A Generic Querying Algorithm for Greedy Sets of Existential Rules

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

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.

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 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 $p(t_1, \ldots, t_k)$ 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.[1] 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, $\mathit{vars}(A)$ and $\mathit{terms}(A)$ denote its set of variables and of terms, respectively. Given conjuncts F and Q, a *homomorphism* π from Q to F is a substitution of $\mathit{vars}(Q)$ by terms of F s.t. $\pi(Q) \subseteq F$ (we say that π 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* $B = \mathit{body}(R)$ *and* $H = \mathit{head}(R)$ *are conjuncts, called the* body *and the* head *of* R*, respectively. The* frontier *of* R*, denoted* $\mathit{fr}(R)$*, is the set of variables* $\mathit{vars}(B) \cap \mathit{vars}(H) = \mathbf{y}$.

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* π *from* $\mathit{body}(R)$ *to* F*; the result of the* application of R to F w.r.t. π *is a fact* $\alpha(F, R, \pi) = F \cup \pi^{\mathrm{safe}}(\mathit{head}(R))$ *where* π^{safe} *is a substitution of* $\mathit{head}(R)$*, that replaces each* $x \in \mathit{fr}(R)$ *with* $\pi(x)$*, and each other variable with a "fresh" variable, i.e., not introduced before. As* α *only depends on* $\pi|_{\mathit{fr}(R)}$ *(the restriction of* π *to* $\mathit{fr}(R)$*), we also write* $\alpha(F, R, \pi|_{\mathit{fr}(R)})$.

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* $F_2 = \alpha(F_1, R_1, h_2) = F_1 \cup \{r(d, z_2)\}$.

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* $(F_0 = F), (R_0, \pi_0, F_1), \ldots, (R_{k-1}, \pi_{k-1}, F_k)$ *s.t. for all* $0 \leq i < k$*,* $R_i \in \mathcal{R}$ *and* π_i *is a homomorphism from* $\mathit{body}(R_i)$ *to* F_i *s.t.* $F_{i+1} = \alpha(F_i, R_i, \pi_i)$*. When only the successive facts are needed, we note* $(F_0 = F), F_1, \ldots, F_k$.

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 $\mathit{vars}(F) \cup \mathcal{C}$, 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 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. Given a fact F_i, a derivation S to F_i or a tree decomposition $\mathcal{T}$ of F_i, we note $\mathit{atoms}(S) = \mathit{atoms}(\mathcal{T}) = F_i$.

[1] Note that hereby we generalize the classical notion of a fact in order to take existential variables into account.

[2] We assume that the reader is familiar with this notion, see e.g. (Robertson and Seymour 1984).

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*$(F') \leq b$.

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

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

1. *the root of $\mathcal{T}$ is B_0 with $\mathsf{terms}(B_0) = T_0$ and $\mathsf{atoms}(B_0) = \mathsf{atoms}(F)$.*
2. *For $0 < i \leq k$, let R_{i-1} be the rule applied according to homomorphism π_{i-1} to produce F_i; then $\mathsf{terms}(B_i) = \mathsf{vars}(A_{i-1}) \cup T_0$ and $\mathsf{atoms}(B_i) = \mathsf{atoms}(A_{i-1})$, where $A_{i-1} = \pi_{i-1}^{\text{safe}}(\mathsf{head}(R_{i-1}))$. There is an edge between B_i and the node B_j s.t. j is the smallest integer for which $\pi_{i-1}(fr(R_{i-1})) \subseteq \mathsf{terms}(B_j)$ (since the derivation is greedy, such a B_j always exists); this edge is labeled by $(R_{i-1}, \pi_{i-1}|_{fr(R_{i-1})})$.*

