(S)$ , and, for any bag  $B$  in  $\mathcal{T}$ ,  $P(\Psi(B))$  in  $DT(S)$  includes  $P(f(B))$  in  $\mathcal{T}^*$ .  $\square$

Summing up, a full blocked tree is built in polynomial time in the number of patterns. The number of pairs in a pattern is upper-bounded by  $2^{a_B} \times b^{t_B}$  (where  $a_B$  (resp.  $t_B$ ) is the maximal number of atoms (resp. terms) in a rule body).  $p$  is thus a double exponential in  $F$  and  $\mathcal{R}$ , which drops to a single exponential when  $\mathcal{R}$  is fixed. This yields a novel upper bound for the combined complexity of CQ entailment. Indeed,  $Q$  can be considered as a rule  $Q \rightarrow match$  (where  $match$  is a new 0-ary predicate). Then,  $F, \mathcal{R} \models Q$  iff a pair  $(Q, \pi)$  (with any  $\pi$ ) appears in a bag pattern from the full blocked tree.

**Theorem 9** *CQ entailment for gbts is in 2EXPTIME for combined complexity.*

*Proof:* Follows from the preceding remark and Theorem 8.  $\square$

### Querying the Full Blocked Tree

As noted in the preceding section, building a full blocked tree is sufficient to solve CQ entailment. However, we would like to process  $Q$  independently from the rules, so that the full blocked tree built from the KB can be reused for any query. Next to providing a tight bound for query complexity, this would allow to design an algorithm that precompiles the KB offline in order to speed up the online query answering step. To this end, in this section, we define an extension of the notion of homomorphism, called  $*$ -homomorphism, s.t. there is a homomorphism from  $Q$  to a fact derived from  $(F, \mathcal{R})$  iff there is a  $*$ -homomorphism from  $Q$  to  $\mathcal{T}^* = \text{FULLBLOCKEDTREE}(F, \mathcal{R})$ .

From the soundness and completeness properties of  $\mathcal{T}^*$ , it follows that  $F, \mathcal{R} \models Q$  iff there is a homomorphism from  $Q$  to (the atoms of) some tree in  $G(\mathcal{T}^*)$ . However,  $G(\mathcal{T}^*)$  being potentially infinite, we will have to ensure that only a bounded finite part of it needs to be explored.

A homomorphism from  $Q$  to some tree  $(\mathcal{T}, f)$  in  $G(\mathcal{T}^*)$  induces a partition of the atoms of  $Q$  (two atoms are in the same set if they are mapped to the same bag), and the tree structure over these bags induces a tree structure over the subsets of the partition. A *pre- $*$ -homomorphism* encodes such a possible structure over  $Q$ , along with the mapping from the partition sets to bags of  $\mathcal{T}^*$ .

**Definition 19 (Pre- $*$ -homomorphism)** *Let  $Q$  be a query and  $\mathcal{T}^*$  be a full blocked tree of  $(F, \mathcal{R})$ . A pre- $*$ -homomorphism from  $Q$  to  $\mathcal{T}^*$  is a tuple  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_p, \pi_p))$  where:*

- the  $Q_i$  are pairwise disjoint (possibly empty) subsets of  $Q$  s.t.  $\cup_{1 \leq i \leq p} Q_i = Q$ ;
- $\mathcal{T}_\Pi$  is a rooted tree whose nodes are the  $Q_i$ ;
- $f_\Pi$  maps each node of  $\mathcal{T}_\Pi$  to a bag of  $\mathcal{T}^*$ ;
- if  $Q_i$  is empty, we define  $\text{terms}(Q_i)$  as the set of terms shared by at least two of its children;

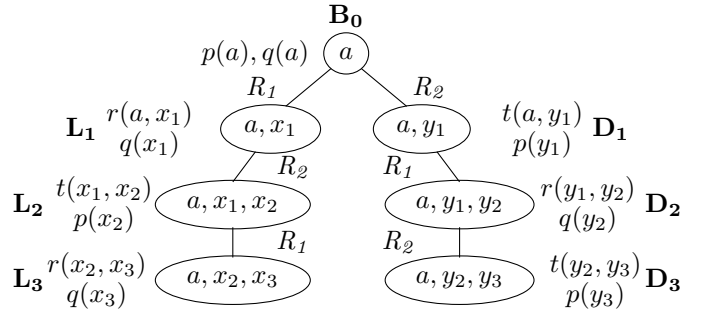


Figure 3: The full blocked tree generated by Algorithm 1 (Example 3);  $L_2$  and  $D_3$  are equivalent, as well as  $L_3$  and  $D_2$

