Regular Matching and Inclusion on Compressed Tree Patterns with Constrained Context Variables

Iovka Boneva, Joachim Niehren, Momar Sakho

Regular Matching and Inclusion on Compressed Tree Patterns with Constrained Context Variables

Iovka Boneva[a], Joachim Niehren[b], Momar Sakho[a,b,*]

[a] *Université de Lille, Cité Scientifique, 59650 Villeneuve d'Ascq, France*
[b] *Inria Lille - Nord Europe, Parc scientifique de la Haute-Borne, 40 avenue Halley, 59650 Villeneuve d'Ascq*

Abstract

We study the complexity of regular matching and inclusion for compressed tree patterns with context variables subject to regular constraints. Context variables with regular constraints permit to properly generalize on unranked tree patterns with hedge variables. Regular inclusion on unranked tree patterns is relevant to certain query answering on XML streams with references. We show that regular matching and inclusion with regular constraints can be reduced in polynomial time to the corresponding problem without regular constraints.

Keywords: tree patterns, tree automata, computational complexity, streams, XML, grammar compression.

1. Introduction

A pattern is a term with variables describing a string, a tree, or some other algebraic value. The following generic problems for patterns were widely studied in the literature:

Pattern matching: Is a given algebraic value an instance of a given pattern?

Pattern unification: Do two given patterns have some common instance?

*This paper extends the LATA'2019 paper [1] with regular constraints.

*Corresponding author
Email addresses: iovka.boneva@univ-lille.fr (Iovka Boneva), joachim.niehren@inria.fr (Joachim Niehren), momar.sakho@inria.fr (Momar Sakho)

	DFAs	NFAs
MATCH	PSPACE-c [5]	PSPACE-c [5]
INCL	PSPACE-c [5]	PSPACE-c [5]

Figure 1: (Compressed) string patterns.

	DFAs	NFAs
MATCH	P [5]	P [5]
INCL	P [5]	PSPACE-c [5]

Figure 2: Linear restriction.

	DTAs	NTAs
MATCH	NP-c [6],Th 7,Prop 26	EXP-c [6], Th 7, Prop 26
INCL	coNP-c Th 7, Prop 26	EXP-c Lem 23, Th 7, Prop 26

Figure 3: (Compressed) tree patterns without context variables.

	DTAs	NTAs
MATCH	P Prop 35	P Prop 35
INCL	P Lem 22, Prop 35	EXP-c Lem 23, Th 27

Figure 4: Linear restriction.

	DTAs	NTAs
MATCH	PSPACE-c Th 27, Prop 28	EXP-c Th 27, Prop 28
INCL	PSPACE-c Th 27, Prop 28	EXP-c Th 27, Prop 28

Figure 5: (Compressed) tree patterns with (constrained) context variables.

	DTAs	NTAs
MATCH	P Prop 35	P Prop 35
INCL	P Lem 22, Prop 35	EXP-c Lem 23 , Th 27

Figure 6: Linear restriction.

Regular pattern matching: Does some instance of the given pattern belong to the given regular language?

Regular pattern inclusion: Do all instances of the given pattern belong to the given regular language?

As inputs, these problems receive descriptors of patterns, values, and regular languages. Most typically, a string pattern may be described in a compressed manner by using a singleton context-free grammar (also called straight-line program), and a regular string language may be represented by a nondeterministic finite automaton (NFA) or by a deterministic finite automaton (DFA). The problem of string pattern matching is well known to be NP-complete for NFAs [2] but in P for DFAs, with and without compression [3]. The more general problem of string unification is of quite different nature. It is known to be in PSPACE [4].

Regular inclusion and matching for string patterns was studied by the au-

thors on the way to the present paper [5]. Both problems were shown to be PSPACE-complete, both for DFAs and NFAs, with and without compression. See Fig. 1 for an overview. When restricted to linear string patterns, the complexity goes down to polynomial time in 3 of the 4 cases, as summarized in Fig. 2. The problem which remains PSPACE-complete is regular inclusion on linear string patterns for NFAs.

The complexity landscape of regular matching and inclusion for tree patterns without context variables looks quite different to the case of string patterns, see Figs. 3 and 4. Here, regular languages are defined by tree automata, which may either be nondeterministic (NTAs) or (bottom-up) deterministic (DTAs). Regular matching of tree patterns without context variables for NTAs was named the *ground instance intersection problem* in [6], where it was shown to be NP-complete for DTAs and EXP-complete for NTAs. Furthermore it was shown that the restriction to linear patterns is in P, both for DTAs and NTAs. Regular inclusion for tree patterns has not been studied so far to the best of our knowledge. We show that it is coNP-complete for DTAs and EXP-complete for NTAs even when restricted to linear tree patterns. Only for DTAs, the problem of regular inclusion for linear tree patterns is in P. Compression can be added to tree patterns by using singleton tree grammars [7]. But as we will see, this doesn't affect the above results. Indeed, all the missing results cited above will be proven in the present paper with and without compression, as a byproduct of our main results.

The prime reason for the asymmetry of the complexity landscapes in the case of strings and trees is that string patterns cannot be encoded as tree patterns with a monadic signature without adding context variables. For instance, the string pattern $aZZbY$ corresponds to the tree pattern $a(Z@(Z@b(Y)))$ with context variable Z, tree variable Y and application symbol @. The interest of adding context variables to tree patterns was already noticed when generalizing string pattern matching to context pattern matching [3], which are both NP-complete, with or without compression. The same was noticed when generalizing string unification to context unification, that are both in PSPACE [8]. Since we

are interested in a proper generalization of regular matching and inclusion from string to tree patterns, we propose to study these problems for tree patterns with context variables.

In this paper, we relate regular matching of tree patterns to inhabitation problems of tree automata in a systematic manner. The naive semi-decision procedure for regular matching guesses some context for all the context variables in the tree pattern and then checks whether the instance of the pattern obtained thereby matches the regular language, i.e. whether it is recognized by the tree automaton defining this language. In order to avoid infinite guesses, our decision algorithm will guess for all context variables a function of type $Q \to 2^Q$ where Q is the set of states of the tree automaton, and then test whether this function is inhabited by some context with respect to the automaton. In order to make this approach work, we need to study the problem of context inhabitation on its own right.

Our first contribution is therefore the formal definition of the problem of *context inhabitation for tree automata* and the study of its complexity. Context inhabitation is a special case of second-order linear λ-definability, except that the input function is represented in a succinct manner. More generally, λ-*definability* is known to be decidable up to the order of three [9], while it is undecidable in general [10, 11]. Context inhabitation for tree automata can also be understood as a generalization of transition inhabitation for word automata, which is sometimes called the membership problem of the transition monoid [12]. We show that context inhabitation for NTAs is EXP-complete. The lower bound is obtained by a reduction from the nonemptiness problem of intersections of a finite number of NTAs [13], and the upper bound by a novel algorithm using determinization. We then show that context inhabitation for DTAs is PSPACE-complete. This result may be surprising[1]. We obtain the PSPACE upper bound

[1] We wrongly stated in the LATA'2019 version of the present paper that the problems of context inhabitation, regular matching and regular inclusion are EXP-complete for DTAs. We correct this error here, showing that all these problems are PSPACE-complete.

by a nontrivial reduction to the nonemptiness problem of intersections of a finite number of DFAs (for words) [12]. The fact that automata on words are enough for this purpose rather than automata for trees explains the otherwise surprising PSPACE upper bound.

The second contribution of the present paper is the study of the complexity of regular matching and inclusion for compressed tree patterns with context variables. All our results are based in a systematic manner on the close relationship between regular matching of tree patterns with context variables and context inhabitation for tree automata. They are summarized in Figs. 5 and 6. The only change compared to compressed string patterns is for NTAs, where the complexity increases from PSPACE-complete to EXP-complete. The main reason for this change is that the context inhabitation for NTAs is EXP-complete. In contrast, for DTAs context inhabitation remains PSPACE-complete, so that there is no difference to the case of DFAs.

The third contribution is the extension of regular matching and inclusion with regular constraints on the possible instantiations of the variables of the pattern.

Regular pattern matching with regular constraints: Does some instantiation of the variables satisfying the given regular constraints produce an instance of the given pattern that belongs to the given regular language?

Regular pattern inclusion with regular constraints: Do all instantiations of the variables satisfying the given regular constraints produce some instance of the given pattern that belong to the given regular language?

We show that the extended problems with regular constraints can be compiled to the original problems without constraints in polynomial time. Our reduction preserves the determinism of the automata and the linearity of the patterns. Therefore all our complexity results on regular matching and inclusion listed above remain valid when adding regular constraints.

The fourth contribution is an application of these results to regular matching and inclusion for compressed patterns on unranked trees with tree and hedge

variables (but without context variables). We show that the complexity results carry over. The idea is that unranked tree patterns can be encoded to ranked tree patterns, while mapping hedge variables to context variables. We contribute a reduction of unranked regular matching and inclusion to the ranked case but with regular constraints. In order to deal with the unranked symbols, we cannot bound the maximal arity of the ranked signature. Therefore we have to consider the uniform variant of all problems, where the signature is part of the input rather than being fixed as a parameter. This complicates the algorithms for the upper bounds, in the case of context inhabitation in particular.

Our original motivation for the present paper is the problem of certain query answering on hyperstreams [5], i.e., a special kind of stream with references [14]. A hyperstream for unranked trees is nothing else than a compressed pattern with hedge variables for unranked trees. In the case of Boolean queries defined by tree automata, the problem of certain query answering on hyperstreams is equal to the problem of regular inclusion of compressed pattern for unranked trees. The fourth contribution determines the exact complexity of this problem: it is the same as for regular inclusion of tree patterns with context variables on ranked trees. In particular, we provide an algorithm to decide certain query answering on hyperstreams with optimal complexity, through a series of reductions which ends with context inhabitation for tree automata.

Compared with the conference version at LATA'2019, we added the third contribution (the addition of regular constraints) and used it to prove the fourth contribution (the case of unranked trees), which was stated without proof before. We also added uniform variants for all the problems, where the signature is part of the input rather than being fixed as a parameter, since these are needed too, to prove our results on unranked tree patterns. The uniformity, however, makes the PSPACE upper bound of context inhabitation for DTAs considerably more difficult to establish. Finally, we added complexity results on the restriction of regular matching and inclusion to linear patterns in the journal version.

Outline. We recall the syntax and semantics of trees and contexts in Section 2. We then study tree and context inhabitation for tree automata in Section 3 for

complexity. Tree patterns with context variables are recalled in Section 4 and enhanced with compression in Section 5. Our main results on regular matching and inclusion are given in Section 6. The extension with regular constraints and how to get rid of them again is discussed in Section 7. Patterns of unranked trees with hedge variables are then investigated in Section 8. Before concluding, we consider the restricted case of linear tree patterns in Section 9.

2. Trees and Contexts

We recall the notion of trees and contexts and then recall how they can be interpreted in Σ-algebras. Throughout the paper we consider the two types in $\mathbf{T} = \{\mathit{tree}, \mathit{context}\}$.

A finite ranked signature is a tuple $\Sigma = \biguplus_{n \geq 0} \Sigma^{(n)}$ of function symbols $f \in \Sigma^{(n)}$ of arity n. **We assume that any ranked signature contains at least one constant and one symbol of arity at least 2.** This will be needed for the lower bounds.

The only values we will consider are the trees and contexts constructed over the symbols of a ranked signature. We take an approach based on the λ-terms – as needed on the way to the final application to regular inclusion of unranked tree patterns – but will not consider values of higher types.

Definition 1. *The set of trees $\mathcal{T}_\Sigma$ is the least set that contains all tuples $f(t_1, \ldots, t_n)$ where $f \in \Sigma^{(n)}$ for some $n \geq 0$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$.*

Atomic trees $a() \in \mathcal{T}_\Sigma$ are deliberately identified with $a \in \Sigma^{(0)}$.

For defining contexts, we fix an arbitrary nonempty set $\mathcal{V}^{\mathit{tree}}$ disjoint from Σ, whose elements are variables of type *tree*.

Definition 2. *The set of contexts $\mathcal{C}_\Sigma$ is the set of all terms $\lambda x.t$ for some tree $t \in \mathcal{T}_{\Sigma \cup \{x\}}$ where $x \in \mathcal{V}^{\mathit{tree}}$ occurs exactly once in t, and this with arity 0.*

We will identify contexts modulo α-renaming so that the choice of variable x does not matter. This means that the contexts $\lambda x.t$ and $\lambda x'.t[x/x']$ are equal for all $x, x' \in \mathcal{V}^{\mathit{tree}}$. The variable serves as the hole marker of the context.

A tree in $\mathcal{T}_\Sigma$ is a value of the first-order type *tree*, and a context in $\mathcal{C}_\Sigma$ a value of the linear second-order type $\mathit{context} = \mathit{tree} \multimap \mathit{tree}$. This is the subtype of the more usual function type $\mathit{tree} \to \mathit{tree}$ that is restricted to linear functions using their argument exactly once. Any context $\lambda x.t \in \mathcal{C}_\Sigma$ denotes a linear function since x must occur exactly once in t by definition. The set of all values of both types is:

$$\mathit{Val}_\Sigma = \mathcal{T}_\Sigma \cup \mathcal{C}_\Sigma$$

Adding nonlinear or higher-order λ-terms as values would lead to a quite different pattern matching problem.

We next present the interpretation of the values of both types in arbitrary Σ-algebras (including tree automata, as we will see later on). Trees will be interpreted as elements of the domain of the Σ-algebra, and contexts as linear functions on this domain.

Definition 3. *A Σ-algebra $\Delta = (\Sigma, D, .^\Delta)$ consists of a ranked signature Σ, a set D called the domain, and a mapping $.^\Delta$ that interprets symbols $f \in \Sigma^{(n)}$ as functions $f^\Delta : D^n \to D$. The domain of Δ is $\mathit{dom}^\Delta = D$.*

We next define the interpretation of values in a Σ-algebra. The *interpretation of a tree* $t = f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ is the domain element $[\![t]\!]^\Delta = f^\Delta([\![t_1]\!]^\Delta, \ldots, [\![t_n]\!]^\Delta)$. This interpretation can be extended to trees over the signature $\Sigma \cup D$ by interpreting any symbol $d \in D$ by itself, i.e., $d^\Delta = d$. The *interpretation of a context* $C = \lambda x.t \in \mathcal{C}_\Sigma$ is the function $[\![C]\!]^\Delta : D \to D$ with $[\![C]\!]^\Delta(d) = [\![t[x/d]]\!]^\Delta$ for all $d \in D$. The elements of D and functions of type $D \to D$ that can be obtained by Δ-interpretation of some tree or context are called Δ-*inhabited*:

$$[\![\,\mathit{Val}_\Sigma]\!]^\Delta = [\![\mathcal{T}_\Sigma]\!]^\Delta \cup [\![\mathcal{C}_\Sigma]\!]^\Delta$$

The set $\mathcal{T}_\Sigma$ of trees can be identified with the free Σ-algebra $(\Sigma, \mathcal{T}_\Sigma, .^{\mathcal{T}_\Sigma})$ whose interpretation function satisfies $f^{\mathcal{T}_\Sigma}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ for all symbols $f \in \Sigma^{(n)}$ and trees $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$. We note that $[\![\mathcal{T}_\Sigma]\!]^{\mathcal{T}_\Sigma} = \mathcal{T}_\Sigma$ while $[\![\mathcal{C}_\Sigma]\!]^{\mathcal{T}_\Sigma}$ is a proper subset of functions of type $\mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$. In other words, the interpretation over the Σ-algebra $\mathcal{T}_\Sigma$ converts any context $C \in \mathcal{C}_\Sigma$ into the function on

trees $[\![C]\!]^{\mathcal{T}_\Sigma} : \mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$ that it defines, i.e., if $C = \lambda x.t$ for some tree t in which x occurs once, then $[\![C]\!]^{\mathcal{T}_\Sigma}(t') = t[x/t']$ for all $t' \in \mathcal{T}_\Sigma$.

3. Inhabitation for Tree Automata

One of the insights of this paper will be that inhabitation is closely related to regular matching and inclusion for tree patterns, depending on the class of tree automata and the type of variables. Therefore, here we study inhabitation problems of tree automata on their own right.

3.1. Tree Automata

We start by recalling the standard notion of tree automata for ranked trees, their notion of bottom-up determinism, and their relationship to Σ-algebras also in the nondeterministic case.

Definition 4. *A (nondeterministic) tree automaton (*NTA*) over a ranked signature Σ is a tuple $A = (Q, \Sigma, F, \Delta)$ where Q is a finite set of states, $F \subseteq Q$ is the set of final states, and $\Delta \subseteq \cup_{n\geq 0}\Sigma^{(n)} \times Q^{n+1}$ is the transition relation.*

A rule $(f, q_1, \ldots, q_n, q) \in \Delta$ is written as $f(q_1, \ldots, q_n) \to q$. We will identify any transition relation Δ of some NTA as a Σ-algebra $(\Sigma, 2^Q, .^\Delta)$, that interprets function symbols $f \in \Sigma^{(n)}$ as the n-ary functions f^Δ that satisfy for any subsets of states $Q_1 \ldots, Q_n \subseteq Q$:

$$f^\Delta(Q_1, \ldots, Q_n) = \{q \mid \exists q_1 \in Q_1 \ldots \exists q_n \in Q_n.\ f(q_1, \ldots, q_n) \to q \text{ in } \Delta\}.$$

It should always be clear from the context whether we consider Δ as a Σ-algebra or as a transition relation.