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

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 $\mathsf{atoms}(B_1) = \{r(b, z_1)\}$ and $\mathsf{terms}(B_1) = \{a, b, c, d, z_1\}$. (R_1, h_2) yields a bag B_2 with $\mathsf{atoms}(B_2) = \{r(d, z_2)\}$ and $\mathsf{terms}(B_2) = \{a, b, c, d, z_2\}$. $fr(R_1) = \{y\}$ and $h_2(y) = d$, which is both in $\mathsf{terms}(B_0)$ and $\mathsf{terms}(B_1)$, 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.*

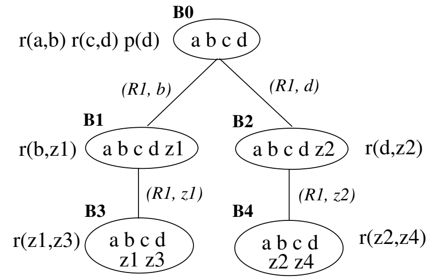

Figure 1: Derivation tree of Example 1. Only the image of the single frontier variable from R_1 is mentioned in edge labels.

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 leafs). 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 wether *gbts* is recognizable,[3] 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., $\mathit{vars}(a) = \mathit{vars}(\mathit{body}(R))$ (Calì, 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 (Calì, Gottlob, and Kifer 2008); second, by noticing that only frontier 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 $a \in \mathit{body}(R)$ that contains all *affected* variables from $\mathit{body}(R)$; 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

[3]We conjecture that it is.

Class	w unbounded	w bounded	Data Comp.
gbts	2ExpTime *	2ExpTime	ExpTime
wfg, jfg	2ExpTime	2ExpTime	ExpTime
fr1, fg	2ExpTime	2ExpTime	PTime
wg	2ExpTime	ExpTime	ExpTime
guarded	2ExpTime	ExpTime	PTime

*: 2ExpTime membership proven in this paper

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

of affected variables. A rule R is *frontier-guarded (fg)* if there is an $a \in body(R)$ with $vars(fr(R)) \subseteq vars(a)$. 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 $a \in body(R)$ that contains all affected variables from $fr(R)$. By refining the notion of an affected variable, *w(f)g* can be further generalized into jointly-(frontier)-guarded *(j-(f)g)* (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 are 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, π), where G is subset of a rule body and π is a partial mapping from $terms(G)$ to $terms(B)$. G and π 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 $(G, \pi) \in P(B)$, π 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 $sb_R \subseteq body(R)$ and any homomorphism π from sb_R to F_k, $P(B)$ contains (sb_R, π'), where π' is the restriction of π to the inverse image of $terms(B)$. $PDT(S)$ 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 $(body(R), -)$; then, the bag created by a rule application (R, π) 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 $(body(R), \pi')$, with the restrictions of π' and π 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, $e_1 = (sb^1_R, \pi_1) \in P(B_1)$ and $e_2 = (sb^2_R, \pi_2) \in P(B_2)$ where sb^1_R and sb^2_R are subsets of $body(R)$ for some rule R. Let $V = vars(sb^1_R) \cap vars(sb^2_R)$. 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 $sb_R = sb^1_R \cup sb^2_R$ and $\pi = \pi_1 \cup \pi'_2$, where π'_2 is the restriction of π_2 to the inverse image of $terms(B_1)$ (i.e., the domain of π'_2 is the set of terms with image in $terms(B_1)$).*

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

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 $e_1 = (sb^1_R, \pi_1) \in P_1$, $e_2 = (sb^2_R, \pi_2) \in P_2$, sb^1_R and sb^2_R are subsets of $body(R)$ 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, π) obtained from $(G, \pi_2) \in P_2$ by restricting π_2 to the inverse image of $terms(B_1)$.