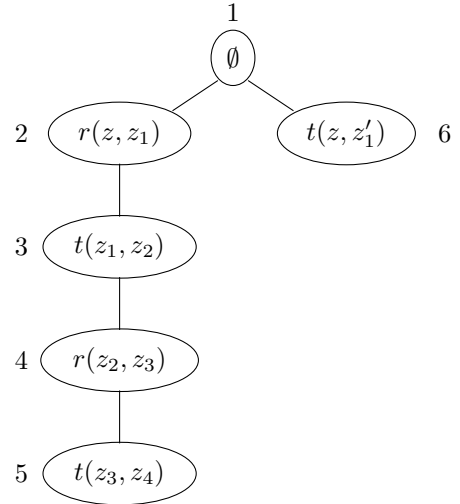


Figure 4:  $\mathcal{T}_{\Pi^1} = \mathcal{T}_{\Pi^2}$  (Example 3)

- $\forall 1 \leq i \leq p$ ,  $\pi_i$  is a substitution from  $\text{terms}(Q_i)$  to  $\text{terms}(f_\Pi(Q_i))$ . If, moreover,  $Q_i$  is not empty, then  $\pi_i$  is a homomorphism from  $Q_i$  to  $\text{atoms}(f_\Pi(Q_i))$ .

**Example 3** *Let  $F = \{p(a), q(a)\}$  and  $\mathcal{R} = \{R_1 : p(x) \rightarrow r(x, y) \wedge q(y); R_2 : q(x) \rightarrow t(x, y) \wedge p(y)\}$ . The full blocked tree output by Algorithm 1 is drawn in Figure 3. Let  $Q = \{r(z, z_1), t(z_1, z_2), r(z_2, z_3), t(z_3, z_4), t(z, z'_1)\}$ . Among the pre- $*$ -homomorphisms are the following two:*

- $\mathcal{T}_{\Pi^1} = \mathcal{T}_{\Pi^2}$ , as drawn in Figure 4,
- $f_{\Pi^1}(1) = f_{\Pi^2}(1) = B_0, f_{\Pi^1}(2) = f_{\Pi^2}(2) = L_1, f_{\Pi^1}(3) = f_{\Pi^2}(3) = L_2, f_{\Pi^1}(4) = f_{\Pi^2}(4) = L_3, f_{\Pi^1}(6) = f_{\Pi^2}(6) = D_1,$
- $f_{\Pi^1}(5) = D_3, f_{\Pi^2}(5) = D_1,$
- for  $i = 1, 2, \pi_1^i(z) = \pi_2^i(z) = \pi_6^i(z) = a, \pi_2^i(z_1) = \pi_3^i(z_1) = x_1, \pi_3^i(z_2) = \pi_4^i(z_2) = x_2, \pi_4^i(z_3) = x_3, \pi_6^i(z'_1) = y_1,$
- $\pi_5^1(z_3) = y_2, \pi_5^1(z_4) = y_3, \pi_5^2(z_3) = a, \pi_5^2(z_4) = y_1$

With a pre- $*$ -homomorphism, we have fixed, relying only upon  $Q$  and  $\mathcal{T}^*$ , what will hopefully become a homomorphism from  $Q$  to some tree in  $G(\mathcal{T}^*)$ . Note that the empty  $Q_i$  are used to encode the possible joins of their children.