The *regular language* $L(A)$ recognized by A is defined as the set of all trees in $\mathcal{T}_\Sigma$ whose evaluation in the Σ-algebra Δ yields some final state in F:

$$L(A) = \{t \in \mathcal{T}_\Sigma \mid [\![t]\!]^\Delta \cap F \neq \emptyset\}.$$

The more general concept of inhabitation from Σ-algebras can now be applied to tree automata, yielding the following definition:

Definition 5 (Inhabitation)**.** *Let $A = (Q, \Sigma, F, \Delta)$ be a tree automaton.*

- *A subset $Q' \subseteq Q$ is called Δ-inhabited by a tree $t \in \mathcal{T}_\Sigma$ if $Q' = [\![t]\!]^\Delta$.*
- *A function $S\colon 2^Q \to 2^Q$ is called Δ-inhabited by a context $C \in \mathcal{C}_\Sigma$ if $S = [\![C]\!]^\Delta$.*

An NTA is called *(bottom-up) deterministic* or equivalently a DTA if no two distinct rules of Δ have the same left-hand side, i.e., if Δ is a partial function from $\bigcup_{n\geq 0} \Sigma^{(n)} \times Q^n$ to Q. The *determinization* of an NTA A is the tree automaton $\det(A) = (2^Q, \Sigma, \det(F), \det(\Delta))$ where $\det(F) = \{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}$ and $\det(\Delta) = \{f(Q_1, \ldots, Q_n) \to f^\Delta(Q_1, \ldots, Q_n) \mid f \in \Sigma^{(n)}, Q_1, \ldots, Q_n \subseteq Q\}$. It is well known that $\det(A)$ is a DTA with $L(A) = L(\det(A))$. Furthermore, for any tree $t \in \mathcal{T}_\Sigma$ it holds that $[\![t]\!]^{\det(\Delta)} = \{[\![t]\!]^\Delta\}$.

An NTA is called *complete* if for all $f \in \Sigma^{(n)}$ and $q_1, \ldots, q_n \in Q$, there exists a state q so that the rule $f(q_1, \ldots, q_n) \to q$ is in Δ.

Let NTA_Σ be the set of all NTAs with signature Σ, and DTA_Σ the set of all DTAs with signature Σ. Clearly, $\text{DTA}_\Sigma \subseteq \text{NTA}_\Sigma$. A class of automata is a function that maps any signature Σ to a subset of NTA_Σ. In particular, NTA and DTA are classes of automata mapping signature Σ to the sets of automata NTA_Σ and DTA_Σ.

In the next subsections, we introduce and study the decision problem of context inhabitation, and its relationship to the problem of intersection nonemptiness. We distinguish the cases of NTAs and DTAs. In both cases, we consider the non-uniform version where the signature Σ is fixed as a parameter of the problem, and the uniform version where the signature is given with the input.

An overview of the results on context inhabitation is given in Fig. 8. Context inhabitation for nondeterministic tree automata $\text{INHAB}_\Sigma^{context}(\text{NTA})$ is EXP-complete, while the deterministic restriction, $\text{INHAB}_\Sigma^{context}(\text{DTA})$ is PSPACE complete. This might be surprising given that intersection nonemptiness is EXP-complete for tree automata, while it is PSPACE-complete for finite automata on words, in both cases independently of determinism (see Fig. 7).

Indeed, we will establish a close correspondence for tree automata between the problem of context inhabitation $\textsc{Inhab}_\Sigma^{context}(\textsc{Nta})$ and the problem of intersection nonemptiness $\textsc{Inter}_\Sigma(\textsc{Dta})$. The surprising result will come from another close relationship between context inhabitation for deterministic tree automata $\textsc{Inhab}_\Sigma^{context}(\textsc{Dta})$ and the intersection nonemptiness problem of deterministic finite automata for words $\textsc{Inter}_\Sigma(\textsc{Dfa})$.

3.2. Intersection NonEmptiness

For any class of automata $\mathcal{A}$ and any signature Σ, the non-uniform version of intersection nonemptiness for a finite number of automata is the following problem.

Inter$_\Sigma(\mathcal{A})$.
Input: a finite number of automata $A_1, \ldots, A_n \in \mathcal{A}_\Sigma$ where $n \geq 0$.
Output: whether $\cap_{i=1}^n L(A_i) \neq \emptyset$.

The uniform variant of this problem where the signature Σ is passed as an input is called $\textsc{Inter}(\mathcal{A})$. Analogous problems can be defined for classes of finite automata on words, i.e. nondeterministic finite automata (Nfas) and deterministic finite automata (Dfas).

In Fig. 7 we recall the complexities of the problems $\textsc{Inter}_\Sigma(\mathcal{A})$ in the cases of deterministic and nondeterministic automata on trees and words, i.e. for $\mathcal{A} \in \{\textsc{Nta}, \textsc{Dta}, \textsc{Nfa}, \textsc{Dfa}\}$. The results hold both for the uniform and the non-uniform variants. In the case of trees, the hardness result requires our assumption that the signature Σ contains at least one constant and one symbol of arity greater than or equal to 2.

3.3. Tree Inhabitation

Before moving to context inhabitation, we reconsider known results on the easier problem of tree inhabitation, that will be instructive for what follows.

For any class of automata $\mathcal{A}$ and any signature Σ, tree inhabitation is the following problem:

	words		trees	
deterministic	DFAs:	PSPACE-c [12]	DTAs:	EXP-c [13]
nondeterministic	NFAs:	PSPACE-c [12]	NTAs:	EXP-c [13]

Figure 7: Emptiness of intersection of a finite number of automata.

	DTAs	NTAs
Tree	P Th 7	EXP-c Th 7
Context	PSPACE-c Th 17	EXP-c Th 14

Figure 8: Inhabitation for Tree Automata.

Inhab$^{tree}_{\Sigma}(\mathcal{A})$.
Input: a tree automaton $A = (Q, \Sigma, F, \Delta) \in \mathcal{A}_\Sigma$, $Q' \subseteq Q$.
Output: whether Q' is Δ-inhabited by some tree in $\mathcal{T}_\Sigma$.

The uniform variant of this problem where the signature Σ is passed as an input is called INHAB$^{tree}(\mathcal{A})$. The complexity of tree inhabitation is folklore, in both cases, uniform or not. An overview of the results is given in Fig. 8. An algorithm for solving the problem for NTAs can be based on determinization. This algorithm will be instructive for context inhabitation as well, so we include it in the proof.

Proposition 6. *Tree inhabitation* INHAB^{tree}(NTA) *is in* EXP. *The restriction to deterministic tree automata* INHAB^{tree}(DTA) *is in* P.

Proof. Let Σ be a ranked signature. If A is a DTA then $Q' \subseteq Q$ is Δ-inhabited by some tree, if either $Q' = \emptyset$ and A is not complete, or Q' is a singleton and the unique state of Q' is accessible wrt. Δ. Hence INHAB^{tree}(DTA) is in polynomial time. For NTA*s* the EXP upper bound can be obtained by determinization. If $A = (Q, \Sigma, F, \Delta)$ is an NTA then by definition $Q' \subseteq Q$ is Δ-inhabited by some tree $t \in \mathcal{T}_\Sigma$ if $[\![t]\!]^\Delta = Q'$. This is equivalent to that

$[\![t]\!]^{\det(\Delta)} = \{Q'\}$. Thus Q' is Δ-inhabited iff $\{Q'\}$ is $\det(\Delta)$-inhabited in $\det(A)$. This can be tested in polynomial time from $\det(A)$, which in turn can be computed in exponential time from A. Thus $\textsc{Inhab}^{tree}(\textsc{Nta})$ is in $\textsc{Exp}$. □

The worst case exponential blow up coming with determinization cannot be avoided for solving tree inhabitation of Ntas, as we show next.

Theorem 7 (Folklore)**.** *Tree inhabitation* $\textsc{Inhab}^{tree}_{\Sigma}(\textsc{Nta})$ *is* $\textsc{Exp}$*-complete, while its restriction to deterministic tree automata* $\textsc{Inhab}^{tree}_{\Sigma}(\textsc{Dta})$ *is in* P*.*

3.4. Context Inhabitation

Since the bound variable of a context occurs exactly once, contexts are interpreted as union homomorphisms in the transition algebras of a tree automata. These will play a key role for defining the problem of context inhabitation and for studying its complexity.

Definition 8. *A union homomorphism on* 2^Q *is a function* $S\colon 2^Q \to 2^Q$ *such that* $S(\emptyset) = \emptyset$ *and* $S(Q' \cup Q'') = S(Q') \cup S(Q'')$ *for all* $Q', Q'' \subseteq Q$*.*

Lemma 9. *For any context* $C \in \mathcal{C}_\Sigma$ *and* Nta $A = (Q, \Sigma, F, \Delta)$ *the* Δ*-inhabited value* $[\![C]\!]^\Delta$ *is a union homomorphism on* 2^Q*.*

The following example shows that Lemma 9 would fail if it were generalized from contexts to nonlinear second-order λ-terms.

Example 1. *Consider* $N = \lambda x.f(x,x)$ *over the signature* $\Sigma = \{a, f\}$ *where* a *is a constant and* f *a symbol of arity* 2*, and the* Nta $A = (Q, \Sigma, F, \Delta)$ *with* $Q = \{q_1, q_2, q_{ok}\}$*,* $F = \{q_{ok}\}$ *and* $\Delta = \{a \to q_1, a \to q_2, f(q_1, q_2) \to q_{ok}\}$*. We have* $[\![N]\!]^\Delta(\{q_1\}) = [\![N]\!]^\Delta(\{q_2\}) = \emptyset$*, while* $[\![N]\!]^\Delta(\{q_1, q_2\}) = \{q_{ok}\}$*. Hence,* $[\![N]\!]^\Delta(\{q_1, q_2\}) \neq [\![N]\!]^\Delta(\{q_1\}) \cup [\![N]\!]^\Delta(\{q_2\})$*, so* $[\![N]\!]^\Delta$ *is not a union homomorphism.*

Any function $s\colon Q \to 2^Q$ defines the union homomorphism $\hat{s}\colon 2^Q \to 2^Q$ such that $\hat{s}(Q') = \bigcup_{q \in Q'} s(q)$ for all $Q' \subseteq Q$. Conversely, any union homomorphism is determined by the images of all singletons.

Lemma 10 (Succinct representations of union homomorphisms)**.** *If* $S\colon 2^Q \to 2^Q$ *is a union homomorphism then* $S = \hat{s}$ *for the function* $s\colon Q \to 2^Q$ *such that* $s(q) = S(\{q\})$ *for all* $q \in Q$*.*

Proof. This is straightforward from the definitions. □

As a consequence, the number of union homomorphisms is equal to the number of functions of type $Q \to 2^Q$ which is exponential. In contrast the number of functions of type $2^Q \to 2^Q$ is doubly exponential. This is the reason why second-order inhabitation is a more difficult problem than context inhabitation that we formalize next.

Context inhabitation receives as input a function $s\colon Q \to 2^Q$ that represents the union homomorphism $\hat{s}\colon 2^Q \to 2^Q$. Note that the representation is exponentially smaller than the union homomorphism it represents. Using this succinct representation of a union homomorphism as an input rather than the union homomorphism itself will permit to relate the complexity of regular matching to context inhabitation.

For any ranked signature Σ and class of automata $\mathcal{A}$, we define the following decision problem.

Inhab$_\Sigma^{context}(\mathcal{A})$.
Input: an automaton $A = (Q, \Sigma, F, \Delta) \in \mathcal{A}_\Sigma$ and a function $s\colon Q \to 2^Q$.
Output: whether the union homomorphism $\hat{s}$ is Δ-inhabited by some context in $\mathcal{C}_\Sigma$.

The uniform variant of the problem where the signature Σ is given with the input is denoted by $\textsc{Inhab}^{context}(\mathcal{A})$. Based on the properties of union homomorphisms, we next show that $\hat{s}$ is Δ-inhabited if and only if its restriction to singletons is.

Proposition 11. *Let* $A = (Q, \Sigma, F, \Delta)$ *be an* NTA *and* $s\colon Q \to 2^Q$*. Then* $\hat{s}$ *is* Δ*-inhabited iff there exists* $C \in \mathcal{C}_\Sigma$ *such that for all* $q \in Q$*,* $s(q) = [\![C]\!]^\Delta(\{q\})$*.*

Proof. The forward implication is straightforward. For the backward direction,

let $C \in \mathcal{C}_\Sigma$ be a context with $s(q) = [\![C]\!]^\Delta(\{q\})$ for all $q \in Q$. Since $\hat{s}$ is a union homomorphism, we have for all $Q' \subseteq Q$ that $\hat{s}(Q') = \bigcup_{q \in Q'} s(q) = \bigcup_{q \in Q'} [\![C]\!]^\Delta(\{q\}) = [\![C]\!]^\Delta(Q')$ since $[\![C]\!]^\Delta$ is a union-homomorphism by Lemma 9. Thus $\hat{s}$ is Δ-inhabited. □

Context inhabitation is a special case of second-order linear λ-definability where the second-order input function is a union homomorphism, except that the union homomorphism is represented in a succinct manner. Note that λ-*definability* is known to be decidable up to the order of three [9], while it is undecidable in general [10, 11]. We now determine the complexity of context inhabitation for NTAs and then for DTAs.

Proposition 12. $\textsc{Inhab}^{context}(\textsc{Nta})$ *is in* Exp.

Proof. As in the case of tree inhabitation, the problem can be solved based on determinization, but in a more tricky manner. Let Σ be a ranked signature, $A = (Q, \Sigma, F, \Delta)$ an NTA where $Q = \{q_1, \ldots, q_n\}$ and $s\colon Q \to 2^Q$. We fix $x \in \mathcal{V}_{tree}$. For each $i \in \{1, \ldots, n\}$, let $\Delta_i = \Delta \cup \{x \to q_i\}$ and $A_i = (Q, \Sigma \uplus \{x\}, F, \Delta_i)$. Let $\tilde{A}$ be the product DTA $\tilde{A} = \det(A_1) \times \ldots \times \det(A_n)$ with transition relation $\tilde{\Delta}$, recognizing the intersection of the languages of the DTAs $\det(A_i)$. Note that the number of states of $\tilde{A}$ is at most $(2^n)^n = 2^{n^2}$, which is exponential.

Claim 13. *Let $p \in \mathcal{T}_{\Sigma \uplus \{x\}}$ be a tree having exactly one occurrence of x. Then $[\![p]\!]^{\tilde{\Delta}} = \{(s(q_1), \ldots, s(q_n))\}$ if and only if $[\![p]\!]^{\Delta_i} = s(q_i)$ for all $1 \leq i \leq n$.*

Recall that for any context $C = \lambda x.p$, the set $[\![p]\!]^{\Delta_i}$ contains all the states to which C can be evaluated when starting at the hole marker x with state q_i. Let B be the DTA with signature $\mathcal{T}_{\Sigma \uplus \{x\}}$ recognizing the set of all trees having exactly one occurrence of x. We assume w.l.o.g. that B has a single final state q_f. Now consider the product DTA $\tilde{A} \times B$ recognizing the language $L(\tilde{A}) \cap L(B)$ of all the elements of $L(\tilde{A})$ having exactly one occurrence of x. Then it follows from Claim 13 that the tuple $(s(q_1), \ldots, s(q_n), q_f)$ is an accessible state of $\tilde{A} \times B$ if and only if there exists a context $\lambda x.p \in \mathcal{C}_\Sigma$ such that $[\![\lambda x.p]\!]^\Delta(\{q_i\}) = s(q_i)$. By Proposition 11 the latter is equivalent to that $\hat{s}$ is Δ-inhabited. Testing

whether $(s(q_1), \ldots, s(q_n), q_f)$ is accessible in $\tilde{A} \times B$ is in polynomial time in the size of $\tilde{A} \times B$, which is in EXP.

□

Theorem 14. $\text{INHAB}_\Sigma^{context}(\text{NTA})$ *is* EXP*-complete.*

Proof. The EXP upper bound was shown in Proposition 12 even for the uniform variant of the problem. For EXP-hardness of $\text{INHAB}_\Sigma^{context}(\text{NTA})$ for any Σ with at least one constant x and one symbol \$ of arity at least 2, we use a reduction from $\text{INTER}_\Sigma(\text{DTA})$. For the sake of simplicity, we assume that \$ is binary, but the following construction can be generalized in the case where \$ has an arity $n > 2$.