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, π) s.t. G is a subset of a rule body and π is a* homomorphism *from G to $atoms(B_c)$.*

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 π be a homomorphism from $sb_R \subseteq \mathit{body}(R)$ to F_i, for some rule R. We show that $J(P(B_c), P(B_p))$ contains (sb_R, π'), where π' is the restriction of π to the inverse image of $\mathit{terms}(B_c)$. Let us partition sb_R into b_{i-1}, the subset of atoms mapped by π to F_{i-1}, and b_i the other atoms from sb_R, which are necessarily mapped by π to $F_i \setminus F_{i-1}$, i.e., $\mathit{atoms}(B_c)$. If b_i is not empty, by definition of the initial pattern, $P(B_c)$ contains (b_i, π_c), where π_c is the restriction of π to $\mathit{terms}(b_i)$. 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}, π_p), where π_p is the restriction of $\pi_{|b_{i-1}}$ to the inverse image of $\mathit{terms}(B_p)$. If b_{i-1} or b_i is empty, (sb_R, π') 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, π') (Point 3 in Def. 10). □

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 Prop. 4. The crucial point is that if π 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 $e \in \mathit{terms}(B_c)$, hence, thanks to the running intersection property of a decomposition tree, $e \in \mathit{terms}(B')$, thus $(e, \pi(e))$ will be propagated to $P(B)$. □

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, π_i) and (R, π_j) of the same rule R. B and B' are structurally equivalent if :*

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

Example 1 (contd.) *Consider the DT in Example 1, depicted in Fig. 1. Although both B_1 and B_2 result from R_1, they are not structurally equivalent because $fr(R_1) = y$, $h_1(y) \in T_0$, 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 $\mathrm{terms}(B_4)$) 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 term introduced in B to the respective term 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 $\mathrm{terms}(B)$ to $\mathrm{terms}(B')$ (in short from B to B'), denoted $\psi_{B \to B'}$, is defined as follows:*

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

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: $(G, \pi') \in P(B') \Rightarrow (G, \psi_{B' \to B} \circ \pi') \in P(B)$.*

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 $\psi_{B_1 \to B_2}$. Let B'_1 be a child of B_1. Copying B'_1 under B_2 (according to $\psi_{B_1 \to B_2}$) consists in adding a child B'_2 to B_2, s.t.* terms(B'_2) *is obtained by the following bijection b, and* atoms$(B'_2) = b($atoms$(B'_1))$*: for all $x \in$* terms(B'_1)*, if $x \in$* terms(B_1) *then $b(x) = \psi_{B_1 \to B_2}(x)$, 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, π) 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. $\psi_{B_1 \to B_2} \circ \pi$ (up to fresh variable renaming). Moreover, B'_1 and B'_2 are structurally equivalent and $\psi_{B'_1 \to B'_2} = b$.*

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 $(\mathcal{T}_b[root], identity)$, where $\mathcal{T}_b[root]$ 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 $\mathcal{T}_{new}$ be obtained from $\mathcal{T}$ by copying B'_c under B (according to $\psi_{B'_r \to B}$), which yields a new bag B_c. Then $(\mathcal{T}_{new}, f \cup (B'_c, B_c))$ 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 $atoms(\mathcal{T}'') = atoms(S)$ (up to fresh variable renaming) and $\mathcal{T}'$ is a prefix subtree of $\mathcal{T}''$.*
- *(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 bags but one 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.

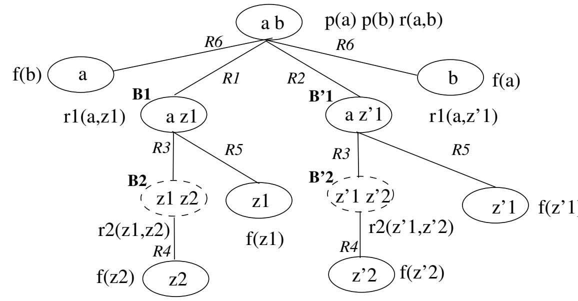

Figure 2: The yoyo example (Example 2) – Partial drawing: a, b omitted in some bags, π not drawn on edges.