**Definition 20 ( $*$ -homomorphism)** Let  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_p, \pi_p))$  be a pre- $*$ -homomorphism from  $Q$  to  $\mathcal{T}^*$ . We say that  $\Pi$  is a  $*$ -homomorphism if there exists  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$  and an injective mapping  $\omega$  (called a  $*$ -proof to  $(\mathcal{T}, f)$ ) from the nodes of  $\mathcal{T}_\Pi$  to the bags of  $\mathcal{T}$  such that:

- $Q_i$  is a descendant of  $Q_j$  in  $\mathcal{T}_\Pi$  iff  $\omega(Q_i)$  is a descendant of  $\omega(Q_j)$  in  $\mathcal{T}$ ;
- for every node  $Q_i$  in  $\mathcal{T}_\Pi$ ,  $f_\Pi(Q_i) = f(\omega(Q_i))$ ;
- $\forall t \in \text{terms}(Q_i)$ , we note  $\pi_i^\omega(t) = \psi_{f_\Pi(Q_i) \rightarrow \omega(Q_i)}(\pi_i(t))$ . Then for every term  $t$  shared by a node  $Q_i$  and its parent  $Q_j$ , we have  $\pi_i^\omega(t) = \pi_j^\omega(t)$ .

**Example 3 (contd.)** The pre- $*$ -homomorphism  $\Pi^1$  given in the previous example is actually a  $*$ -homomorphism: we build  $\mathcal{T}$  by copying  $D_3$  below  $L_3$  (creating  $L_4$ ) and we define  $\omega(1) = B_0, \omega(2) = L_1, \omega(3) = L_2, \omega(4) = L_3, \omega(5) = L_4, \omega(6) = D_1$ .

$\Pi^2$  is not a  $*$ -homomorphism: since  $D_1$  cannot be copied under any node and 5 is mapped to  $D_1$ , all ancestors of 5 should be mapped to  $B_0$ , which is not the case.

Let us first point out that, if there exists a  $*$ -homomorphism from  $Q$  to  $\mathcal{T}^*$ , then there exists a  $*$ -homomorphism  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_q, \pi_q))$  from  $Q$  to  $\mathcal{T}^*$  such that:

- the leaves of  $\mathcal{T}_\Pi$  are not empty and any empty node has at least 2 children;
- a term  $t$  of  $Q_i$  is called *fixed* if  $\pi_i(t) \in T_0$ ; in that case, for every  $Q_j$  in which  $t$  appears one has  $\pi_j(t) = \pi_i(t)$ ;
- for any non-fixed term  $t$ , the nodes  $Q_i$  containing  $t$  form a connected component of  $\mathcal{T}_\Pi$ .

A pre- $*$ -homomorphism that fulfills these additional conditions is called *reduced* (since any  $*$ -homomorphism  $\Pi$  can be put into this smaller form  $\Pi'$ , we only need to check reduced pre- $*$ -homomorphisms). Note already that there are at most  $(q^2 \times t \times a)^q$  reduced pre- $*$ -homomorphisms from  $Q$  to  $\mathcal{T}^*$ , where  $q = |Q|$ ,  $t = |\mathcal{T}^*|$ , and  $a = |\text{atoms}(\mathcal{T}^*)|$ .

**Property 10** There exist  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$  and a homomorphism from  $Q$  to the atoms of  $\mathcal{T}$  iff there exists a  $*$ -homomorphism from  $Q$  to  $\mathcal{T}^*$ .

**Note JFB:** Here is the proof of the property, rewritten to match the new definitions.

*Proof:*

- ( $\Rightarrow$ ) Suppose a homomorphism  $\pi$  from  $Q$  to the atoms of  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$ . Consider a minimal set of bags  $B_1, \dots, B_n$  of  $\mathcal{T}$  s.t.  $\pi(Q) \subseteq \cup_{1 \leq i \leq n} \text{atoms}(B_i)$ . We complete recursively these bags to a set of bags  $B_1, \dots, B_n, B_{n+1}, \dots, B_p$  by adding the lower common ancestors of the bags already present. We can now exhibit a partition  $Q_1, \dots, Q_n$  of  $Q$  such that,  $\forall 1 \leq i \leq n$ ,  $\pi(Q_i) \subseteq B_i$ . We complete it with empty sets  $Q_{n+1}, \dots, Q_p$ , and can now fix,  $\forall 1 \leq i \leq p$ ,