Let $A_1, \ldots, A_n$ be DTAs where $A_i = (Q_i, \Sigma, F_i, \Delta_i)$ for $1 \leq i \leq n$, and such that their sets of states are pairwise-disjoint. Let $q_1, q_1^f, \ldots, q_n, q_n^f$ be fresh states, i.e. not in $Q_1 \cup \cdots \cup Q_n$. We build the NTA $B = (Q, \Sigma, F, \Delta)$ obtained by setting $Q = Q_1 \cup \ldots \cup Q_n \cup \{q_1, q_1^f, \ldots, q_n, q_n^f\}$, $F = \{q_1^f, \ldots, q_n^f\}$ and $\Delta = \Delta_1 \cup \ldots \cup \Delta_n \cup \{x \to q_i \mid 1 \leq i \leq n\} \cup \{\$(q, q_i) \to q_i^f \mid q \in F_i\}$. Now let $s \colon Q \to 2^Q$ be the function so that for all $q \in Q$,

$$s(q) = \begin{cases} \{q_i^f\} & \text{if } q = q_i \text{ for } 1 \leq i \leq n \\ \emptyset & \text{otherwise} \end{cases}$$

Then $\bigcap_{i=1}^{n} L(A_i) \neq \emptyset$ if and only if $\hat{s}$ is Δ-inhabited. Indeed,there exists a tree $t \in \bigcap_{i=1}^{n} L(A_i)$ iff $[\![t]\!]^{\Delta_i} \cap F_i \neq \emptyset$ for all $1 \leq i \leq n$, iff $[\![\$(t, x)[x/\{q\}]]\!]^\Delta = s(q)$ for all $q \in Q$ iff $[\![\lambda x.\$(t, x)]\!]^\Delta(\{q\}) = s(q)$ for all $q \in Q$. By Proposition 11, the latter is equivalent to $\hat{s}$ is Δ-inhabited. This concludes the proof of the theorem.

□

We finally show that context inhabitation is in PSPACE for DTAs, even though this is a problem concerning automata for trees and not words. Indeed, inhabitation for DTAs can be reduced to the nonemptiness of intersection for words $\text{INTER}_\Sigma(\text{DFA})$, which is PSPACE-complete [12]. The PSPACE upper bound holds even for the uniform variant of the problem. Showing this requires two

additional tricks in the proof, but is worth the effort since the uniform version of context inhabitation will be needed for solving the uniform version of regular matching, to which the non-uniform version of regular matching with constraints will be reduced. The constraints will allow us to solve regular matching in the case of unranked trees, the original motivation of the present work.

Lemma 15. *The problem* $\textsc{Inhab}^{context}(\textsc{Dta})$ *can be reduced in polynomial time to its restriction where the input function* $s : Q \to 2^Q$ *always maps to singletons.*

Proof sketch. If there exists $q \in Q$ such that $s(q)$ contains more than one element then s cannot be inhabited for any Dta. It remains to remove cases where $s(q) = \emptyset$ for some $q \in Q$. The main idea to deal with empty sets is to complete A to A' by adding a sink state q_{sink} and to replace function s by s' such that $s'(q) = s(q)$ if $s(q) \neq \emptyset$ and $s'(q) = \{q_{sink}\}$ otherwise. Inhabitation of s with respect to A is then equivalent to inhabitation of s' with respect to A'. However, this construction may take exponential time in the maximal arity of function symbols of A, which is not fixed for the uniform problem. This problem can be circumvented by permitting to complete A only partially. The trick that makes this work is presented in the appendix. □

Proposition 16. $\textsc{Inhab}^{context}(\textsc{Dta})$ *is in* Pspace.

Proof sketch. All the technical details of this proof are formally presented in the appendix. Let Σ be a ranked signature, $A = (Q, \Sigma, F, \Delta)$ a Dta and $s : Q \to 2^Q$, where $Q = \{q_1, \ldots, q_n\}$. By Lemma 15, we can assume w.l.o.g. that $s(q_i)$ is a singleton for all $1 \leq i \leq n$. Let $x \in \mathcal{V}_{tree}$ be a tree variable and $\Sigma_x = \Sigma \cup \{x\}$ the ranked signature where x is given arity 0.

We reduce the Δ-inhabitation of $\hat{s}$ to nonemptiness of intersection of $n+1$ related Dtas $A_1, \ldots, A_{n+1}$. For any $i \in \{1, \ldots, n\}$, define the tree automaton $A_i = (Q, \Sigma_x, s(q_i), \Delta_i)$ on Σ_x having the same states as A, set of final states $s(q_i)$, and whose transition relation is $\Delta_i = \Delta \cup \{x \to q_i\}$, and let A_{n+1} be the automaton that accepts the trees having exactly one occurrence of x. We show that $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$ iff $\hat{s}$ is Δ-inhabited. Intuitively, any tree t in the above

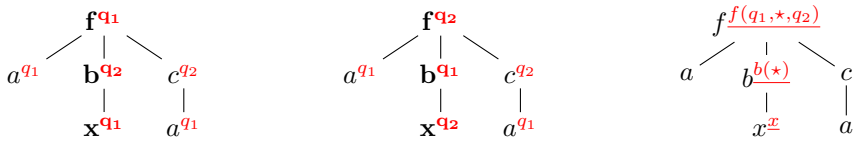


Figure 9: Left, respectively middle: the run (in red superscript) on the tree $f(a, b(x), c(a))$ of the automaton A_1, respectively A_2. Right: the run path (in red superscript) on the tree $f(a, b(x), c(a))$ w.r.t. automata A_1 and A_2.

intersection of languages defines a context $\lambda x.t$ such that $[\![\lambda x.t]\!]^{\Delta}(\{q_i\}) = s(q_i)$ (for any $1 \leq i \leq n$), and this correspondence is bijective.

Now, the automata $A_1, \ldots, A_n$ are related as they differ only by the state in which x is evaluated and by their final state. Actually, for any tree t in Σ_x that contains exactly one occurrence of x, the runs of A_j and A_k differ only along the path from the root of the tree to the leaf x. This is illustrated by the following example. Consider an automaton B on signature $\{f^{(3)}, a^{(0)}, b^{(1)}, c^{(1)}\}$, with states $\{q_1, q_2\}$ and transition relation $\{a \to q_1, b(q_1) \to q_2, b(q_2) \to q_1, c(q_1) \to q_2, f(\_, q_1, \_) \to q_2, f(\_, q_2, \_) \to q_1\}$ (where $\_$ stands for any state). The run of B_1 and of B_2 on the tree $f(a, b(x), c(a))$ are presented on Fig. 9. We now introduce what we call a *run path*, which basically represents the sequences of transitions that are triggered along the path from x to the root of the tree during the evaluation of an automaton B_i, while erasing the way x and its path contribute to the run. Back to the example here above, the run path for B_1 on the tree $f(a, b(x), c(a))$ is presented on the right of Figure 9. It turns out the run path for A_2 is the same.

More formally, define the alphabet Σ_Q that contains the symbol $\underline{x}$ and all symbols of the form $\underline{f(u_1, \ldots, u_m)}$ where f is a symbol in $\Sigma^{(m)}$ and $f(u_1, \ldots, u_m)$ is the left hand side of some rule of A in which exactly one of the states is replaced by $\star$. Back to the example automaton B, e.g. $\underline{f(q_1, q_2, \star)}$ is a letter in Σ_Q because B contains the transition $f(q_1, q_2, q_1) \to q_2$. The letters of Σ_Q are underlined in order to not confuse them with the left hand sides of the actual rules. Then a run path is a word over the alphabet $\Sigma_Q \cup \{\bot\}$, where $\bot$ is a special

symbol that indicates that there is a subtree that the DTA couldn't evaluate. For instance, the run path on Figure 9 is the word $\underline{x}\ \underline{b(\star)}\ \underline{f(q_1, \star, q_2)}$. The run path for $f(a, b(x), c(b))$ is $\underline{x}\ \underline{b(\star)}\ \bot$, as the subtree $c(b)$ cannot be evaluated by the automata B_i.

Naturally, if $t \in \bigcap_{i=1}^{n+1} A_{n+1}$, then it has a corresponding run path with no occurrence of $\bot$. More interestingly, given a run path that doesn't contain the symbol $\bot$, we can always construct a tree "around" it, since all the states of A are accessible. Back to the example, the tree $f(a, b(x), a)$ is a possible tree for the run path $\underline{x}\ \underline{b(\star)}\ \underline{f(q_1, \star, q_1)}$. Now, for any DTA A_i, we construct a DFA A'_i s.t. for all trees t it holds $t \in L(A_i)$ iff the run path of t is in $L(A'_i)$. Thus the nonemptiness of $\bigcap_{i=1}^{n+1} A_{n+1}$ is reduced to the nonemptiness of intersection of n DFAs, which is in PSPACE.

□

Theorem 17. $\text{INHAB}_{\Sigma}^{context}(\text{DTA})$ *is* PSPACE*-complete.*

Proof. The upper bound follows from Proposition 16. The lower bound can be even shown for less general ranked signatures containing only one constant symbol and unary symbols. This is done by reduction from the nonemptiness problem of the intersection of a finite number of DFAs, i.e $\text{INTER}_{\Sigma'}(\text{DFA})$ for some – word – alphabet Σ' containing at least two symbols. Any finite word $a_1, \ldots, a_{m-1}, a_m$ over the alphabet Σ' can be encoded by a "string" tree $a_m(a_{m-1}(\ldots a_1(x)\ldots))$ over the ranked signature Σ, where $a_1, \ldots, a_m$ are unary symbols and x is a constant symbol. Similarly, any DFA A can be transformed in linear time to a DTA A' that accepts exactly the string encodings of words from $L(A)$, and which transitions are trivial encodings of transitions of A plus an additional transition $x \to q^x$ for some fresh state q^x. Now given DFA*s* $A_1, \ldots, A_n$ over alphabet Σ', which we assume w.l.o.g. to have pairwise disjoint sets of states and single final states, let $A'_1, \ldots, A'_n$ be the respective corresponding DTAs over the ranked signature Σ as described above. We write q_i^f to denote the only final state of the DTA A'_i, for $1 \leq i \leq n$. Let B be the union of the A'_i excluding the rules of the form $x \to q_i^x$, and Δ be the transition relation of

$$
\begin{array}{rcll}
p, p', p_1, \ldots, p_n \in \mathcal{P}_\Sigma^{tree} & ::= & x \mid f(p_1, \ldots, p_n) \mid P@p & \\
P \in \mathcal{P}_\Sigma^{context} & ::= & X \mid \lambda x.p' & \text{where } x \text{ occurs exactly once in } p'
\end{array}
$$

Figure 10: Tree and context patterns where $x \in \mathcal{V}^{tree}$, $X \in \mathcal{V}^{context}$, $f \in \Sigma^{(n)}$ and $n \geq 0$.

B. Notice that B is deterministic. Then the intersection of the languages of $A_1, \ldots, A_n$ is non-empty iff $\hat{s}$ is Δ-inhabited, where $\hat{s}$ is defined by $s(q) = \{q_i^f\}$ if $q = q_i^x$ for any $1 \leq i \leq n$, and $s(q) = \emptyset$ otherwise. □

4. Tree Patterns

We next extend trees and contexts to patterns by adding variables of both types. For this we assume a set $\mathcal{V} = \uplus_{\tau \in \mathbf{T}} \mathcal{V}^\tau$ with two kinds of variables. Variables $x, y, z \in \mathcal{V}^{tree}$ have type *tree* and variables $X, Y \in \mathcal{V}^{context}$ type *context*.

We next introduce patterns for trees $p \in \mathcal{P}_\Sigma^{tree}$ and patterns for contexts $P \in \mathcal{P}_\Sigma^{context}$ in Fig. 10. The set of all patterns is $\mathcal{P}_\Sigma = \uplus_{\tau \in \mathbf{T}} \mathcal{P}_\Sigma^\tau$. In both kind of patterns, tree variables x may now occur freely but can also be bound in the scope of a λ-binder as before. For instance, the tree pattern $X@(\lambda y.f(y, a)@x)$ in $\mathcal{P}_\Sigma^{context}$ contains the free context variable X, the bound tree variable y and the free tree variable x. Up to β-reduction this pattern is equal to $X@f(x, a)$ which also belongs to $\mathcal{P}_\Sigma^{context}$.

The set of free variables $fv(p)$ and $fv(P)$ and of bound variables $bv(p)$ and $bv(P)$ are defined as usual for λ-terms. A pattern is called *linear* if each of its free variables has at most one free occurrence.

The set $\mathcal{P}_\Sigma^{gr,\tau}$ of ground patterns of type $\tau \in \mathbf{T}$ is the subset of patterns in $\mathcal{P}_\Sigma^\tau$ without free variables. The set of all ground patterns is denoted by $\mathcal{P}_\Sigma^{gr} = \mathcal{P}_\Sigma^{gr,tree} \cup \mathcal{P}_\Sigma^{gr,context}$. Clearly, any *tree* is a ground pattern of type *tree* and any *context* is a ground pattern of type *context*, i.e., $\mathcal{T}_\Sigma \subseteq \mathcal{P}_\Sigma^{gr,tree}$ and $\mathcal{C}_\Sigma \subseteq \mathcal{P}_\Sigma^{gr,context}$. The converse is not true. The ground pattern $\lambda x.x@f(a)$ for instance is not a tree. However, it becomes equal to the tree $f(a)$ by β-reduction. The situation is similar for ground pattern for contexts, which can

always be reduced to a context by exhaustive β-reduction. The ground context pattern $\lambda x.\lambda y.y@f(x)$ for instance can be β-reduced to the context $\lambda x.f(x)$. In general, each β-reduction step replaces some redex of the form $(\lambda x.p)@p'$ in a bigger pattern by $p[x/p']$ if $x \notin bv(p)$ and otherwise renames x beforehand. Exhaustive β-reduction can be done in any order, but always leads to the same result. We denote the β-reduced form of a tree pattern $p \in \mathcal{P}_\Sigma^{gr,tree}$ by $norm_\beta(p)$ and of a context pattern $P \in \mathcal{P}_\Sigma^{gr,context}$ by $norm_\beta(P)$. The overall reduction requires at most a linear number of steps, since all λ-bound variables in patterns are constrained to occur exactly once (in the scope of the λ-binder). As a consequence, we have $norm_\beta(\mathcal{P}_\Sigma^{gr,tree}) = \mathcal{T}_\Sigma$ and $norm_\beta(\mathcal{P}_\Sigma^{gr,context}) = \mathcal{C}_\Sigma$.

A substitution $\mu\colon V \to \mathcal{P}_\Sigma^{gr}$ on a subset of variables V is called well-typed if it maps tree variables to $\mathcal{P}_\Sigma^{gr,tree}$ and context variables to $\mathcal{P}_\Sigma^{gr,context}$. For any pattern $p \in \mathcal{P}_\Sigma^{tree}$, the grounding $\mu(p) \in \mathcal{P}_\Sigma^{gr,tree}$ is obtained from p by replacing free variables v by $\mu(v)$. The set of all instances of p is obtained by β-normalizing all groundings of p:

$$Inst(p) = \{norm_\beta(\mu(p)) \mid \mu\colon fv(p) \to \mathcal{P}_\Sigma^{gr} \text{ well-typed}\}.$$

Clearly any instance of p is a tree, that is $Inst(p) \subseteq \mathcal{T}_\Sigma$.

Example 2. *Consider the tree pattern $p = X@(X@x)$ and the substitution μ where $\mu(X) = \lambda x.f(b,x)$ and $\mu(x) = a$. The β-normal form of $\mu(p) = \mu(X)@(\mu(X)@\mu(x))$ is the tree $norm_\beta(\mu(p)) = f(b,f(b,a))$ belonging to $Inst(p)$.*

We next lift the algebra interpretation of values to patterns. Let $\sigma\colon V \to [\![Val_\Sigma]\!]^\Delta$ be a well-typed variable assignment in that it maps tree variables to $[\![\mathcal{T}_\Sigma]\!]^\Delta$ and context variables to $[\![\mathcal{C}_\Sigma]\!]^\Delta$. In Fig. 11, we define for any tree pattern p with $fv(p) \subseteq V$ the evaluation $[\![p]\!]^{\Delta,\sigma} \in [\![\mathcal{T}_\Sigma]\!]^\Delta$, and similarly for all context patterns P with $fv(P) \subseteq V$ the interpretation $[\![P]\!]^{\Delta,\sigma} \in [\![\mathcal{C}_\Sigma]\!]^\Delta$.

Note that the interpretation of a ground pattern does not depend on the variable assignment. In this case we can write $[\![p]\!]^\Delta$ instead of $[\![p]\!]^{\Delta,\sigma}$ and $[\![P]\!]^\Delta$ instead of $[\![P]\!]^{\Delta,\sigma}$. Note also that these notations for ground patterns are consistent with the same notations introduced for trees and contexts earlier. Fur-

$$[\![x]\!]^{\Delta,\sigma} = \sigma(x), \qquad [\![f(p_1,\ldots,p_n)]\!]^{\Delta,\sigma} = f^{\Delta}([\![p_1]\!]^{\Delta,\sigma},\ldots,[\![p_n]\!]^{\Delta,\sigma}),$$
$$[\![X]\!]^{\Delta,\sigma} = \sigma(X), \qquad [\![\lambda x.p]\!]^{\Delta,\sigma}(d) = [\![p]\!]^{\Delta,\sigma[x/d]}, \qquad [\![P@p]\!]^{\Delta,\sigma} = [\![P]\!]^{\Delta,\sigma}([\![p]\!]^{\Delta,\sigma}).$$

Figure 11: Algebra interpretation of patterns.

thermore, remark that the algebra interpretation is invariant under β-reduction, i.e. $[\![p]\!]^{\Delta} = [\![norm_{\beta}(p)]\!]^{\Delta}$ and in analogy $[\![P]\!]^{\Delta} = [\![norm_{\beta}(P)]\!]^{\Delta}$.

5. Compressed Tree Patterns

We now show how to define patterns with grammar compression for both types by using a variant of singleton tree grammars with contexts.