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

R_1: $r(x, y) \to r_1(x, z)$ $\quad$ R_4: $r_2(x, y) \to f(y)$
R_2: $r(x, y) \to r_1(y, z)$ $\quad$ R_5: $r_2(x, y) \wedge f(y) \to f(x)$
R_3: $r_1(x, y) \to r_2(y, z)$ $\quad$ R_6: $r_1(x, y) \wedge f(y) \to f(x)$

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 Γ of *pseudo-rules* to propagate pattern updates. A pseudo-rule has the form $P \to P'$ ("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 Γ 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 Γ, 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 gbts rule set R
Result: A full blocked tree of (F, R)
Let 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
B = {([P(B_0)], [B_0])};
Γ = ∅;
while There is a new rule application (R, π) on an unblocked
node B_p do
    Apply (R, π), creating B_c (with P(B_c) its initial
    pattern);
    UPDATECHILD(T, B, Γ, B_c, B_p);
    UPDATEPARENT(T, B, Γ, B_p, B_c) // Γ may grow;
    while There is a pseudo-rule P(B) → P' applicable on
    a bag B do
        Perform this application
        UPDATEPARENT(T, B, Γ, B, Parent(B))
return T, 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.

Algorithm 2: UPDATECHILD

```
Data: T, B, Γ, B_c and B_p, two bags of T, where B_c is a
      child of B_p
Result: T and B updated
Let P = JOIN(P(B_c), P(B_p));
if P ≠ P(B_c) then
    Remove B_c from L_b(B_c);
    if P appears in 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 B; // B_c is unblocked
    for any child B'_c of B_c do
        UPDATECHILD(T, B, Γ, B'_c, B_c)
```

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: 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 is 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 a, 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. □

Algorithm 3: UPDATEPARENT

```
Data: T, B, Γ, B_p and B_c, two bags of T, where B_c is a child
      of B_p
Result: T and B updated
Let P = JOIN(P(B_p), P(B_c));
if P ≠ P(B_p) then
    for any P' ∈ L_p(B_p) do
        Γ = Γ ∪ {P' → P}
    if P appears in 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(T, B, Γ, Parent(B_p), B_p)
    for any child B'_c ≠ B_c of B_p do
        UPDATECHILD(T, B, Γ, B'_c, B_p)
```

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 Ψ) 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 $\mathcal{T}^*$. □

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, π) (with any π) 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 $\mathcal{T}^* =$ FullBlockedTree$(F, \mathcal{R})$.

From the soundness and completeness properties of 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 $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \ldots, (Q_p, \pi_p))$ 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_Π maps each node of $\mathcal{T}_\Pi$ to a bag of $\mathcal{T}^*$;*
- *if Q_i is empty, we define* terms(Q_i) *as the set of terms shared by at least two of its children;*
- *$\forall 1 \leq i \leq p$, π_i is a substitution from* terms(Q_i) *to* terms$(f_\Pi(Q_i))$. *If, moreover, Q_i is not empty, then π_i is a homomorphism from Q_i to* 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 Fig. 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 Fig. 4,*
- *$f_{\Pi^1}(1) = f_{\Pi^2}(1) = B_0, f_{\Pi^1}(2) = f_{\Pi^2}(2) = L_1, f_{\Pi^1}(3) = f_{\Pi^2}(3) = L_2, f_{\Pi^1}(4) = f_{\Pi^2}(4) = L_3, f_{\Pi^1}(6) = f_{\Pi^2}(6) = D_1$,*
- *$f_{\Pi^1}(5) = D_3, f_{\Pi^2}(5) = D_1$,*