$\omega(Q_i) = B_i$ . We define  $f_\Pi(Q_i) = f(B_i)$  and,  $\forall 1 \leq i \leq n$ , for every term  $t \in Q_i$ , we define  $\pi_i(t) = \psi_{f_\Pi(Q_i) \rightarrow B_i}(\pi(t))$ . The tree  $T_i$  is built from the nodes  $Q_i$  to satisfy the property " $Q_i$  is a descendant of  $Q_j$  in  $\mathcal{T}_\Pi$  iff  $\omega(Q_i)$  is a descendant of  $\omega(Q_j)$  in  $\mathcal{T}$ ". Finally,  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_p, \pi_p))$  is a pre- $*$ -homomorphism from  $Q$  to  $\mathcal{T}^*$ , and  $(\mathcal{T}, f)$  and  $\omega$  is a  $*$ -proof that  $\Pi$  is a  $*$ -homomorphism.

- ( $\Leftarrow$ ) Let us consider a  $*$ -homomorphism  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_p, \pi_p))$  from  $Q$  to  $\mathcal{T}^*$ . Then there exists  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$  and  $\omega$  that satisfy Def. 20. For every term  $t$  appearing both in a  $Q_i$  and a  $Q_j$ , we know that  $\pi_i^\omega(t) = \pi_j^\omega(t) = t'$ . We define  $\pi(t) = t'$ . Then  $\pi$  is a homomorphism from  $Q$  to the atoms of  $\mathcal{T}$ . □

The latter property provides us with the skeleton of a brute-force algorithm to solve CQ entailment. Indeed, it is sufficient to generate all reduced pre- $*$ -homomorphisms from  $Q$  to  $\mathcal{T}^*$ , and check if one of them is a  $*$ -homomorphism (or, in the case of a nondeterministic algorithm, we guess and check the right  $\Pi$ ). To show for some  $\Pi$  that it is a  $*$ -homomorphism, we explore the tree  $\mathcal{T}_\Pi$  (in a breadth-first manner). At each step of the exploration, we have explored a prefix tree  $\mathcal{T}'_\Pi$  of  $\mathcal{T}_\Pi$ . Let  $Q_{i_1}, \dots, Q_{i_p}$  be the  $Q_i$  appearing in  $\mathcal{T}'_\Pi$ , and  $Q' = Q_{i_1} \cup \dots \cup Q_{i_p}$ . Then  $\Pi' = ((\mathcal{T}'_\Pi, f_{\Pi|_{\mathcal{T}'_\Pi}}), (Q_{i_1}, \pi_{i_1}), \dots, (Q_{i_p}, \pi_{i_p}))$  is a pre- $*$ -homomorphism from  $Q'$  to  $\mathcal{T}^*$  (we call it the pre- $*$ -homomorphism generated by  $\mathcal{T}'_\Pi$ ). We consider that, having explored  $\mathcal{T}'_\Pi$ , we have exhibited a  $*$ -proof, i.e., some mapping  $\omega'$  to  $(\mathcal{T}', f') \in G(\mathcal{T}^*)$  that respects the criteria of Definition 20.

Now let us select another node  $Q_i$ , child in  $\mathcal{T}_\Pi$  of a node  $Q_j$  of  $\mathcal{T}'_\Pi$ . An *expansion* of  $(\Pi', (\mathcal{T}', f'), \omega')$  to  $Q_i$  is a triple  $(\Pi'', (\mathcal{T}'', f''), \omega'')$  where  $\Pi''$  is a pre- $*$ -homomorphism generated by  $\mathcal{T}'_\Pi \cup \{Q_i\}$ ,  $\mathcal{T}'' \in G(\mathcal{T}^*)$  is built from  $\mathcal{T}'$  by adding a finite branch from  $\omega'(Q_j)$ ,  $\omega''$  extends  $\omega'$  by stating that  $\omega''(Q_i)$  is the leaf  $B_i$  of the added branch, and  $f''(B_i) = f_\Pi(Q_i)$ . Moreover, we want the mapping  $\omega''$  to the bags of  $(\mathcal{T}'', f'')$  to be a  $*$ -proof of  $\Pi''$ : only the last item from Definition 20 remains to be checked.