Definition 18. *A* compressed pattern *of type $\tau \in \boldsymbol{T}$ is an acyclic context-free tree grammar $G = (N, \Sigma, R, S)$ where $N \subseteq \mathcal{V}$ is a finite set of nonterminals, $S \in N$ of type τ is the start symbol, R is a partial well-typed function from N to patterns in $\mathcal{P}_{\Sigma}$ with free variables in N. The set of all compressed patterns of type τ is denoted by $\mathcal{P}_{\Sigma}^{comp,\tau}$.*

For instance, consider the compressed tree pattern $G_0 \in \mathcal{P}_{\Sigma}^{comp,tree}$ with the nonterminals $N = \{z, X, Y, Z, y\}$, with $S = z$ and with two rules $R(z) = X@a(X@b, Y@c)$, and $R(X) = \lambda x.Z@a(x, y)$. We illustrate G_0 by the graph in Fig. 12. Each nonterminal is annotated to the left of the corresponding node. Note that the circled empty nodes correspond to the nonterminals without any rule. The compressed pattern G_0 is acyclic, in that no variable on the left hand side of some rule does appear in any subsequent rule. In other words, the graph of G_0 is a DAG. It should also be noticed that the tree language of the grammar G_0 is $\emptyset$. What interests us instead is its tree pattern:

$$pat(G_0) = (\lambda x.Z@a(x, y))@a((\lambda x.Z@a(x, y))@b, Y@c)$$

The grammar serves to represent this pattern in a compressed manner, by sharing the context pattern referred to by X. By exhaustive β-reduction of

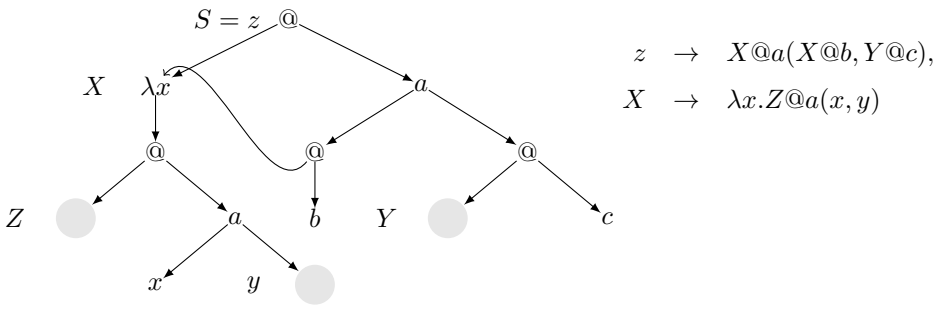


Figure 12: Graph and rules of the compressed tree pattern G_0.

$pat(G_0)$ we obtain the following tree pattern with context variables but without λ-binders:

$$norm_\beta(pat(G_0)) = Z@a(a(Z@a(b, y), Y@c), y)$$

A compressed tree pattern is called compression-free if the structure of its grammar is a tree, that is, every nonterminal appears at most once in all the right-hand sides of the rules. We define the free variables of a compressed tree pattern G as the free variables of $pat(G)$, and the bound variables of G as the nonterminals in $dom(R)$.

In what follows we will identify tree patterns $p \in \mathcal{P}_\Sigma^{tree}$ with the compressed tree pattern $ctp_\Sigma(p) = (\{S\}, \Sigma, \{S \to p\}, S)$, which has a single rule mapping a fixed start symbol S to p. Note that $ctp_\Sigma(p)$ is compression-free. In this sense, $\mathcal{P}_\Sigma^{tree} \subseteq \mathcal{P}_\Sigma^{comp,tree}$. A compressed tree pattern G is called *linear* if its tree pattern $pat(G)$ is linear.

Our next objective is to evaluate compressed tree patterns efficiently over the Σ-algebra of some NTA for a given variable assignment into this algebra. In particular, we want to avoid any kind of decompression when doing so.

The precise formalization of this statement needs a little care, since we have to work with representations of variable assignments as inputs rather than with variable assignment themselves. Let $A = (Q, \Sigma, F, \Delta)$ be an NTA and $\sigma \colon V \to [\![Val_\Sigma]\!]^\Delta$ a well-typed variable assignment into the Σ-algebra $\Delta = (\Sigma, 2^Q, .^\Delta)$. The problem is that the context variables X in V are mapped

to union homomorphisms $\sigma(X)\colon 2^Q \to 2^Q$ (see Definition 8) which may be of exponential size, but can be represented in polynomial space by a function $s(X)\colon Q \to 2^Q$ with $\sigma(X) = \widehat{s(X)}$.

Definition 19. *A function s represents a variable assignment $\sigma\colon V \to [\![Val_\Sigma]\!]^\Delta$ into the Σ-algebra of the* NTA *$A = (Q, \Sigma, F, \Delta)$ if $dom(s) = dom(\sigma)$, $\sigma(X) = \widehat{s(X)}$ for all context variables $X \in dom(s)$, and $\sigma(x) = s(x)$ for all tree variables $x \in dom(s)$. In this case, we write $\sigma = \hat{s}$.*

A similar result to the following lemma can be found for instance in [15].

Lemma 20. *For any* NTA *$A = (Q, \Sigma, F, \Delta)$, compressed tree pattern $G = \mathcal{P}_\Sigma^{comp,tree}$, and representation s of a variable assignment $\hat{s}$ into the Σ-algebra Δ with $fv(G) \subseteq dom(s)$ we can compute the Δ-value of the pattern $[\![pat(G)]\!]^{\Delta,\hat{s}}$ in polynomial time from Σ, Δ, G, and s.*

Proof. The algorithm evaluates the pattern inductively along the partial order on the nonterminals of G; the latter exists because G is acyclic. For any $v \in V$, let G_v be the compressed tree pattern equal to G except that the start symbol is changed to v. Then we can show for all $v \in V$ that $[\![pat(G_v)]\!]^{\Delta,\hat{s}}$ can be computed in polynomial time from Σ, Δ, G, and s. In particular this holds for $[\![pat(G)]\!]^{\Delta,\hat{s}} = [\![pat(G_S)]\!]^{\Delta,\hat{s}}$. □

6. Regular Matching and Inclusion

We now study the complexity of regular matching and inclusion for classes of compressed tree patterns with context variables.

A class of compressed tree patterns $\mathcal{G}$ is a function that maps any signature Σ to a subset of compressed tree patterns $\mathcal{G}_\Sigma \subseteq \mathcal{P}_\Sigma^{comp,tree}$. Typical examples are the classes $\mathcal{P}^{tree}$ and $\mathcal{P}^{comp,tree}$ given that $\mathcal{P}_\Sigma^{tree} \subseteq \mathcal{P}_\Sigma^{comp,tree}$. To see this recall that we identify any tree pattern p with the compression-free compressed tree pattern $ctp_\Sigma(p) = (\{S\}, \Sigma, \{S \to p\}, S)$ where S is the fixed start symbol.

For any class $\mathcal{G}$ of compressed tree patterns, any class $\mathcal{A}$ of NTAs, and for any ranked alphabet Σ we define two decision problems:

Regular pattern inclusion: $\mathbf{Incl}_\Sigma(\mathcal{G}, \mathcal{A})$.
Input: a compressed tree pattern $G \in \mathcal{G}_\Sigma$ and a tree automaton $A \in \mathcal{A}_\Sigma$.
Output: whether $Inst(pat(G)) \subseteq L(A)$.

Regular pattern matching: $\mathbf{Match}_\Sigma(\mathcal{G}, \mathcal{A})$.
Input: a compressed tree pattern $G \in \mathcal{G}_\Sigma$ and a tree automaton $A \in \mathcal{A}_\Sigma$.
Output: whether $Inst(pat(G)) \cap L(A) \neq \emptyset$.

The uniform versions of these problems where the signature Σ is given with the input are called $\textsc{Incl}(\mathcal{G}, \mathcal{A})$ and respectively $\textsc{Match}(\mathcal{G}, \mathcal{A})$.

6.1. Lower Bounds

We first establish the lower bounds for regular matching by reduction from automata intersection problems. In the second step, we establish the lower bounds for the dual problem of regular inclusion. In the deterministic case, the lower bounds for regular matching can be lifted to regular inclusion based on automaton complementation. In the nondeterministic case, another lower bound result needs to be established.

Proposition 21 (Regular matching)**.** $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ *is* Exp*-hard, while* $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ *is* Pspace*-hard.*

Proof. We first notice that $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ generalizes the ground instance intersection problem from [6] by adding compression and context variables. The latter problem is known to be Exp-complete for Ntas, so the Exp-hardness of $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ follows. In order to clarify the role of nondeterminism here, we recall the proof of this result, which is based on a reduction from the nonemptiness of the intersection of a finite number of Dtas, i.e. $\textsc{Inter}_\Sigma(\textsc{Dta})$.

The reduction is as follows. Given a sequence of Dtas $A_1, \ldots, A_n$ over the same signature Σ we can construct in P an Nta A over $\Sigma \cup \{f\}$ that recognizes the language $f(L(A_1), \ldots, L(A_n))$, where f is a fresh function symbol of arity n. The transition relation of A is the union of the transition relations

of $A_1, \ldots, A_n$ extended with rule $f(q_1^f, \ldots, q_n^f) \to q_{ok}$ where q_i^f is the final state of A_i, whose uniqueness can be assumed without loss of generality. Note that A is nondeterministic. We fix a tree variable $x \in \mathcal{V}$ arbitrarily. The regular tree pattern matching task

$$Inst(f(\underbrace{x, \ldots, x}_{n})) \cap L(A) = \emptyset$$

is then equivalent to the intersection emptiness task $L(A_1) \cap \ldots \cap L(A_n) = \emptyset$. To finish the reduction, we note that one can reduce the problem with signature $\Sigma \cup \{f\}$ to the same problem with signature Σ by simulating the new symbol f by the function symbol of arity at least 2 and the constant available in Σ by assumption.

It should be noticed that A is inherently nondeterministic by construction. Therefore, this EXP-hardness proof does not apply to $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$. And indeed, as we will see – in Theorem 27 – this problem is not EXP-hard but only PSPACE-complete. For the moment we show that it's PSPACE-hard.

The PSPACE-hardness of $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ follows from the special case of regular string matching, which was shown to be PSPACE-complete for deterministic finite automata (DFAs) recently [5]. String patterns H over a finite alphabet Γ have the syntax:

$$H, H' \;:=\; a \mid \varepsilon \mid Z \mid HH' \qquad \text{where } a \in \Gamma, Z \in \mathcal{V}^{context}$$

Here we abuse context variables as string variables, i.e., any instantiation of H maps variables to words in Γ^*. String patterns can be translated to tree patterns with context variables as follows: The signature of the trees contains all letters of Γ as monadic function symbols and a fresh constant $\#$. Any string pattern can then be encoded by a monadic tree pattern, such as for instance $aZbZ'c$ by $c(Z'@b(Z@a(\#)))$. In this way, regular string pattern matching can be reduced to regular tree pattern matching with context variables in polynomial time.

□

Lemma 22 (Duality via Complementation)**.** *For any class of compressed tree patterns $\mathcal{G}$, the problems* $\textsc{Incl}_\Sigma(\mathcal{G}, \textsc{Dta})$ *and* $\textsc{coMatch}_\Sigma(\mathcal{G}, \textsc{Dta})$ *are equivalent modulo polynomial time reductions.*

Proof. For any compressed tree pattern G and Dta A, we have $\mathit{Inst}(\mathit{pat}(G)) \subseteq L(A)$ iff $\mathit{Inst}(\mathit{pat}(G)) \cap \overline{L(A)} = \emptyset$ iff $\mathit{Inst}(\mathit{pat}(G)) \cap L(\overline{A}) = \emptyset$, where $\overline{A}$ is the complement automaton for A that can be computed in polynomial time since A is a Dta. □

As a consequence of Lemma 22 the problem $\textsc{Incl}_\Sigma(\mathcal{G}, \textsc{Nta})$ is equivalent to $\textsc{coMatch}_\Sigma(\mathcal{G}, \textsc{Nta})$ modulo Nta determinization, which however requires exponential time. We now show that regular inclusion for Ntas is Exp-hard even for linear tree patterns. Even the class of tree patterns $\mathcal{V}^{tree}$ in which each pattern consists simply of a tree variable is enough. More formally this is the class of compressed tree patterns such that $\mathcal{V}^{tree}_\Sigma = \{\mathit{ctp}_\Sigma(x) \mid x \in \mathcal{V}^{tree}\}$ for all signatures Σ.

Lemma 23. $\textsc{Incl}_\Sigma(\mathcal{V}^{tree}, \textsc{Nta})$ *is* Exp*-hard.*

Proof. Let A be an Nta. The instance set of any pattern $x \in \mathcal{V}^{tree}$ is equal to $\mathcal{T}_\Sigma$. This set is included in $L(A)$ if and only if A is universal. The universality problem for Ntas is well known to be ExpTime-complete. □

Proposition 24 (Regular inclusion)**.** $\textsc{Incl}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ *is* Pspace*-hard, while* $\textsc{Incl}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ *is* Exp*-hard.*

Proof. Lemma 22 states that $\textsc{Incl}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta}) = \textsc{coMatch}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ modulo polynomial time reductions. By Proposition 21, $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ is Pspace-hard and since Pspace is closed by complement, $\textsc{coMatch}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ is Pspace-hard too. Hence $\textsc{Incl}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ is Pspace-hard.

In the case of Ntas, the Exp-hardness of $\textsc{Incl}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ follows immediately from Lemma 23. □

6.2. Upper Bounds

All upper bounds will be obtained in a systematic manner by some algorithm that instead of guessing trees or contexts in Val_Σ will guess Δ-inhabited values in $[\![Val_\Sigma]\!]^\Delta$. For the guessing, a subroutine will be applied that decides tree or context inhabitation.

We start with a characterization of regular matching and inclusion, on which our decision procedure will rely.

Lemma 25 (Characterization)**.** *Let $A = (Q, \Sigma, F, \Delta)$ be an* NTA *and $p \in \mathcal{P}^{tree}_\Sigma$ a tree pattern.*

Regular matching: *$Inst(p) \cap L(A) \neq \emptyset$ holds iff there exists some well-typed variable assignment to Δ-inhabited values $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$ such that $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$.*

Regular inclusion: *$Inst(p) \subseteq L(A)$ holds iff all well-typed variable assignments to Δ-inhabited values $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$ satisfy $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$.*

Proof. We start with the case of regular matching. For the forward direction, we assume $Inst(p) \cap L(A) \neq \emptyset$. By definition of instances, there exists a well-typed assignment $\mu\colon fv(p) \to Val_\Sigma$ such that $norm_\beta(\mu(p)) \in L(A)$. Let $\sigma = [\![.]\!]^\Delta \circ \mu$. Clearly $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$ is a well-typed variable assignment. Since $[\![p]\!]^{\Delta,\sigma} = [\![\mu(p)]\!]^\Delta = [\![norm_\beta(\mu(p))]\!]^\Delta$ it follows that $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$.

For the inverse direction, we fix a well-typed variable assignment to Δ-inhabited values $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$ such that $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$. By Δ-inhabitation there exists a well-typed variable assignment $\mu\colon fv(p) \to Val_\Sigma$ such that $\sigma = [\![.]\!]^\Delta \circ \mu$. Hence, $[\![\mu(p)]\!]^\Delta \cap F \neq \emptyset$, so that $[\![norm_\beta(\mu(p))]\!]^\Delta \cap F \neq \emptyset$. Thus $norm_\beta(\mu(p)) \in L(A)$, that is $norm_\beta(\mu(p)) \in Inst(p) \cap L(A)$.

The case of regular inclusion is similar. For the forward direction, we assume $Inst(p) \subseteq L(A)$ and fix a variable assignment to Δ-inhabited values $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$. By Δ-inhabitation, there exists a variable assignment $\mu\colon fv(p) \to Val_\Sigma$ such that $\sigma = [\![.]\!]^\Delta \circ \mu$. Since $norm_\beta(\mu(p)) \in Inst(p)$ it follows from $Inst(p) \subseteq$

$L(A)$ that $norm_\beta(\mu(p)) \in L(A)$. Therefore, it follows from $[\![p]\!]^{\Delta,\sigma} = [\![\mu(p)]\!]^\Delta = [\![norm_\beta(\mu(p))]\!]^\Delta$ that $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$.

For the inverse direction, we assume that any variable assignment to Δ-inhabited values $\sigma\colon fv(p) \to [\![Val_\Sigma]\!]^\Delta$ satisfies $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$. We fix an element of $t \in Inst(p)$, which must be of the form $t = norm_\beta(\mu(p))$ for some $\mu\colon fv(p) \to Val_\Sigma$. The variable assignment $\sigma = [\![.]\!]^\Delta \circ \mu$ then maps to Δ-inhabited values, so that by assumption $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$. Since $[\![p]\!]^{\Delta,\sigma} = [\![\mu(p)]\!]^\Delta = [\![norm_\beta(\mu(p))]\!]^\Delta = [\![t]\!]^\Delta$ it follows that $t \in L(A)$. □

We now show how to decide regular matching and inclusion based on algorithms with oracles for solving inhabitation problems. Given two complexity classes Ξ_1 and Ξ_2, we will write $\Xi_1(\Xi_2)$ for problems solvable in Ξ_1 when having an oracle in Ξ_2. We recall in particular that $\mathrm{NP}(\Xi) \subseteq \textsc{Exp}(\Xi)$, $\mathrm{coNP}(\Xi) \subseteq \textsc{Exp}(\Xi)$ and that $\textsc{Exp}(\textsc{Exp}) \subseteq \textsc{Exp}$. As a consequence, $\mathrm{NP}(\textsc{Exp}) \subseteq \textsc{Exp}$ and $\mathrm{coNP}(\textsc{Exp}) \subseteq \textsc{Exp}$. We also equip $\mathbf{T}$ with the total order $\leq_{\mathbf{T}}$ defined by *tree* $\leq_{\mathbf{T}}$ *context*.