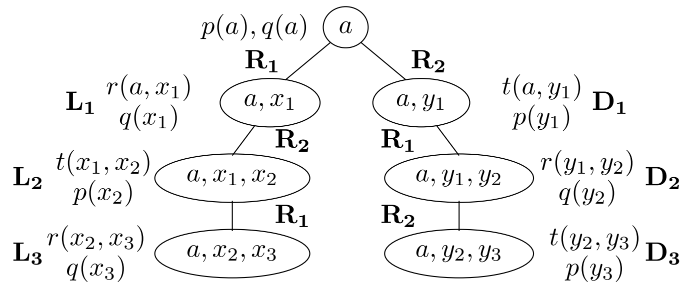

Figure 3: The full blocked tree generated by Algorithm 1 (Example 3); L_2 and D_3 are equivalent, as well as L_3 and D_2

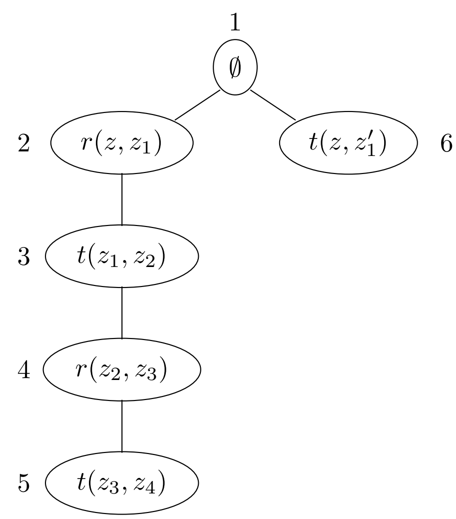

Figure 4: $\mathcal{T}_{\Pi^1} = \mathcal{T}_{\Pi^2}$ (Example 3)

- *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 $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \ldots, (Q_p, \pi_p))$ be a pre-$*$-homomorphism from Q to $\mathcal{T}^*$. We say that Π is a* $*$-homomorphism *if there exists $(\mathcal{T}, f) \in G(\mathcal{T}^*)$ and an injective mapping ω (called a $*$-proof) 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 $\mathcal{T}_\Pi$, $f_\Pi(Q_i) = f(\omega(Q_i))$;*
- *$\forall t \in$ 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 Π^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 $\omega(1) = B_0, \omega(2) = L_1, \omega(3) = L_2, \omega(4) = L_3, \omega(5) = L_4, \omega(6) = D_1$.*

Π^2 is not a $*$-homomorphism: since D_1 cannot be copied under any node, and that 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 $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \ldots, (Q_q, \pi_q))$ 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 Π can be put into this smaller form Π', we only need to check reduced pre-$*$-homomorphism). 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 $a = |atoms(\mathcal{T}^*)|$.

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 $\mathcal{T}^*$.*

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 Π). To show for some Π that it is a $*$-homomorphism, we explore (in a breadth-first manner) the tree $\mathcal{T}_\Pi$. At each step of the exploration, we have explored a prefix tree $\mathcal{T}'_\Pi$ of $\mathcal{T}_\Pi$. Let $Q_{i_1}, \ldots, Q_{i_p}$ be the Q_i appearing in $\mathcal{T}'_\Pi$, and $Q' = Q_{i_1} \cup \ldots \cup Q_{i_p}$. Then $\Pi' = ((\mathcal{T}'_\Pi, f_{\Pi|\mathcal{T}'_\Pi}), (Q_{i_1}, \pi_{i_1}), \ldots, (Q_{i_p}, \pi_{i_p}))$ 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 ω' to $(\mathcal{T}', f') \in G(\mathcal{T}^*)$ that respects the criteria of Def. 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 Π'' 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 $\omega'(Q_j)$, ω'' extends ω' 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 ω'' to the bags of $(\mathcal{T}'', f'')$ to be a $*$-proof of Π'': only the last item from Def. 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 ω to $(\mathcal{T}, f)$ that Π is a $*$-homomorphism, and that we have exhibited a $*$-proof ω' to $(\mathcal{T}', f')$ that some prefix Π' of Π is a $*$-homomorphism (where $\mathcal{T}'$ is not necessarily a prefix of $\mathcal{T}$). Then the $*$-proof ω' can be expanded to a $*$-proof ω'' to $(\mathcal{T}'', f'')$ that Π 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', σ) where B' is a bag of $\mathcal{T}^*$ and σ is a partial mapping from* terms(B') *to* terms(B) *s.t. there is a $(\mathcal{T}, f) \in G(\mathcal{T}^*)$ 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* terms(B'), *$\psi_{B' \to B_2}(t) = \psi_{B \to B_1}(\sigma(t))$. We say that B_2 is in state (B', σ) 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 of B in time $t \times (t \times f^f)^2$ (by maintaining a boolean matrix M s.t. $M_{i,j}$ = 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 effectively be used to compute $*$-proofs.