An important feature is that the *expansion* procedure is greedy, thus does not require backtracking. Suppose that there exists a  $*$ -proof  $\omega$  to  $(\mathcal{T}, f)$  that  $\Pi$  is a  $*$ -homomorphism, and that we have exhibited a  $*$ -proof  $\omega'$  to  $(\mathcal{T}', f')$  that some prefix  $\Pi'$  of  $\Pi$  is a  $*$ -homomorphism (where  $\mathcal{T}'$  is not necessarily a prefix of  $\mathcal{T}$ ). Then the  $*$ -proof  $\omega'$  can be expanded to a  $*$ -proof  $\omega''$  to  $(\mathcal{T}'', f'')$  that  $\Pi$  is a  $*$ -homomorphism (sketch of proof: for every leaf  $B'_i$  of  $\mathcal{T}'$ , the bag  $B_i = \omega(\omega'^{-1}(B'_i))$  is equivalent to  $B'_i$ , thus the subtree of  $\mathcal{T}$  rooted in  $B_i$  can be merged with  $B'_i$  to obtain the tree  $\mathcal{T}''$ ).

It remains to bound the length of a branch necessary to yield an expansion.

**Definition 21 (Reachable state)** Let  $B$  a bag of  $\mathcal{T}^*$ . A reachable state from  $B$  is a tuple  $(B', \sigma)$  where  $B'$  is a bag of

$\mathcal{T}^*$  and  $\sigma$  is a partial mapping from  $\text{terms}(B')$  to  $\text{terms}(B)$  s.t. there exists  $(\mathcal{T}, f) \in G(\mathcal{T}^*)$  that contains a bag  $B_1$  with  $f(B_1) = B$ , a descendant  $B_2$  of  $B_1$  with  $f(B_2) = B'$ , and for each  $t$  in  $\text{terms}(B')$ ,  $\psi_{B' \rightarrow B_2}(t) = \psi_{B \rightarrow B_1}(\sigma(t))$ . We say that  $B_2$  is in state  $(B', \sigma)$  from  $B$ .

Intuitively, a reachable state is an equivalence class for bags that can be generated under some  $B$  (or some copy of  $B$ ) and that join their terms to those of  $B$  in the same way.

There are at most  $t \times f^f$  different reachable states from a bag  $B$  (with  $t = |\mathcal{T}^*|$  and  $f$  being the maximal size of a rule frontier). We can write an algorithm that generates all reachable states from  $B$  in time  $t \times (t \times f^f)^2$  (by maintaining a boolean matrix  $M$  s.t.  $M_{i,j} = \text{true}$  iff there is a path from  $B$  that contains first a bag  $B'$  in state  $i$  from  $B$ , then a bag in state  $j$  from  $f(B')$ ).

The following property ensures that reachable states can be effectively used to compute  $*$ -proofs.

**Property 11** A pre- $*$ -homomorphism  $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \dots, (Q_p, \pi_p))$  from  $Q$  to  $\mathcal{T}^*$  is a  $*$ -homomorphism iff there exist mappings  $\tau_1, \dots, \tau_p$  where  $\tau_i$  maps  $Q_i$  (with parent  $Q_j$ ) to a reachable state  $(f_\Pi(Q_i), \sigma)$  from  $f_\Pi(Q_j)$  such that:

- for the root  $Q_1$  of  $\mathcal{T}_\Pi$ ,  $\tau_1 = \pi_1$ ;
- if  $\tau_j$  is defined for some  $Q_j$ ,  $Q_i$  is a child of  $Q_j$ , and  $\tau(Q_i) = (f_\Pi(Q_i), \sigma)$ , then for every  $t \in \text{terms}(Q_i)$ , if  $\pi_i(t)$  belongs to the domain of  $\sigma$ , we define  $\tau_i(t) = \sigma(\pi_i(t))$ , otherwise  $\tau_i(t) = t$ .
- for all term  $t$  shared by two nodes  $Q_i$  and  $Q_j$ , we have  $\tau_i(t) = \tau_j(t)$ .