Proposition 26. *Let $\mathcal{G}$ be a class of compressed tree patterns and $\mathcal{A}$ a class of* NTAs*. Let τ be the maximal type of free variables in a pattern in $\mathcal{G}$ wrt. $\leq_{\boldsymbol{T}}$ and suppose that $\textsc{Inhab}^\tau(\mathcal{A})$ belongs to complexity class Ξ. In this case, $\textsc{Match}(\mathcal{G}, \mathcal{A})$ belongs to $\mathrm{NP}(\Xi)$ and $\textsc{Incl}(\mathcal{G}, \mathcal{A})$ to $\mathrm{coNP}(\Xi)$.*

Proof. Let Σ be a ranked signature, $G = (N, \Sigma, \_, S)$ a compressed tree pattern of type *tree* in class $\mathcal{G}$, and $A = (Q, \Sigma, F, \Delta)$ be a tree automaton in class $\mathcal{A}$. According to Lemma 25, $pat(G)$ matches $L(A)$ iff some well-typed variable assignment $\sigma\colon fv(G) \to [\![Val_\Sigma]\!]^\Delta$ satisfies $[\![pat(G)]\!]^{\Delta,\sigma} \cap F \neq \emptyset$. For all context variables $X \in fv(G)$, the value $\sigma(X)$ belongs to $[\![\mathcal{C}_\Sigma]\!]^\Delta$ so it is a union homomorphism. Therefore, σ can be associated to a function s representing it in the sense of Definition 19. In order to find a suitable value for $\sigma(X)$, we guess a function $s(X)\colon Q \to 2^Q$ of which there are exponentially many (while the number of functions of type $2^Q \to 2^Q$ is doubly exponential) and test whether $\widehat{s(X)}$ is Δ-inhabited. The procedure is analogous for tree variables $x \in fv(G)$, except that sets of states $s(x) \subseteq Q$ are guessed and tested for inhabitation.

The inhabitation test is an instance of $\textsc{Inhab}^{\tau}(\mathcal{A})$ which can be done by an Ξ oracle by assumption. Therefore, the guessing can be done by an algorithm in $\mathrm{NP}(\Xi)$. After having found Δ-inhabited values for all the free variables of G, the computation of $[\![pat(G)]\!]^{\Delta,\sigma}$ which equals to $[\![pat(G)]\!]^{\Delta,\hat{s}}$ can be done in polynomial time by Lemma 20, so the characterization of regular matching can be tested by an algorithm in $\mathrm{NP}(\Xi)$.

For $\textsc{Incl}(\mathcal{G},\mathcal{A})$, the procedure is almost the same, except that by Lemma 25 we now have to guess a representation of a variable assignment $s\colon fv(G) \to [\![Val_\Sigma]\!]^\Delta$ such that $[\![pat(G)]\!]^{\Delta,\hat{s}} \cap F = \emptyset$ in order to contradict regular inclusion. This can be done by an algorithm in $\mathrm{coNP}(\Xi)$. □

We next establish the complexity of the regular matching and inclusion problems.

Theorem 27. $\textsc{Match}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Dta})$ *and* $\textsc{Incl}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Dta})$ *are* Pspace*-complete, while* $\textsc{Match}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Nta})$ *and* $\textsc{Incl}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Nta})$ *are* Exp*-complete.*

Proof. The hardness results were shown in Proposition 21 and 24, so only the upper bounds remain to be proven. Let Σ be a ranked signature.

On one hand, since $\textsc{Inhab}^{context}(\textsc{Dta})$ is in Pspace by Theorem 17, it follows from Proposition 26 that $\textsc{Match}(\mathcal{P}^{comp,tree}, \textsc{Dta})$ is in $\mathrm{NP}(\textsc{Pspace})$ and thus in $\textsc{NPspace} \subseteq \textsc{Pspace}$ by Savitch's Theorem [16]. It also follows that $\textsc{Incl}(\mathcal{P}^{comp,tree}, \textsc{Dta})$ is in $\mathrm{coNP}(\textsc{Pspace})$ which is in $\textsc{coNPspace} = \textsc{NPspace}$ and thus in Pspace too. This allows to conclude that the problems $\textsc{Match}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Dta})$ and $\textsc{Incl}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Dta})$ are in Pspace.

On the other hand, since $\textsc{Inhab}^{context}(\textsc{Nta})$ is in Exp by Theorem 17, it follows by Proposition 26 that $\textsc{Match}(\mathcal{P}^{comp,tree}, \textsc{Nta})$ is in $\mathrm{NP}(\textsc{Exp})$ and that $\textsc{Incl}(\mathcal{P}^{comp,tree}, \textsc{Nta})$ is in $\mathrm{coNP}(\textsc{Exp})$. Hence both problems are in Exp, which imply that $\textsc{Match}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Nta})$ and $\textsc{Incl}_\Sigma(\mathcal{P}^{comp,tree}, \textsc{Nta})$ are also in Exp. □

7. Adding Regular Constraints

So far, regular matching and inclusion consider all the possible instances of the compressed tree pattern given as input, but it may be interesting to consider only instances satisfying some constraints. This is the case when schemas are defined for XML documents. In this part, we generalize the regular matching and inclusion problems by allowing constraints restricting how free variables are instantiated. Let Σ be a ranked signature and G a compressed tree pattern over Σ. An instantiation constraint ρ on G is a total function that maps every free tree variable of G to a DTA over Σ and every free context variable of G to a DTA over $\Sigma \uplus \{x_\rho\}$ where $x_\rho \in \mathcal{V}^{tree}$. Furthermore, DTAs associated with context variables are allowed to recognize only languages of trees having exactly one occurrence of x_ρ. Note that x_ρ is used to indicate the position of the *hole* in the contexts, that is the variable to be instantiated. A well-typed variable assignment $\sigma : \mathit{fv}(G) \to \mathcal{P}^{gr}_\Sigma$ satisfies ρ if for every free tree variable $x \in \mathit{fv}(G)$, $\mathit{norm}_\beta(\sigma(x)) \in L(\rho(x))$ and for every free context variable $X \in \mathit{fv}(G)$, $\mathit{norm}_\beta(\sigma(X)@x_\rho) \in L(\rho(X))$. We can now define the set of instances of $p_G = \mathit{pat}(G)$ that satisfy ρ as the set:

$$\mathit{Inst}^\rho(p_G) = \{\mathit{norm}_\beta(\sigma(p_G)) \mid \sigma : \mathit{fv}(G) \to \mathcal{P}^{gr}_\Sigma \text{ well-typed and satisfies } \rho\}.$$

For any class of compressed tree patterns $\mathcal{G}$ and of NTAs $\mathcal{A}$ and any ranked signature Σ, the problems of regular pattern inclusion and matching with constraints are the following:

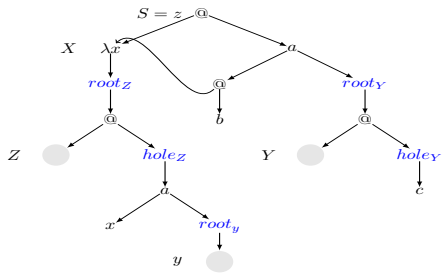
Regular pattern inclusion with constraints: cIncl$_\Sigma(\mathcal{G}, \mathcal{A})$.
Input: a compressed tree pattern $G \in \mathcal{G}_\Sigma$, a tree automaton $A \in \mathcal{A}_\Sigma$ and an instantiation constraint $\rho : \mathit{fv}(G) \to \text{DTA}_\Sigma \cup \text{DTA}_{\Sigma \uplus \{x_\rho\}}$.
Output: whether $\mathit{Inst}^\rho(\mathit{pat}(G)) \subseteq L(A)$.

Regular pattern matching with constraints: cMatch$_\Sigma(\mathcal{G}, \mathcal{A})$.
Input: a compressed tree pattern $G \in \mathcal{G}_\Sigma$, a tree automaton $A \in \mathcal{A}_\Sigma$ and an instantiation constraint $\rho : fv(G) \to \text{DTA}_\Sigma \cup \text{DTA}_{\Sigma \uplus \{x_\rho\}}$.
Output: whether $Inst^\rho(pat(G)) \cap L(A) \neq \emptyset$.

The uniform versions of these problems, where the signature can vary with the input, are written cMatch$(\mathcal{G}, \mathcal{A})$ and cIncl$(\mathcal{G}, \mathcal{A})$. It can easily be seen that regular matching (resp. regular inclusion) is a special case of regular matching with constraints (resp. regular inclusion with matching), and that an algorithm for the general case can be used to solve the special case. What is more interesting is that regular matching with constraints (resp. regular inclusion with constraints) can also be reduced to uniform regular matching (resp. uniform regular inclusion), as stated in the next proposition:

Proposition 28. *For any class $\mathcal{G}$ of compressed tree patterns and any class of tree automata $\mathcal{A} \in \{\text{NTA}, \text{DTA}\}$,* cMatch$(\mathcal{G}, \mathcal{A})$ *and* cIncl$(\mathcal{G}, \mathcal{A})$ *are reducible in polynomial time to respectively* Match$(\mathcal{G}, \mathcal{A})$ *and* Incl$(\mathcal{G}, \mathcal{A})$.

Proof. Let $\mathcal{G}$ be a class of compressed tree patterns, $\mathcal{A} \in \{\text{NTA}, \text{DTA}\}$ a class of tree automata, Σ a ranked signature, $G \in \mathcal{G}_\Sigma$ a compressed tree pattern, $A = (Q, \Sigma, F, \Delta) \in \mathcal{A}_\Sigma$ a tree automaton and ρ an instantiation constraint on G. The general idea is to build a new compressed tree pattern wherein there are places marked as *test zones*, that is, places that tell the automaton where constraints should be tested. Then we restrict the instances of this compressed tree pattern to the instances that satisfy ρ using two new automata, before testing matching and inclusion. We first associate to every free tree variable $x \in \mathcal{V}^{tree} \cap fv(G)$ a fresh unary symbol $root_x$ and to every free context variable $X \in \mathcal{V}^{context} \cap fv(G)$ two fresh unary symbols $root_X, hole_X$. These symbols, called markers, are used to delimit the test zones. Let $\Theta = \{root_\nu \mid \nu \in fv(G)\} \cup \{hole_X \mid X \in fv(G) \cap \mathcal{V}^{context}(G)\}$ be the set of markers. We define a function $mark_\Sigma$ that associates every compressed tree pattern G_1 over Σ with a new compressed tree pattern G_2 over $\Sigma \cup \Theta$ that is almost equal to G_1, except



$$
\begin{aligned}
z &\rightarrow root_X(X@hole_X(a(root_X(X@hole_X(b)), root_Y(Y@hole_Y(c))))),\\
X &\rightarrow \lambda x.root_Z(Z@hole_Z(a(x, root_y(y))))
\end{aligned}
$$

Figure 13: $G' = mark_\Sigma(G)$ built from the compressed tree pattern G in Figure 12.

that

- every occurrence of a free tree variable $x \in fv(G_1)$ in G_1 is replaced by $root_x(x)$ in G_2
- every subterm $X@p$ of G_1 where $X \in fv(G_1)$ is a free context variable and $p \in \mathcal{P}_\Sigma$ a pattern is replaced by $root_X(X@hole_X(p))$ in G_2

Figure 13 illustrates the compressed tree pattern G' obtained after applying the $mark_\Sigma$ function on the compressed tree pattern of Figure 12.

Let A' be the automaton over $\Sigma \cup \Theta$ built from A, so that $L(A') = \{mark_\Sigma(t) \mid t \in L(A)\}$. A' can be built in linear time from A such that if A is deterministic, then so will be A'. We now build a new NTA B that will allow to test the constraints specified in ρ. Let q_{wait} be a fresh state. The state q_{wait} is the state in which B waits before testing a constraint, but also its final state. For every part of its input, B guesses whether it's in a test zone, and guesses the constraint to test. Thus, if B is reading the test zone of some free variable $\nu \in fv(G)$, it runs the automaton $\rho(\nu)$. If the constraint in $\rho(\nu)$ is satisfied, B returns to q_{wait} and waits for the next constraint to test. However, if no constraint is satisfied in a test zone, B blocks and doesn't get back to q_{wait}. For all $\nu \in fv(h)$, define Q_ν as the set of states of $\rho(\nu)$ and Δ_ν as its transition relation. We set

$B = (Q_B, \Sigma \cup \Theta, \{q_{wait}\}, \Delta_B)$ where $Q_B = \{q_{wait}\} \cup \bigcup_{\nu \in fv(G)} Q_\nu$. The transition relation Δ_B is defined as the union of Δ_ν for all $\nu \in fv(G)$, plus the following updates:

1. for all $f \in \Sigma^{(n)}$ where $n \geq 0$, add $f(\underbrace{q_{wait}, \ldots, q_{wait}}_{n}) \to q_{wait}$ to Δ_B
2. for all $X \in fv(G) \cap \mathcal{V}^{context}$, replace the only rule $x_\rho \to q_X$ by $hole_X(q_{wait}) \to q_X$ in Δ_B
3. for all $\nu \in fv(G)$ and final state q_ν^f of $\rho(\nu)$, add $root_\nu(q_\nu^f) \to q_{wait}$ to Δ_B.

Note the rule (2) that allows to simulate the reading of x_ρ by constraint automata of contexts. So x_ρ is not in the signature of B. Furthermore, B checks only whether the constraints that have been tested are satisfied, but cannot guarantee that all the constraints are tested. For this, one could have built an automaton that tests whether all the occurrences of all the variables of G' are instantiated. However, the instance set of G' is not a regular language in general. Instead, a DTA C that just tests whether all the variables of G' have one occurrence that is instantiated is enough. C is built in a way that it recognizes all the trees that have the same skeleton as G'. By same skeleton we mean that the language of trees recognized by C is a regular language obtained by removing all non-linearities from G'. By replacing for instance the occurrences of variables – bound or free – in G' that are not first occurrences by fresh variables, we have a new compressed tree pattern whose instance set is a regular language. The DTA C recognizes this language. We illustrate in Figure 14 a compressed tree pattern obtained with this construction. Notice the new free variable X' replacing the second occurrence of X. Now remark that for some tree $t \in \mathcal{T}_\Sigma$, $t \in Inst^\rho(pat(G))$ if and only if $mark_\Sigma(t) \in Inst(pat(G')) \cap L(B) \cap L(C)$.

The main problem with our reduction is that B is not deterministic, although it is built from the DTAs $\rho(\nu)$. In order to solve this, we consider a new ranked signature Σ' where symbols $f \in \Sigma$ are associated to the variables ν, such that the tuple (f, ν) is used only in some instantiation of ν. More formally, $\Sigma' =$

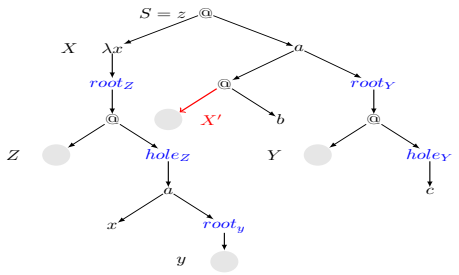


Figure 14: Compressed tree pattern built from G' in Figure 13 and used to build C.

$\Theta \cup \Sigma \cup (\Sigma \times fv(G))$. We modify G', A', B, and C to take into account the new signature Σ'. For G' we build a new compressed tree pattern G'' in linear time, that is equal to it but has the extended signature Σ'. Note that G'' is linear if G is. For B, we construct in linear time a DTA B' over Σ' equal to B except that every rule $f(q_1, \ldots, q_n) \to q \in \Delta_B$ – where $n \geq 0$ – that originates from a DTA $\rho(\nu)$ for some $\nu \in fv(G)$ is *replaced* by $(f, \nu)(q_1, \ldots, q_n) \to q$. This way, the set of rules of B is partitioned, according to their automata $\rho(\nu)$ of origin. Assuming – w.l.o.g. – that the state sets of the automata $\rho(\nu)$ for $\nu \in fv(G)$ are disjoint, B' is indeed deterministic. Another consequence is that all letters of an instance of a free variable $\nu \in fv(G)$ must be annotated by the free variable ν itself. Unlike B, B' does not need to guess the constraint to test, as this is now indicated in the input. Finally, for A' and C, we build automata A'' and C' over Σ' – in polynomial time – so that for any rule $f(q_1, \ldots, q_n) \to q$ – where $n \geq 0$ – of their transition relations and any free variable $\nu \in fv(G)$, a new rule $(f, \nu)(q_1, \ldots, q_n) \to q$ is added.

Now observe that

Claim 29. *There exists a bijection* $\varphi : Inst^{\rho}(pat(G)) \to Inst(pat(G'')) \cap L(B') \cap L(C')$ *such that for all* $t \in Inst^{\rho}(pat(G))$, $t \in L(A)$ *if and only if* $\varphi(t) \in L(A'')$.