Property 11 *A pre-$*$-homomorphism $\Pi = ((\mathcal{T}_\Pi, f_\Pi), (Q_1, \pi_1), \ldots, (Q_p, \pi_p))$ from Q to $\mathcal{T}^*$ is a $*$-homomorphism iff there exists mappings $\tau_1, \ldots, \tau_p$ where τ_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 τ_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 $t \in$ terms(Q_i), if $\pi_i(t)$ belongs to the domain of σ, 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 $\tau_i(t) = \tau_j(t)$.*

The overall algorithm deciding whether $F, \mathcal{R} \models Q$ can now be sketched as follows: We first build the full blocked tree 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 Π 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 Prop. 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 a double exponential in F and $\mathcal{R}$, the algorithm runs in 2EXPTIME. Last, given a (nondeterministically 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 $|terms(bag)|^w$ 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 *fr1* 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 them in account the latter. Since all terms to which the frontier is mapped are argument 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 (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 2EXPTIMEworst 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 implement our algorithm on restricted subclasses of g.b.t.s., such as rules translating $\mathcal{EL}$ ontologies or generalizations of them.

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 Kaelbling, L., and Saffiotti, A., eds., *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, 364–369. Professional Book Center.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2009. 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,*, 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 *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Canada*. 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 *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia*, 70–80.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2009. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the 28th symposium on Principles of database systems (PODS'09)*, 77–86. New York, NY, USA: 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 Zakharyaschev, 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 *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, 452–457. AAAI Press.

Lutz, C.; Toman, D.; and Wolter, F. 2009. 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,*, 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 sroiq. 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.

Appendix

Derivation Scheme

Definition 22 (Fusion of the frontier) *Let R be a rule. A* fusion *of* $\mathsf{fr}(R)$ *is a substitution of* $\mathsf{fr}(R)$ *by* $\mathsf{fr}(R) \cup T_0$.

Definition 23 (Fusion induced by π) *Let $R = (B, H)$ be a rule, V be a set of variables. Let π be a substitution of* $\mathsf{terms}(B)$ *by $T_0 \cup V$. A fusion of* $\mathsf{fr}(R)$ *induced by π, denoted by σ_π is such that*

- $\sigma_\pi(x) = \sigma_\pi(y) \Leftrightarrow \pi(x) = \pi(y)$
- $\sigma_\pi(x) = a \in T_0 \Leftrightarrow \pi(x) = a$

Definition 24 (Abstract bag) *Let $R = (B, H)$ be a rule, σ a fusion of* $\mathsf{fr}(R)$. *The abstract bag assigned to (R, σ) (notation:* $\mathsf{ab}(R, \sigma)$*) is the set of terms* $\sigma(\mathsf{terms}(H)) \cup T_0$ *labeled by the atoms $\sigma(H)$.*

We naturally extend the notion of structural equivalence between bags to structural equivalence between bags and abstract bags. We then define the usual ψ's.

Definition 25 (Path) *A path δ is a sequence of pairs $((R_i, \pi_i))_{i \leq n}$ s.t. π_i is a mapping from* $\mathsf{fr}(R_i)$ *to* $\mathsf{terms}(\mathsf{ab}(R_{i-1}, \sigma_{\pi_{i-1}}))$. *The end of the path (denoted by $\iota(\delta)$) is* $\mathsf{ab}(R_n, \sigma_{\pi_n})$.