*Proof:* TO DO JFB

□

The overall algorithm deciding whether  $F, \mathcal{R} \models Q$  can now be sketched as follows: We first build the full blocked tree  $\mathcal{T}^*$  of  $(F, \mathcal{R})$ , then compute all reachable states from all bags in  $\mathcal{T}^*$  (in time  $t^2 \times (t \times f^f)^2$ ). These two operations can be performed offline. We consider now, online, the query  $Q$ . For each reduced pre- $*$ -homomorphism  $\Pi$  from  $Q$  to  $\mathcal{T}^*$  (there are at most  $(q^2 \times t \times a)^q$  such tuples), we greedily try to assign a satisfying reachable state (as in Property 11) to each node of  $\mathcal{T}_\Pi$  (in time  $q \times (t \times f^f)$ ). The querying part is thus polynomial in  $\mathcal{T}^*$ , and simply exponential in  $Q$ . Since  $\mathcal{T}^*$  is in the worst-case double exponential w.r.t.  $F$  and  $\mathcal{R}$ , the algorithm runs in 2EXPTIME. Last, given a (non-deterministically guessed) reduced pre- $*$ -homomorphism, we can check in polynomial time (if  $\mathcal{R}$  and  $F$  are fixed) that it is indeed one, yielding:

**Theorem 12** CQ entailment for gbts is NP-complete for query complexity.

Thereby, the lower bound comes from the well-known NP-complete query complexity of plain (i.e., rule-free) CQ entailment.

## Complexity of the Algorithm on gbts Subclasses

We point out here that the algorithm can be adapted to specific gbts rules with smaller complexity (see Table 1) in order to maintain worst-case optimality.

The combined complexity for (weakly) guarded rules drops down to EXPTIME in the bounded arity case. For these kinds of rules, a rule application necessarily maps all the terms of a rule body to terms occurring in a single bag. If we store all the possible mappings of a rule body atom, we are able to construct any homomorphism from a rule body to the current fact. The blocking procedure is then unchanged. Since there are at most  $|\text{terms}(\text{bag})|^w$  such homomorphisms for an atom, the number of (non-equivalent) patterns is simply exponential in the input ( $w$  being fixed).

The data complexity for guarded, *fg* and *frl* rules drops down to PTIME. To get that bound, we slightly modify the content of a bag. Instead of including in each bag all initial terms ( $T_0$ ) in addition to the terms occurring in the atoms created by the rule application, we just take in account the latter. Since all terms to which the frontier is mapped are arguments of the same atom, we can still build a correct decomposition tree this way. The number of patterns is then upper-bounded by  $1 + 2^{|\mathcal{R}| \times 2^{a_B} \times t_H^{t_B}}$  (where  $a_B$  (resp.  $t_B$ ) is the maximal number of atoms (resp. terms) in a rule body and  $t_H$  is the maximal number of terms in a rule head). When  $\mathcal{R}$  is fixed, this number is polynomial. The querying part being polynomial when  $Q$  is fixed, we get the PTIME upper-bound.

## Conclusion

In this paper we introduced a query answering algorithm that is uniformly applicable to existential rule fragments that qualify as greedy bounded treewidth sets. The algorithm gives rise to novel tight complexity bounds w.r.t. combined and query complexity and is also worst-case optimal for data complexity. A slight modification of the algorithm allowed us to maintain worst-case optimality for certain easier gbts subclasses.

Compared to a predecessor version, the algorithm comes with several theoretically and practically advantageous features:

First, it provides a way of finitely representing infinite models by means of a specific data-structure. This data-structure and the way it is obtained exhibits similarities to tableaux procedures in Description Logics (Baader et al. 2007) with the notable exception that instead of single individuals, whole bags are created, updated, and blocked. Conceptually, the applied blocking technique is reminiscent of *anywhere blocking* (Buchheit, Donini, and Schaerf 1993; Motik, Shearer, and Horrocks 2009) in that the blocking entity does not need to be a predecessor in the tree.