Using Claim 29, we show that one can build an automaton D (resp. D') with signature Σ' such that $Inst^{\rho}(pat(G)) \cap L(A) \neq \emptyset$ (resp. $Inst^{\rho}(pat(G)) \subseteq L(A)$) if and only if $Inst(pat(G'')) \cap L(D) \neq \emptyset$ (resp. $Inst(pat(G'')) \subseteq L(D')$). This

allows to reduce uniform regular matching (resp. inclusion) with constraints to uniform regular matching (resp. inclusion).

Claim 30. $\textsc{cMatch}(\mathcal{G}, \mathcal{A})$ *is reducible in polynomial time to* $\textsc{Match}(\mathcal{G}, \mathcal{A})$.

Claim 31. $\textsc{cIncl}(\mathcal{G}, \mathcal{A})$ *is reducible in polynomial time to* $\textsc{Incl}(\mathcal{G}, \mathcal{A})$.

The Proposition thus follows from Claims 30 and 31.

Note that because of the constraints, Σ' depends not only on Σ, but also on G. So $\textsc{cMatch}_\Sigma(\mathcal{G}, \mathcal{A})$ (resp. $\textsc{cIncl}_\Sigma(\mathcal{G}, \mathcal{A})$) cannot be reduced to $\textsc{Match}_{\Sigma''}\mathcal{G}\mathcal{A}$, (resp. $\textsc{Incl}_{\Sigma''}\mathcal{G}\mathcal{A}$) for some signature Σ'', but only to $\textsc{Match}(\mathcal{G}, \mathcal{A})$ (resp. $\textsc{Incl}(\mathcal{G}, \mathcal{A})$). □

8. Encoding Patterns for Unranked Trees

The original motivation of the present work was to understand the problems of regular matching and inclusion for hedge patterns. We next show that these problems can be solved using reductions to the corresponding problems of (ranked) tree patterns with context variables.

Unlike ranked trees, unranked trees are constructed from symbols without fixed arities. We fix a finite set Γ of such symbols. The set of hedges $\mathcal{H}_\Gamma$ is the least set that contains all words of hedges in ${\mathcal{H}_\Gamma}^*$ and all pairs $a(H)$ where $a \in \Gamma$ and $H \in \mathcal{H}_\Gamma$ is a hedge. The set of unranked trees $\mathcal{U}_\Gamma$ is the subset of hedges of the form $a(H)$.

We assume a set of variables for unranked trees $Y \in \mathcal{V}^u$ and a set of hedge variables $Z \in \mathcal{V}^h$. The set of hedge patterns $H \in \mathcal{P}^h_\Gamma$ with these two types of variables is then defined by the abstract syntax in Fig. 15. The set $\mathcal{P}^u_\Gamma$ of *patterns for unranked trees* is the subset of hedge patterns of the forms $a(H)$ or $Y \in \mathcal{V}^u$. The set of free variables $fv(H)$ is defined as usual. A well-typed variable assignment $\sigma : V \to \mathcal{H}_\Gamma$ where $V \subseteq \mathcal{V}^u \uplus \mathcal{V}^h$ is a function that maps variables from $\mathcal{V}^u$ to unranked trees in $\mathcal{U}_\Gamma$ and variables from $\mathcal{V}^h$ to hedges in $\mathcal{H}_\Gamma$. The application $\sigma(H)$ is the hedge obtained from H by replacing all variables Y by the unranked tree $\sigma(Y)$ and all variables Z by the hedge $\sigma(Z)$. The instance

Hedge patterns	$H, H' \in \mathcal{P}^h_\Gamma \quad ::= \quad Y \mid a(H) \mid \varepsilon \mid Z \mid HH'$
Encoding	$\langle \varepsilon \rangle^{context} = \lambda y.y, \quad \langle H \rangle^{tree} = \langle H \rangle^{context} @ \#,$
	$\langle Y \rangle^{context} = Y, \quad \langle Z \rangle^{context} = Z,$
	$\langle a(H) \rangle^{context} = \lambda y.a(\langle H \rangle^{context} @ \#, y),$
	$\langle HH' \rangle^{context} = \lambda y.(\langle H \rangle^{context} @ (\langle H' \rangle^{context} @ y)).$

Figure 15: Encoding of a hedge pattern $H \in \mathcal{P}^h_\Gamma$ into a context pattern $\langle H \rangle^{context} \in \mathcal{P}^{context}_\Sigma$, where $Y \in \mathcal{V}^u$, $Z \in \mathcal{V}^h$, $a \in \Gamma$, and ε is the empty word.

set of H is denoted $Inst(H) = \{\sigma(H) \mid \sigma : fv(H) \to \mathcal{H}_\Gamma \text{ well-typed}\}$. Note that $Inst(H) \subseteq \mathcal{U}_\Gamma$ for any unranked tree pattern $H \in \mathcal{P}^u_\Gamma$.

We next show in Fig. 15 how to encode hedge patterns into (ranked) context patterns over a ranked signature $\Sigma = \Sigma^{(2)} \uplus \Sigma^{(0)}$ where $\Sigma^{(2)} = \Gamma$, $\Sigma^{(0)} = \{\#\}$ and $\#$ is a fresh symbol not in Γ. Our encoding is an extension of the *first-child-next-sibling* encoding [6]. For instance, the hedge pattern $H_0 = a(ZbcY)$ is encoded into the context pattern $\langle H_0 \rangle^{context} = \lambda y.a(Z@(b(\#, c(\#, Y@\#))), y)$. The concatenation operation on hedges is simulated by the application operation of contexts. The set of context variables used in the encoding is $\mathcal{V}^{context} = \mathcal{V}^u \uplus \mathcal{V}^h$. Finally, we define for any unranked tree $H \in \mathcal{P}^u_\Gamma$ its encoding as a tree pattern $\langle H \rangle^{tree} \in \mathcal{P}^{tree}_\Sigma$ by $\langle H \rangle^{tree} = \langle H \rangle^{context} @ \#$.

In order to show the soundness of this encoding (Lemma 32 below), we need to restrict the instantiation operation. Intuitively, we cannot allow arbitrary substitutions to be applied to $\langle H \rangle^{tree}$ because then the resulting tree pattern might not be a correct encoding of an unranked tree. A variable assignment $\sigma : V \to Val_\Sigma$ is called *unranked* if it maps unranked tree variables to $\langle \mathcal{U}_\Gamma \rangle^{context}$ and hedge variables to $\langle \mathcal{H}_\Gamma \rangle^{context}$. The *unranked-restricted instance set* of a tree pattern p is defined by $Inst^{unr}(p) = \{norm_\beta(\sigma(p)) \mid \sigma : fv(p) \to Val_\Sigma \text{ well-typed and unranked}\}$.

Lemma 32. *$norm_\beta(\langle Inst(H) \rangle^{tree}) = Inst^{unr}(\langle H \rangle^{tree})$ for any $H \in \mathcal{P}^u_\Gamma$.*

Let $\mathcal{P}^{comp,u}_\Gamma$ be the set of compressed unranked tree patterns over Γ, defined

in an analogous way as compressed tree patterns. For a class of automata $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$ we define problems of regular matching and inclusion of compressed unranked tree patterns:

Unranked regular matching: $\mathbf{Match}_\Gamma(\mathcal{P}^{comp,u}, \mathcal{A})$.
Input: an unranked tree pattern $H \in \mathcal{P}_\Gamma^{comp,u}$ and an automaton $A \in \mathcal{A}_\Sigma$
Output: whether $Inst^{unr}(\langle H \rangle^{tree}) \cap L(A) \neq \emptyset$.

Unranked regular inclusion: $\mathbf{Incl}_\Gamma(\mathcal{P}^{comp,u}, \mathcal{A})$.
Input: an unranked tree pattern $H \in \mathcal{P}_\Gamma^{comp,u}$ and an automaton $A \in \mathcal{A}_\Sigma$
Output: whether $Inst^{unr}(\langle H \rangle^{tree}) \subseteq L(A)$.

The uniform versions of these problems where the signature Γ is given with the input are called $\text{MATCH}(\mathcal{P}^{comp,u}, \mathcal{A})$ and respectively $\text{INCL}(\mathcal{P}^{comp,u}, \mathcal{A})$. Note that using tree automata in the above definitions is not a restriction, as it is well known [6] that for any unranked tree language L recognizable by a hedge automaton, there exists a tree automaton that recognizes the first-child-next-sibling encoding of the trees in L.

Proposition 33. *For any class of automata $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$ there exist reductions in polynomial time from $\text{MATCH}(\mathcal{P}^{comp,u}, \mathcal{A})$ to $\text{MATCH}(\mathcal{P}^{comp,tree}, \mathcal{A})$ and from $\text{INCL}(\mathcal{P}^{comp,u}, \mathcal{A})$ to $\text{INCL}(\mathcal{P}^{comp,tree}, \mathcal{A})$.*

Proof. Let Γ be an alphabet, $\Sigma = \Sigma^{(2)} \cup \Sigma^{(0)}$ a ranked signature constituted of binary symbols taken from Γ and a constant $\#$, that is $\Sigma^{(2)} = \Gamma$, $\Sigma^{(0)} = \{\#\}$ and $\# \notin \Gamma$. Let $H \in \mathcal{P}_\Gamma^{comp,u}$ be a compressed pattern, $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$ a class of automata and $A \in \mathcal{A}_\Sigma$ a tree automaton. Thanks to Lemma 32, we have $norm_\beta(\langle Inst(H) \rangle^{tree}) = Inst^{unr}(\langle H \rangle^{tree})$, and thus deciding whether $norm_\beta(\langle Inst(H) \rangle^{tree}) \cap L(A) \neq \emptyset$ is equivalent to deciding whether the inequality $Inst^{unr}(\langle H \rangle^{tree}) \cap L(A) \neq \emptyset$ holds. Notice that *unr* is actually an instantiation constraint. It associates every free tree variable with the universal DTA over Σ. Context variables are mapped to the DTA that recognizes all the trees over $\Sigma \uplus \{y\}$ having only one occurrence of y, which is furthermore either the only

node of the tree, or the second son of its parent, as enforced by the encoding. We have thus reduced the problem of regular matching of compressed unranked tree patterns to the problem of regular matching with constraints – on ranked patterns – in polynomial time. Then the regular matching problem with constraints is reduced to uniform regular matching using Proposition 28. We use an analogous procedure for the inclusion problem. □

Theorem 34. *For any alphabet* Γ *having at least two symbols, the problems* $\textsc{Match}_\Gamma(\mathcal{P}^{comp,u}, \textsc{Dta})$ *and* $\textsc{Incl}_\Gamma(\mathcal{P}^{comp,u}, \textsc{Dta})$ *are* Pspace*-complete while* $\textsc{Match}_\Gamma(\mathcal{P}^{comp,u}, \textsc{Nta})$ *and* $\textsc{Incl}_\Gamma(\mathcal{P}^{comp,u}, \textsc{Nta})$ *are* Exp*-complete.*

Proof. The upper bounds follow via the polynomial time reduction from Proposition 33 and the complexities in Proposition 26. The lower bounds can be obtained by reducing the equivalent problems on ranked patterns to the version on unranked patterns, and further using the results in Propositions 21 and 24. □

9. Linearity Restriction

We now study the complexity of regular matching and inclusion for the class $\mathit{Lin}\mathcal{P}^{comp,tree}$ that maps ranked signatures Σ to the set of linear compressed tree patterns $\mathit{Lin}\mathcal{P}^{comp,tree}_\Sigma$.

Proposition 35. $\textsc{Match}(\mathit{Lin}\mathcal{P}^{comp,tree}, \textsc{Nta})$ *is in* P.

Proof. Let Σ be a ranked signature, $G = (N, \Sigma, R, S) \in \mathit{Lin}\mathcal{P}^{comp,tree}_\Sigma$ a linear compressed tree pattern and $A = (Q, \Sigma, F, \Delta)$ an Nta. Given that the instance set of the linear pattern $\mathit{pat}(G)$ is regular, one could think of building an Nta that recognizes $\mathit{Inst}(\mathit{pat}(G))$, but since $\mathcal{P}(G)$ may be exponential in the size of G, this approach does not work in polynomial time.

Instead we evaluate the pattern G directly in the Σ-algebra Δ, while mapping context variables to the accessibility relation of Δ. So let $\mathit{acc}_\Delta \colon Q \to 2^Q$ be the function that maps every $q \in Q$ to the set of states accessible from state q with respect to Δ. We consider the well-typed assignment s that maps all

tree variables x in $fv(G)$ to $s(x) = Q$ and all context variables $X \in fv(G)$ to $s(X) = acc_\Delta$. The following then holds:

Claim 36. *$Inst(pat(G)) \cap L(A) \neq \emptyset$ if and only if $[\![pat(G)]\!]^{\Delta,\widehat{s}} \cap F \neq \emptyset$.*

Thanks to Claim 36, one can simply test $[\![pat(G)]\!]^{\Delta,\widehat{s}} \cap F \neq \emptyset$ in order to decide whether $Inst(pat(G)) \cap L(A) \neq \emptyset$. By Lemma 20, it takes polynomial time in the sizes of Δ, G and s to compute $[\![pat(G)]\!]^{\Delta,\widehat{s}}$. It follows that the problem $\textsc{Match}_\Sigma(Lin\mathcal{P}^{comp,tree}, \textsc{Nta})$ is in P. □

We next consider regular inclusion for linear tree patterns. Proposition 35 and the duality via complementation (Lemma 22) yield for Dtas that regular inclusion for linear patterns is in P too. So it remains to consider the case of regular inclusion for Ntas. By Lemma 23, this problem is Exp-hard even without context variables and without compression. Therefore regular inclusion for Ntas and (compressed) linear patterns with or without context variables is Exp-complete.

Conclusion

We have shown that regular matching and inclusion of ranked tree patterns with context variables against nondeterministic tree automata is Exp-complete with and without compression, while the problem is Pspace-complete for bottom-up deterministic tree automata. The complexity goes down to P for linear compressed tree patterns in 3 of 4 cases. The analogous results hold for unranked tree patterns with hedge variables, which is relevant to certain query answering on hyperstreams. Previous approaches were limited to hyperstreams containing words (compressed string patterns), while the present approach can deal with hyperstreams containing unranked data trees (compressed unranked tree patterns).

Acknowledgments

We are grateful to Sylvain Salvati for discussing the relationship of inhabitation and lambda-definability, for pointing out the additional difficulties when generalizing from context to unrestricted second-order functions, and for helping us to reduce the upper bounds from double exponential time to exponential time.

We also thank the reviewers of Information & Computation for their high quality feedback which helped us improve the presentation of this article.

This work was partially supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

[1] I. Boneva, J. Niehren, M. Sakho, Regular Matching and Inclusion on Compressed Tree Patterns with Context Variables, in: LATA 2019 - 13th International Conference on Language and Automata Theory and Applications, Saint Petersburg, Russia, 2019.
URL https://hal.inria.fr/hal-01811835

[2] D. Angluin, Finding patterns common to a set of strings., Journal of Computer and System Sciences 21 (1980) 46–62.

[3] A. Gascón, G. Godoy, M. Schmidt-Schauß, Context matching for compressed terms, in: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA, IEEE Computer Society, 2008, pp. 93–102. doi:10.1109/LICS.2008.17.
URL http://dx.doi.org/10.1109/LICS.2008.17

[4] W. Plandowski, Satisfiability of word equations with constants is in PSPACE, J. ACM 51 (3) (2004) 483–496. doi:10.1145/990308.990312.
URL https://doi.org/10.1145/990308.990312

[5] I. Boneva, J. Niehren, M. Sakho, Certain query answering on compressed string patterns: From streams to hyperstreams, in: Reachability Problems - 12th International Conference, RP 2018, Marseille, France, September 24-26, 2018, Proceedings, 2018, pp. 117–132. doi:10.1007/978-3-030-00250-3_9.
URL https://doi.org/10.1007/978-3-030-00250-3_9

[6] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, Available online since 1997: http://tata.gforge.inria.fr (Oct. 2007).

[7] M. Schmidt-Schauß, Linear pattern matching of compressed terms and polynomial rewriting, Mathematical Structures in Computer Science 28 (8) (2018) 1415–1450. doi:10.1017/S0960129518000208.
URL https://doi.org/10.1017/S0960129518000208

[8] A. Jez, Context unification is in PSPACE, in: J. Esparza, P. Fraigniaud, T. Husfeldt, E. Koutsoupias (Eds.), Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II, Vol. 8573 of Lecture Notes in Computer Science, Springer, 2014, pp. 244–255. doi:10.1007/978-3-662-43951-7_21.
URL https://doi.org/10.1007/978-3-662-43951-7_21

[9] M. Zaionc, Probabilistic approach to the lambda definability for fourth order types, Electr. Notes Theor. Comput. Sci. 140 (2005) 41–54. doi:10.1016/j.entcs.2005.06.025.
URL https://doi.org/10.1016/j.entcs.2005.06.025

[10] R. Loader, The undecidability of λ-definability, in: Z. M. Anderson C.A. (Ed.), Logic, Meaning and Computation, Vol. 305, Springer, 2001. doi:https://doi.org/10.1007/978-94-010-0526-5_15.