Definition 26 (Path in a derivation tree) *Let S be a derivation, $B = B_0, \ldots, B' = B_n$ a sequence of bags of $DT(S)$ s.t. B_i is a child of B_{i-1} for every $i \geq 1$. The path from B to B' is the sequence of pairs* $(R_i, \psi_{B_{i-1} \to \mathsf{ab}(R_{i-1}, \sigma_{\pi_{i-2}})} \circ \pi_i)$ *where (R_i, π_i) is the label of the edge (B_{i-1}, B_i) in $DT(S)$. If $B = B'$, the path is empty.*

Definition 27 (Derivation scheme) *A* derivation scheme *is a pair (C, Λ) such that:*

- *C is an abstract bag assign to (R, σ) for some rule R and some fusion σ of $fr(R)$*
- *$\Lambda = ((R_1, \delta_1, \pi_1), \ldots, (R_n, \delta_n, \pi_n))$ such that:*
  - *$R_i = (H_i, C_i)$ is a rule,*
  - *δ_i is a path,*
  - *π_i is a mapping from* $\mathsf{fr}(R_i)$ *to* $\mathsf{terms}(\iota(\delta_i))$.

Definition 28 *Let S be a derivation, B a bag of $DT(S)$, Δ a derivation scheme. $\Delta = (C, \Lambda)$ is applicable to B in S if B is structurally equivalent to C and:*

- *either Λ is empty. The application of Δ to B in S leaves S unchanged.*
- *or $\Lambda = \Lambda'.((R_n, \delta_n, \pi_n))$ and the following conditions hold:*
  - *Δ' is applicable to B in S,*
  - *there is B' s.t. the path from B to B' is δ_n (and if it exists, it is unique)*
  - *$\psi_{\sigma_{\pi_{n-1}}(H_{n-1})} \circ \pi_n$ can be extended to a homomorphism from H_n to $\Delta'(S, B)$.*

In the later case, applying Δ results in $\Delta'(S, B)$ if (R_n, π_n) has already been applied to B', and $\alpha(\Delta'(S, B), R_n, \pi_n)$ otherwise.

The next property shows that the definition of bag equivalence captures the idea that the same derivations can be performed on and below these bags.

Definition 29 *Let S be a derivation, B a bag of $DT(S)$. The derivation scheme $\Delta_S(B)$ applied to B in S is defined as follows:*

- *the root of $\Delta_S(B)$ is the abstract bag structurally equivalent to B*
- *start with Λ empty. For each rule application (R, π) (taken in the order of the derivation) that adds a child to a descendant B' of B (B' may be equal to B), let δ be the path from B to B', we add at the end of Λ the triple $(R, \delta, \psi_{B' \to \iota(\delta)} \circ \pi)$.*

Property 13 *Let S be a derivation, B and B' two bags of $DT(S)$. If B and B' are equivalent then, for any derivation scheme Δ applicable to B, the derivation scheme $\Delta_S(B).\Delta$ is applicable to B'.*

Proof:(sketch) We first show that $\Delta_S(B)$ is applicable to B', by induction on the length of $\Delta_S(B)$. We then prove by induction on the length of Δ that $\Delta_S(B).\Delta$ is applicable to B'. □

Let T^* be the blocked tree produced by the algorithm.

Property 14 *Let S be a derivation to F_k s.t. $DT(S)$ is isomorphic to a T with $(T, f) \in G(T^*)$. For every bag B of T, $P(B)$ is included in $P(f(B))$ in T^*.*

Proof:(sketch) By induction on the length of S, using the monotonicity of the join w.r.t. pattern inclusion. □

Property 15 *If $P \to P'$ is a rule created by the algorithm, then there is a derivation scheme $\Delta_{P \to P'}$ applicable to any bag B with $P \sqsubseteq P(B)$ s.t. $P' \sqsubseteq P(B)$ after application of $\Delta_{P \to P'}$.*

Proof:(sketch) By induction on the *rank* of $P \to P'$ (i.e., order in which pseudo-rules are created by the algorithm).