Second, our algorithm allows for pre-compilation of the fact and rule set (i.e. the knowledge base) irrespective of the query. This speeds up the subsequent query answering step and particularly pays off if many different queries are to be posed against a fixed knowledge base. It also

allows for establishing our query complexity result. Similar pre-compilation (aka materialization) techniques have been proposed and proven useful for query answering in lightweight description logics (Lutz, Toman, and Wolter 2009; Kontchakov et al. 2011). These logics however, feature the tree model property which makes blocking and query answering much more straightforward than in our case.

Third, by a more economic definition of the notion of patterns, the algorithm does now run in 2EXPTIME worst case complexity and allows for earlier blocking. Note also that patterns (which could be referred to as “bag-types”) are created on the fly as they occur in the constructed model representation (instead of being a-priori present like in type-elimination-based reasoning techniques), hence just like for tableaux procedures, a good performance for practical cases can be expected.

There is still much to do to enhance the practical interest of the algorithm by (1) improving the complexity of the \*-homomorphism search (2) tuning the algorithm for special subclasses of *gbts*, including lightweight description logics like  $\mathcal{EL}$ , which should lead to simpler patterns and \*-homomorphism tests. This theoretical work done, we will first implement and experiment restricted versions of our algorithm dedicated to specific *gbts* subclasses with polynomial data complexity.

**Acknowledgements** S. Rudolph was supported by an exchange stipend of the CNRS and the DFG project ExpreST.

## References

- Andréka, H.; Németi, I.; and van Benthem, J. 1998. Modal Languages and Bounded Fragments of Predicate Logic. *J. of Philosophical Logic* 27:217–274.
- Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition.
- Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the  $\mathcal{EL}$  envelope. In *IJCAI*, 364–369.
- Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2009. Extending Decidable Cases for Rules with Existential Variables. In *IJCAI*, 677–682.
- Baget, J.-F.; Mugnier, M.-L.; Rudolph, S.; and Thomazo, M. 2011. Walking the Complexity Lines for Generalized Guarded Existential Rules. In *IJCAI*, 712–717.
- Baget, J.-F.; Leclère, M.; and Mugnier, M.-L. 2010. Walking the Decidability Line for Rules with Existential Variables. In *KR*. AAAI Press.
- Beeri, C., and Vardi, M. 1984. A Proof Procedure for Data Dependencies. *Journal of the ACM* 31(4):718–741.
- Buchheit, M.; Donini, F. M.; and Schaerf, A. 1993. Decidable Reasoning in Terminological Knowledge Representation Systems. *JAIR* 1:109–138.
- Calì, A.; Gottlob, G.; and Kifer, M. 2008. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In *KR*, 70–80.
- Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2009. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. In *PODS*, 77–86. ACM.
- Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning* 39(3):385–429.
- Courcelle, B. 1990. The Monadic Second-Order Logic of Graphs: I. Recognizable Sets of Finite Graphs. *Inf. Comput.* 85(1):12–75.
- Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data Exchange: Semantics and Query Answering. *Theor. Comput. Sci.* 336(1):89–124.
- Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyashev, M. 2011. The Combined Approach to Ontology-Based Data Access. In *IJCAI*, 2656–2661.
- Krötzsch, M., and Rudolph, S. 2011. Extending Decidable Existential Rules by Joining Acyclicity and Guardedness. In *IJCAI*, 963–968.
- Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2007. Complexity Boundaries for Horn Description Logics. In *AAAI*, 452–457. AAAI Press.
- Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive Query Answering in the Description Logic  $\mathcal{EL}$  Using a Relational Database System. In *IJCAI*, 2070–2075.
- Motik, B.; Shearer, R.; and Horrocks, I. 2009. Hypertableau reasoning for description logics. *JAIR* 36:165–228.
- Ortiz, M.; Rudolph, S.; and Šimkus, M. 2011. Query Answering in the Horn Fragments of the Description Logics *SHOIQ* and *SRIOIQ*. In *IJCAI*, 1039–1044.
- Robertson, N., and Seymour, P. D. 1984. Graph Minors. III. Planar Tree-Width. *J. Comb. Theory, Ser. B* 36(1):49–64.
- Thomazo, M.; Mugnier, M.-L.; Baget, J.-F.; and Rudolph, S. 2012. A Generic Querying Algorithm for Greedy Sets of Existential Rules. Research Report, LIRMM.