[11] T. Joly, Encoding of the halting problem into the monster type & applications, in: M. Hofmann (Ed.), Typed Lambda Calculi and Applications, 6th

International Conference, TLCA 2003, Valencia, Spain, June 10-12, 2003, Proceedings., Vol. 2701 of Lecture Notes in Computer Science, Springer, 2003, pp. 153–166. doi:10.1007/3-540-44904-3_11.
URL https://doi.org/10.1007/3-540-44904-3_11

[12] D. Kozen, Lower bounds for natural proof systems, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 254–266. doi:10.1109/SFCS.1977.16.
URL https://doi.org/10.1109/SFCS.1977.16

[13] H. Seidl, Deciding equivalence of finite tree automata, SIAM J. Comput. 19 (3) (1990) 424–437. doi:10.1137/0219027.
URL https://doi.org/10.1137/0219027

[14] S. Maneth, A. O. Pereira, H. Seidl, Transforming XML streams with references, in: C. S. Iliopoulos, S. J. Puglisi, E. Yilmaz (Eds.), String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings, Vol. 9309 of Lecture Notes in Computer Science, Springer, 2015, pp. 33–45. doi:10.1007/978-3-319-23826-5_4.
URL https://doi.org/10.1007/978-3-319-23826-5_4

[15] M. Lohrey, S. Maneth, M. Schmidt-Schauß, Parameter reduction and automata evaluation for grammar-compressed trees, J. Comput. Syst. Sci. 78 (5) (2012) 1651–1669. doi:10.1016/j.jcss.2012.03.003.
URL https://doi.org/10.1016/j.jcss.2012.03.003

[16] W. J. Savitch, Relationships between nondeterministic and deterministic tape complexities, Journal of Computer and System Sciences 4 (2) (1970) 177 – 192. doi:https://doi.org/10.1016/S0022-0000(70)80006-X.
URL http://www.sciencedirect.com/science/article/pii/S002200007080006X

Appendix A. Proofs for Section 3 (Inhabitation for Tree Automata)

Theorem 7 (Folklore)**.** *Tree inhabitation* $\textsc{Inhab}_\Sigma^{tree}(\textsc{Nta})$ *is* Exp*-complete, while its restriction to deterministic tree automata* $\textsc{Inhab}_\Sigma^{tree}(\textsc{Dta})$ *is in* P*.*

Proof. The upper bounds were shown in Proposition 6. The Exp lower bound for Ntas follows from a reduction from nonemptiness of intersection of a finite number of deterministic tree automata $\textsc{Inter}_\Sigma(\textsc{Dta})$, which is well known to be ExpTime-complete [13].

Let $A_1, \ldots, A_n$ be a sequence of Dtas with signature Σ. We want to know whether $\cap_{i=1}^n L(A_i) \neq \emptyset$. Suppose that $A_i = (Q_i, \Sigma, F_i, \Delta_i)$. Without loss of generality, we can assume that each of them has a single final state $F^i = \{q_f^i\}$ [2]. Let A be the disjoint union of all A_i, that is $A = (Q, \Sigma, F, \Delta)$ where $Q = \uplus_{i=1}^n Q_i$, $\Delta = \uplus_{i=1}^n \Delta_i$ and $F = \{q_f^1, \ldots, q_f^n\}$. Since all A_i are deterministic, it holds that (*) F is Δ-inhabited if and only if $\bigcap_{i=1}^n L(A_i) \neq \emptyset$. In order to see (*), let $t \in \mathcal{T}_\Sigma$ be a tree. It then holds that:

$$
\begin{aligned}
t \in \bigcap_{i=1}^n L(A_i) \quad &\text{iff} \quad \text{for all } i \in \{1, \ldots, n\} : \ q_f^i \in [\![t]\!]^{\Delta_i} \\
&\text{iff} \quad \text{for all } i \in \{1, \ldots, n\} : \ \{q_f^i\} = [\![t]\!]^{\Delta_i} \qquad (A_i \text{ is deterministic}) \\
&\text{iff} \quad \{q_f^1, \ldots, q_f^n\} = [\![t]\!]^{\Delta} \\
&\text{iff} \quad \{q_f^1, \ldots, q_f^n\} \text{ is } \Delta\text{-inhabited by } t.
\end{aligned}
$$

The property (*) shows that $\textsc{Inter}_\Sigma(\textsc{Dta})$ can be reduced to $\textsc{Inhab}_\Sigma^{tree}(\textsc{Nta})$ in polynomial time, so $\textsc{Inhab}_\Sigma^{tree}(\textsc{Nta})$ is Exp hard. □

Lemma 9. *For any context* $C \in \mathcal{C}_\Sigma$ *and* Nta $A = (Q, \Sigma, F, \Delta)$ *the* Δ*-inhabited value* $[\![C]\!]^\Delta$ *is a union homomorphism on* 2^Q*.*

Proof. Any context $C \in \mathcal{C}_\Sigma$ has the form $\lambda x.t$ such that x occurs exactly once in t. The proof is by induction on the structure of t.

[2] Otherwise, we fix a nonconstant $g \in \Sigma \setminus \Sigma^{(0)}$ and a constant $a \in \Sigma^{(0)}$. We then compute automata A_i' with $L(A_i') = g(L(A_i), a, \ldots, a)$. These can be constructed in P from A_i such that they have a unique final state. Furthermore, $\cap_{i=1}^n L(A_i) \neq \emptyset$ if and only if $\cap_{i=1}^n L(A_i') \neq \emptyset$.

- Case $t = x$. We then have that $[\![C]\!]^{\Delta}(Q') = [\![\lambda x.x]\!]^{\Delta}(Q') = [\![Q']\!]^{\Delta} = Q'$ for all $Q' \subseteq Q$. In particular $[\![C]\!]^{\Delta}(\emptyset) = \emptyset$. Furthermore for any two subsets $Q', Q'' \subseteq Q$, it holds that $[\![C]\!]^{\Delta}(Q' \cup Q'') = Q' \cup Q'' = [\![C]\!]^{\Delta}(Q') \cup [\![C]\!]^{\Delta}(Q'')$. Thus $[\![C]\!]^{\Delta}$ is a union homomorphism.

- Case $t = f(t_1, \ldots, t_n)$ and x occurs exactly once in t, say in t_k but not elsewhere.

  Let $S_k = [\![\lambda x.t_k]\!]^{\Delta}$ and $Q_i = [\![t_i]\!]^{\Delta}$ for all $i \neq k$. Clearly S_k is Δ-inhabited. We then have by the induction hypothesis that S_k is a union homomorphism. Furthermore, we have that for all $Q' \subseteq Q$, $[\![C]\!]^{\Delta}(Q') = [\![\lambda x.f(t_1, \ldots, t_n)]\!]^{\Delta}(Q') = [\![f(t_1, \ldots, \lambda x.t_k, \ldots, t_n)]\!]^{\Delta}(Q')$. By definition of algebra evaluation, we have $[\![f(t_1, \ldots, \lambda x.t_k, \ldots, t_n)]\!]^{\Delta}(Q') = \{q \mid \exists q_1 \in Q_1, \ldots, q_k \in S_k(Q'), \ldots, \exists q_n \in Q_n.\ f(q_1, \ldots, q_n) \to q \text{ in } \Delta\}$. This implies that for any two subsets $Q', Q'' \subseteq Q$, $[\![C]\!]^{\Delta}(Q' \cup Q'') = \{q \mid \exists q_1 \in Q_1, \ldots, q_k \in S_k(Q') \cup S_k(Q''), \ldots \exists q_n \in Q_n.\ f(q_1, \ldots, q_n) \to q \text{ in } \Delta\} = [\![C]\!]^{\Delta}(Q') \cup [\![C]\!]^{\Delta}(Q'')$. In particular, $[\![C]\!]^{\Delta}(\emptyset) = \emptyset$. So $[\![C]\!]^{\Delta}$ is a union homomorphism.

□

Lemma 15. *The problem* $\textsc{Inhab}^{context}(\textsc{Dta})$ *can be reduced in polynomial time to its restriction where the input function* $s : Q \to 2^Q$ *always maps to singletons.*

Proof. If there exists $q \in Q$ such that $s(q)$ contains more than one element then s cannot be inhabited for any Dta. It remains to remove cases where $s(q) = \emptyset$ for some $q \in Q$. The main idea to deal with empty sets is to complete A to A' by adding a sink state q_{sink} and to replace function s by s' such that $s'(q) = s(q)$ if $s(q) \neq \emptyset$ and $s'(q) = \{q_{sink}\}$ otherwise. Inhabitation of s with respect to A is then equivalent to inhabitation of s' with respect to A'. However, this construction may take exponential time in the maximal arity of function symbols of A with is not fixed for the uniform problem. This problem can be circumvented by a trick, permitting to complete A only partially.

Here is how it works. We consider a DTA $A = (Q, \Sigma, F, \Delta)$ and a function $s : Q \to 2^Q$. We construct another DTA $A' = (Q', \Sigma', F', \Delta')$ and a function $s' : Q' \to 2^{Q'} \setminus \emptyset$ such that $\hat{s}$ is Δ-inhabited if and only if $\hat{s'}$ is Δ'-inhabited.

The first idea would be to set A' as the completion of A. We then have $\Sigma' = \Sigma$ and $Q' = Q \cup \{q_{sink}\}$ where q_{sink} is some fresh sink state. Furthermore, the set of rules Δ' subsumes Δ and all the rules $f(q_1, \ldots, q_n) \to q_{sink}$ with $q_1, \ldots, q_n \in Q'$ for which $f(q_1, \ldots, q_n)$ is not a left-hand side of any rule in Δ. The function s' is defined such that $s'(q) = s(q)$ if $s(q) \neq \emptyset$ and $s'(q) = \{q_{sink}\}$ otherwise. One can then see for any context $C \in \mathcal{C}_\Sigma$ that $\hat{s}$ is Δ-inhabited by C if and only if $\hat{s'}$ is Δ'-inhabited by C. The size of Δ' is in $O(|\Delta| + |\Sigma||Q|^n)$ where n is the maximal arity of function symbols in Σ. Unfortunately, the maximal arity is not fixed in the uniform version since Σ is part of the input. Therefore, this reduction requires exponential space in the worst case, while polynomial time was claimed.

The second idea is to perform some kind of partial completion, so that only polynomially many rules need to be added. For this, we define the signature $\Sigma' = \Sigma \cup \{g\}$ where g is a fresh monadic function symbol. For any context $C \in \mathcal{C}_\Sigma$ we define a context in $\mathcal{C}_{\Sigma'}$ by $C' = \lambda x.C@g(x)$. The state set of A' remains $Q' = Q \cup \{q_{sink}\}$ where q_{sink} is some fresh state as before. The set of rules Δ' extends Δ by the following rules for all $q \in Q$:

$$h(q) \to \begin{cases} q_{sink} & \text{if } s(q) = \emptyset \\ q & \text{else} \end{cases}$$

Furthermore, we add the following rule for all rules $f(q_1, \ldots, q_n) \to q'$ of Δ and all $1 \leq i \leq n$:

$$f(q_1 \ldots, q_{i-1}, q_{sink}, q_{i+1}, \ldots, q_n) \to q_{sink}$$

It can then be shown for any context $C \in \mathcal{C}_\Sigma$, that $\hat{s}$ is Δ-inhabited by C if and only if $\hat{s'}$ is Δ'-inhabited by C'. Now the construction of A' is in time $O(|A|^2 + |s|)$ which is polynomial even if the maximal arity of function symbols in Σ is not bounded. □

Proposition 16. INHAB^{context}(DTA) *is in* PSPACE.

Proof. Let Σ be a ranked signature, $A = (Q, \Sigma, F, \Delta)$ a DTA where $Q = \{q_1, \ldots, q_n\}$ and all the states are accessible, $s\colon Q \to 2^Q$ a function and x a fresh constant not in Σ. We assume w.l.o.g. that $s(q_i) \neq \emptyset$ for all $1 \leq i \leq n$ (see Lemma 15). If $|s(q_i)| > 1$ for some $1 \leq i \leq n$, then $\hat{s}$ is not Δ-inhabited, given that A is deterministic. The following lines consider the case where all the images by s are singletons. First we reduce the inhabitation of $\hat{s}$ to the nonemptiness of the intersection of $n+1$ DTAs $A_1, \ldots, A_{n+1}$. In a second step, we reduce the nonemptiness of the intersection of $A_1, \ldots, A_{n+1}$ to the nonemptiness of the intersection of n DFAs $A'_1, \ldots, W_i$.

We write $\Sigma_x = \Sigma \cup \{x\}$. For any $i \in \{1, \ldots, n\}$, let $A_i = (Q, \Sigma_x, s(q_i), \Delta_i)$ be the tree automaton on Σ_x having the same states as A, whose set of final states is $s(q_i)$, and whose transition relation is $\Delta_i = \Delta \cup \{x \to q_i\}$. We also write A_{n+1} to denote the simple DTA that accepts all trees $t \in \mathcal{T}_{\Sigma_x}$ having exactly one occurrence of x. We first show that

Claim 37. *There exists a context $\lambda x.p \in \mathcal{C}_\Sigma$ such that $[\![\lambda x.p]\!]^\Delta = \hat{s}$ if and only if $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$.*

Proof. On one hand, if there is a context $\lambda x.p \in \mathcal{C}_\Sigma$ such that $[\![\lambda x.p]\!]^\Delta = \hat{s}$, then by Proposition 11 we have $[\![p[x/\{q_i\}]]\!]^\Delta = s(q_i)$ for any $1 \leq i \leq n$. This implies that $p \in L(A_i)$ for any $1 \leq i \leq n$, and since $\lambda x.p$ is a context, p contains exactly one occurrence of x and thus belongs to $L(A_{n+1})$. Hence $\bigcap_{i=1}^{n+1} L(A_i) \supseteq \{p\} \neq \emptyset$. On the other hand, assume $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$ and let $p \in \bigcap_{i=1}^{n+1} L(A_i)$. Given that $p \in L(A_{n+1})$, it contains exactly one occurrence of x. Furthermore, since the automata A_j are all deterministic with unique final states $s(q_j)$, and $p \in \bigcap_{j=1}^{n} L(A_j)$, we have $[\![p]\!]^{\Delta_i} = s(q_i)$ for $1 \leq i \leq n$. This implies that $[\![p[x/\{q_i\}]]\!]^\Delta = s(q_i)$ for $1 \leq i \leq n$, and thus $[\![\lambda x.p]\!]^\Delta = \hat{s}$. □

According to Claim 37, deciding whether or not $\hat{s}$ is Δ-inhabited is equivalent to determining if the DTAs A_i have an nonempty intersection. Next we show a PSPACE algorithm to decide $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$, by reduction to INTER(DFA).

Let Σ_Q be the alphabet that contains the symbol $\underline{x}$, and for any rule $f(q'_1, \ldots, q'_{k-1}, q'_k, q'_{k+1}, \ldots, q'_m) \to q''$ in Δ and any $1 \leq k \leq m$, Σ_Q contains the symbol $\underline{f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m)}$, where m is the arity of f. Formally,

$$\Sigma_Q = \{\underline{x}\} \cup \left\{ \begin{array}{l} \underline{f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m)} \mid m \text{ is an arity in } \Sigma, \\ f \in \Sigma^{(m)}, 1 \leq k \leq m \text{ and } \exists q'_k, q'_{m+1} \in Q. \\ f(q'_1, \ldots, q'_{k-1}, q'_k, q'_{k+1}, \ldots, q'_m) \to q'_{m+1} \in \Delta \end{array} \right\}$$

The notation introduced for the elements of Σ_Q allows us to distinguish them from the trees in $\mathcal{T}_{\Sigma_x \cup Q}$. This is because the elements of Σ_Q are considered as atomic symbols. Now let the alphabet $\mathfrak{S} = \Sigma_Q \cup \{\bot\}$. For some $i \in \{1, \ldots, n\}$ and a tree $t \in \mathcal{T}_{\Sigma_x} \cap L(A_{n+1})$ over Σ_x containing exactly one occurrence of x, we define inductively the *run path* $rp_i(t)$ of t with respect to the DTA A_i as a word over $\mathfrak{S}$ such that:

- if $t = x$, then $rp_i(t) = \underline{x}$
- if $t = f(t_1, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_m)$ for some arity $m \geq 0$, symbol $f \in \Sigma^{(m)}$, integer $k \in \{1, \ldots, m\}$ so that t_k contains the only occurrence of x in t, then

$$rp_i(t) = \left\{ \begin{array}{l} rp_i(t_k)\underline{f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m)} \text{ if } \{q'_j\} = [\![t_j]\!]^{\Delta_i} \text{ for all } j \neq k, \\ 1 \leq j \leq m \text{ and } f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m) \in \Sigma_Q \\ \\ rp_i(t_k)\bot \text{ otherwise} \end{array} \right.$$

Example 3. *For instance, consider that* $\Sigma = \{f^{(2)}, b^{(1)}, c^{(1)}, a^{(0)}\}$ *and the transition relation* Δ_1 *of the* DTA A_1 *is such that* $\Delta_1 = \{x \to q_1, a \to q_2, b(q_1) \to q_3, f(q_2, q_3) \to q_4\} \cup \Delta'$ *where* Δ' *consists of the remaining rules that make* Δ_1 *complete. Then the run path of the tree* $f(a, b(x))$ *with respect to* A_1 *is* $\underline{x}\ \underline{b(\star)}\ \underline{f(q_2, \star)}$ *as illustrated in Figure A.16. On the other hand, the run path of* $f(c(a), b(x))$ *with respect to* A_1 *is* $\underline{x}\ \underline{b(\star)}\ \bot$*, as the subtree* $c(a)$ *cannot be evaluated.*

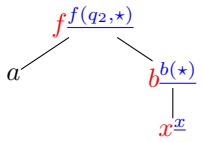


Figure A.16: Run path (in blue) of the tree $f(a, b(x))$ with respect to A_1.