- if $P \to P'$ has rank 1, we apply to a bag B with $P \sqsubseteq P(B)$ the derivation scheme that has been applied to B_P (the representative of P). This derivation scheme is well defined since no blocking has been performed yet.
- if $P \to P'$ is a rule of rank n, let B_P the representative of P when $P \to P'$ is created. We define $\Delta_{P \to P'}$ by appending the existential rule applications and the derivation schemes associated with pseudo-rules that have been applied below B. $\Delta_{P \to P'}$ is applicable to B, and after application of $\Delta_{P \to P'}$ we have $P' \sqsubseteq P(B)$.

□

Proof of Theorem 8

Completeness: let S be a derivation from F. By induction on the length of S, we prove that $DT(S) \in G(T^*)$. If S is of length 0, this is trivial. Otherwise, let S' be a derivation of length $n-1$, R a rule applied to F_{n-1} by π, and B the bag of $DT(S')$ to which we add a child B' by applying (R, π). Since $(DT(S'), f)$ is generated by T^*, $f(B)$ is a bag of T^* whose pattern includes $P(B)$. Since (R, π) is applicable to B, $(\mathit{body}(R), \pi) \in P(B)$. Thus, $(\mathit{body}(R), \psi_{B \to f(B)}) \in P(f(B))$. Since no more rule can be applied to T^*, there is a child of $f(B)$ in T^* which has been created by the application $(R, \psi_{B \to f(B)})$. Then, B' can be obtained by copying this child under B, which shows that $DT(S)$ can be generated by T^*.

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

- if T is a single bag B (then f is defined by $f(B) = \mathit{root}(T^*)$), there is a trivial isomorphism Ψ from B to a bag containing F. Moreover, since the root of T^* never appears as second argument of UPDATECHILD, there is a pseudo-rule $P \to P^*$, where P is the initial pattern of F, and P^* is the pattern of $\mathit{root}(T^*)$. By Property 15, we can apply $\Delta_{p \to p^*}$ to the only bag of the derivation. After this application,$P^* \sqsubseteq P(\Psi(B))$.
- let $(T', f') \in G(T^*)$ fulfilling the induction assumption. Let B' be a bag of T' and $B = f'(B')$; let B_r be the representative of the equivalence class of B. We copy a child B_c of B_r under B', yielding B'_c. Since B_c is a child of B_r, there is a rule R and a mapping π from $fr(R)$ s.t. $(\mathit{Body(R)}, \pi) \in P(B_r)$. Since B_r is the representative of B' and $P(\Psi(B))$ includes $P(B')$ (induction assumption), $(\mathit{body}(R), \psi_{B_r \to \Psi(B)} \circ \pi)$ belongs to $P(\Psi(B))$. Thus, R is applicable to $\Psi(B)$ by $\psi_{B_r \to \Psi(B)} \circ \pi$, yielding a bag B''_c. If this rule application is new, we denote by S' the extension of S with this rule application, otherwise $S' = S$. $P(B''_c)$ in $DT(S')$ is equal to the join of $P(\Psi(B))$ and the initial pattern of B''_c. In particular, $P(B''_c)$ includes the initial $P(B_c)$. Thus, by Property 15, we can build from the sequence of pseudo-rules $P_1 \to P_2, \ldots, P_{n-1} \to P_n$, with $P_n = P(B_c)$, a sequence of derivation schemes $\Delta_{P_1 \to P_2}, \ldots, \Delta_{P_{n-1}, P_n}$ which is applicable to B''_c, and s.t. $P(B_c) \sqsubseteq P(B''_c)$ after these derivation scheme applications.