Claim 38. *Let $t \in \mathcal{T}_{\Sigma_x} \cap L(A_{n+1})$ be a tree over Σ_x containing exactly one occurrence of x. Then $rp_i(t) = rp_j(t)$ for all $i, j \in \{1, \ldots, n\}$.*

Proof. Let $i, j \in \{1, \ldots, n\}$. The proof is by induction on the structure of t.

Case $t = x$**.** Then $rp_i(t) = rp_j(t) = \underline{x}$ by definition.

Case $t = f(t_1, \ldots, t_k, \ldots, t_m)$ for some arity m, symbol $f \in \Sigma^{(m)}$, integer $k \in \{1, \ldots, m\}$ so that t_k contains the only occurrence of x in t. By definition,

- $rp_i(t) = rp_i(t_k)a_i$
- and $rp_j(t) = rp_j(t_k)a_j$

where a_i and a_j are such that $a_i \in \{\underline{f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m)}, \bot\}$, $a_j \in \{\underline{f(q''_1, \ldots, q''_{k-1}, \star, q''_{k+1}, \ldots, q''_m)}, \bot\}$ for states $q'_l \in Q$, $q''_l \in Q$, $l \neq k$ and $l \in \{1, \ldots, m\}$. By the induction hypothesis, $rp_i(t_k) = rp_j(t_k)$. Furthermore, we show that $a_i = a_j$. Let $l \in \{1, \ldots, m\}$ be different from k. Since t_l contains no occurrence of x, we have $[\![t_l]\!]^{\Delta_i} = [\![t_l]\!]^{\Delta} = [\![t_l]\!]^{\Delta_j}$. Two cases may occur, depending on the run of A on t_l. Either the run blocks, that is $[\![t_l]\!]^{\Delta} = \emptyset$, or it doesn't, implying that $[\![t_l]\!]^{\Delta} = \{q'_l\}$ for some state $q'_l \in Q$. Now if for all $l \in \{1, \ldots, m\}$ different from k, $[\![t_l]\!]^{\Delta}$ equals some singleton $\{q'_l\}$, then by definition $a_i = \underline{f(q'_1, \ldots, q'_{k-1}, \star, q'_{k+1}, \ldots, q'_m)} = \underline{f(q''_1, \ldots, q''_{k-1}, \star, q''_{k+1}, \ldots, q''_m)} = a_j$. And if there is some $l \in \{1, \ldots, m\}$ different from k such that $[\![t_l]\!]^{\Delta} = \emptyset$, then by definition $a_i = \bot = a_j$. In both cases, $a_i = a_j$. Thus $rp_i(t) = rp_j(t)$.

□

Next we build DFAs that accept run paths. Let q_0 and $q_\perp$ be fresh states, and note $Q_{\text{DFA}} = Q \cup \{q_0, q_\perp\}$. For all $1 \leq i \leq n$, we build the DFA W_i having Q_{DFA} as its set of states, $\mathfrak{S}$ as its alphabet, $\{q_0\}$ as its set of initial states, $s(q_i)$ as its set of final states and δ_i as its transition function, so that

- $\delta_i(q_0, \underline{x}) = q_i$ (1)
- for all $f(q'_1, \ldots, q'_m) \to q'_{m+1} \in \Delta_i$ where $f \in \Sigma^{(m)}$ for some arity m, we have $\delta_i(q'_1, \underline{f(\star, q'_2, \ldots, q'_m)}) = q'_{m+1}$, $\delta_i(q'_2, \underline{f(q'_1, \star, q'_3, \ldots, q'_m)}) = q'_{m+1}$, $\ldots, \delta_i(q'_m, \underline{f(q'_1, \ldots, q'_{m-1}, \star)}) = q'_{m+1}$ (2)
- for all $q \in Q_{\text{DFA}}$, $\delta_i(q, \perp) = q_\perp$ (3)
- for all state $q \in Q_{\text{DFA}}$ and symbol $\underline{f(q'_1, \ldots, q'_{i-1}, \star, q'_{i+1}, \ldots, q'_m)} \in \Sigma_Q$ where $m \geq 0$ and $1 \leq i \leq m$, if no rule in Δ_i having $f(q'_1, \ldots, q'_{i-1}, q, q'_{i+1}, q'_m)$ as its left-hand side exists, then $\delta_i(q, \underline{f(q'_1, \ldots, q'_{i-1}, \star, q'_{i+1}, \ldots, q'_m)}) = q_\perp$ (4).

Note that any DFA W_i has a size that is polynomial in $|A|$. Now let $p \in \mathcal{T}_{\Sigma_x} \cap L(A_{n+1})$ be a tree over Σ_x containing exactly one occurrence of x.

Claim 39. *For all $i \in \{1, \ldots, n\}$ and state $q \in Q$, $[\![p]\!]^{\Delta_i} = \{q\}$ if and only if $rp_i(p)$ is evaluated to q by the* DFA *W_i.*

Proof. Let $i \in \{1, \ldots, n\}$ and $q \in Q$. The proof is by induction on the structure of p. The backward direction is shown by contraposition.

Case p=x. Then we have $rp_i(p) = \underline{x}$. First let's assume that $[\![p]\!]^{\Delta_i} = \{q\}$. So we have $q = q_i$, since $[\![p]\!]^{\Delta_i} = [\![x]\!]^{\Delta_i} = \{q_i\}$. Furthermore, W_i in its initial state q_0 reads $\underline{x}$ and enters by (1) in state $q_i = q$. Thus $rp_i(p)$ is evaluated to q by W_i.

For the backwards direction, assume that $[\![p]\!]^{\Delta_i} \neq \{q\}$. This implies that $[\![x]\!]^{\Delta_i} = \{q_i\} \neq \{q\}$, that is $q_i \neq q$. On the other hand, starting from q_0, W_i evaluates $\underline{x}$ to $q_i \neq q$ according to (1).

Case $p = f(p_1, \ldots, p_k)$ where $f \in \Sigma^{(k)}$ and $p_1, \ldots p_k \in \mathcal{T}_{\Sigma_x}$. Then there exists a unique $l \in \{1, \ldots, k\}$ such that p_l contains exactly one occurrence of x, and for all $j \in \{1, \ldots, k\}$, if $j \neq l$ then $p_j \in \mathcal{T}_\Sigma$ – that is p_j contains only symbols in Σ.

First assume that $[\![p]\!]^{\Delta_i} = \{q\}$. Then there exist states $\gamma_1, \ldots, \gamma_k$ s.t. for all $1 \leq j \leq k$, $[\![p_j]\!]^{\Delta_i} = \{\gamma_j\}$. By the induction hypothesis, $[\![p_l]\!]^{\Delta_i} = \{\gamma_l\}$ if and only if $rp_i(p_l)$ is evaluated to γ_l by W_i. By definition $rp_i(p) = rp_i(p_l)\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$. We also have the equalities $[\![p]\!]^{\Delta_i} = [\![f(p_1, \ldots, p_k)]\!]^{\Delta_i} = \{q\}$. So the rule $f(\gamma_1, \ldots, \gamma_k) \to q$ exists in Δ_i. By (2), we also have $\delta_i(\gamma_l, \underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}) = q$. So the DFA W_i in state q_0 first reads the word $rp_i(p_l)$ to get in state γ_l, before finally entering state q after having read $\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$. So $rp_i(p)$ can be evaluated to q by W_i.

For the backwards direction, assume that $[\![p]\!]^{\Delta_i} \neq \{q\}$. Two cases may occur:

Case $[\![p]\!]^{\Delta_i} = \emptyset$. Then

- either $[\![p_j]\!]^{\Delta_i} = \emptyset$ for some $j \in \{1, \ldots, k\}$ (i),
- or there exist states $\gamma_1, \ldots, \gamma_k$ s.t. for all $j \in \{1, \ldots, k\}$, $[\![p_j]\!]^{\Delta_i} = \{\gamma_j\}$, but there is no rule in Δ_i having $f(\gamma_1, \ldots, \gamma_k)$ as its left-hand side (ii).

In (i), if $j \neq l$ we have by definition that $rp_i(p) = rp_i(p_l)\bot$. According to rule (3), whatever the state in which the DFA W_i is after having read $rp_i(p_l)$, W_i goes to state $q_\bot$ when reading $\bot$. And since $q_\bot \neq q$, the claim holds. On the other hand, if $j = l$ and $[\![p_{j'}]\!]^{\Delta_i} = \{\gamma_{j'}\}$ for all $j' \in \{1, \ldots, k\}$ different from j, then $rp_i(p) = rp_i(p_l)\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$. By the induction hypothesis, W_i evaluates $rp_i(p_l)$ to a state that is not in Q. The only states in Q_{DFA} that are not in Q are q_0 and $q_\bot$, and given that $rp_i(p_l) \neq \varepsilon$ and q_0 has no looping transition – W_i can't stay in state

q_0 after having read p_l –, it follows that the only possible state to which $rp_i(p_l)$ has been evaluated by W_i is $q_\perp$. All the transitions in δ_i that leave $q_\perp$ end up in $q_\perp$ by the rule (4). Thus W_i evaluates $rp_i(p)$ in state $q_\perp \neq q$, and the claim holds.

In (ii), $rp_i(p) = rp_i(p_l)\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$. By the induction hypothesis, W_i evaluates $rp_i(p_l)$ to state γ_l. But since no rule $f(\gamma_1, \ldots, \gamma_k) \to q'$ exists in Δ_i, W_i in state γ_l – after having read $rp_i(p_l)$ – goes to state $q_\perp$ after reading $\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$, according to rule (4). Thus the claim holds.

Case $[\![p]\!]^{\Delta_i} = \{q'\} \neq \{q\}$. Then there exist states $\gamma_1, \ldots, \gamma_k$ s.t. for all $1 \leq j \leq k$, $[\![p_j]\!]^{\Delta_i} = \{\gamma_j\}$. Moreover, there is a rule $f(\gamma_1, \ldots, \gamma_k) \to q' \in \Delta_i$, but no rule $f(\gamma_1, \ldots, \gamma_k) \to q$ in Δ_i. Thus by (2), we have that $\delta(\gamma_l, \underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}) = q'$. This implies that the DFA W_i in state q_0, first reads $rp_i(p_l)$ to get in state γ_l, then reads the symbol $\underline{f(\gamma_1, \ldots, \gamma_{l-1}, \star, \gamma_{l+1}, \ldots, \gamma_k)}$ to enter state $q' \neq q$. Thus the claim holds.

□

We next state:

Claim 40. $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$ *if and only if* $\bigcap_{i=1}^{n} L(W_i) \neq \emptyset$.

Proof. Let $p \in \Sigma_x \cap L(A_{n+1})$ be a tree containing exactly one occurrence of x. By Claim 39, for all $i \in \{1, \ldots, n\}$, $[\![p]\!]^{\Delta_i} = s(q_i)$ if and only if $rp_i(p)$ is evaluated to the single element of $s(q_i)$ by W_i. So $p \in L(A_i)$ if and only if $rp_i(p) \in L(W_i)$ for $1 \leq i \leq n$. Claim 38 has established that $rp_j(p) = rp_k(p)$ for all $j, k \in \{1, \ldots, n\}$. It then follows that $p \in \bigcap_{i=1}^{n} L(A_i)$ if and only if $rp_1(p) = \ldots = rp_n(p) \in \bigcap_{i=1}^{n} L(W_i)$. So $\bigcap_{i=1}^{n+1} L(A_i) \neq \emptyset$ if and only if $\bigcap_{i=1}^{n} L(W_i) \neq \emptyset$. □

It follows from Claim 37 and Claim 40 that $\hat{s}$ is Δ-inhabited if and only if $\bigcap_{i=1}^{n} L(W_i) \neq \emptyset$. Thus INHAB^{context}(DTA) is reducible in polynomial time to INTER(DFA). Hence INHAB^{context}(DTA) is in PSPACE.

□

Appendix B. Proofs for Section 7 (Adding Regular Constraints)

Claim 29. *There exists a bijection* $\varphi : Inst^{\rho}(pat(G)) \to Inst(pat(G'')) \cap L(B') \cap L(C')$ *such that for all* $t \in Inst^{\rho}(pat(G))$, $t \in L(A)$ *if and only if* $\varphi(t) \in L(A'')$.

Proof. We construct φ as the function that transforms an element of $t \in Inst^{\rho}(pat(G))$ satisfying the constraints in ρ to an element of $t' \in Inst(pat(G''))$ in which all the constraints in ρ are satisfied – modulo the change of signature from Σ to Σ' –, thus implying that $t' \in L(B') \cap L(C')$. We first introduce a function ann_{ν} for all variable $\nu \in \mathcal{V}$, such that for all tree variable x, n-ary function symbol f and trees $t_1, \ldots, t_n$ where $n \geq 0$:

$$ann_{\nu}(x) = hole_{\nu}(x)$$
$$ann_{\nu}(f(t_1, \ldots, t_n)) = (f, \nu)(ann_{\nu}(t_1), \ldots, ann_{\nu}(t_n))$$

Then we define φ so that for all well-typed substitution $\mu : fv(G) \to \mathcal{P}^{gr}_{\Sigma}$, the image of the grounding $p = norm_{\beta}(\mu(pat(G)))$ is such that

- every subterm of p obtained by instantiating some tree variable x of G is replaced by $root_x(ann_x(\mu(x)))$
- every subterm of p obtained by instantiating some context variable X is replaced by $root_X(ann_X(t))$, where $\mu(X) = \lambda x_{\rho}.t$

For example, if we set $\Sigma = \{f^{(2)}, a^{(0)}, b^{(0)}\}$, $G = (\{z, x, X\}, \Sigma, \{z \to f(x, X@b)\}, z)$, $\mu(x) = a$ and $\mu(X) = \lambda x_{\rho}.f(a, x_{\rho})$, then the pattern in Figure B.17 gives the value of $\varphi(norm_{\beta}(\mu(pat(G))))$.

Furthermore, for all $t \in Inst^{\rho}(pat(G))$, $t \in L(A)$ if and only if $\varphi(t) \in L(A'')$.

□

Claim 30. cMatch$(\mathcal{G}, \mathcal{A})$ *is reducible in polynomial time to* Match$(\mathcal{G}, \mathcal{A})$.

Proof. Let $t \in Inst^{\rho}(pat(G))$ be a constrained instance of G by ρ. By Claim 29, $t \in L(A)$ iff $\varphi(t) \in L(A'')$. Given that $\varphi(t) \in Inst(pat(G'')) \cap L(B') \cap L(C')$,

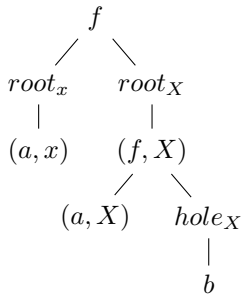


Figure B.17: Example of image value by φ

it follows that $Inst^{\rho}(pat(G)) \cap L(A) \neq \emptyset$ iff $Inst(pat(G'')) \cap (L(B') \cap L(C') \cap L(A'')) \neq \emptyset$. One can compute a product automaton D in polynomial time from A'' and B' and C' so that $L(D) = L(A'') \cap L(B') \cap L(C')$. Furthermore, if A'' is deterministic, then D is also deterministic – knowing that B' and C' are deterministic. Thus $Inst^{\rho}(pat(G)) \cap L(A) \neq \emptyset$ iff $Inst(pat(G'')) \cap L(D) \neq \emptyset$, hence cMatch$(\mathcal{G}, \mathcal{A})$ is reducible in polynomial time to Match$(\mathcal{G}, \mathcal{A})$. □

Claim 31. cIncl$(\mathcal{G}, \mathcal{A})$ *is reducible in polynomial time to* Incl$(\mathcal{G}, \mathcal{A})$.

Proof. Let $t \in Inst^{\rho}(pat(G))$ be a constrained instance of G by ρ. By Claim 29, $t \in L(A)$ iff $\varphi(t) \in L(A'')$. Given that $\varphi(t) \in Inst(pat(G')) \cap L(B') \cap L(C')$, it follows that $Inst^{\rho}(pat(G)) \subseteq L(A)$ iff $Inst(pat(G'')) \cap L(B') \cap L(C') \subseteq L(A'')$. Let the product automaton $B' \times C'$ recognizing the language $L(B') \cap L(C')$. Then $Inst^{\rho}(pat(G)) \subseteq L(A)$ iff $Inst(pat(G'')) \cap L(B' \times C') \subseteq L(A'')$, that is $Inst(pat(G'')) \subseteq L(A'') \cup L(\overline{B' \times C'})$ where $\overline{B' \times C'}$ is the automaton recognizing the complement of $L(B' \times C')$. The Dta $B' \times C'$ can be complemented in linear time to obtain $\overline{B' \times C'}$, since it is deterministic. Moreover a product automaton D' recognizing $L(A'') \cup L(\overline{B' \times C'})$ can be built in polynomial time from A'' and $B' \times C'$, so that D' is deterministic if A'' is deterministic. Thus $Inst^{\rho}(pat(G)) \subseteq L(A)$ iff $Inst(pat(G')) \subseteq L(D')$, hence cIncl$(\mathcal{G}, \mathcal{A})$ is reducible in polynomial time to Incl$(\mathcal{G}, \mathcal{A})$. □

Appendix C. Proofs for Section 8 (Encoding Patterns for Unranked Trees)

Lemma 32. *$norm_\beta(\langle Inst(H)\rangle^{tree}) = Inst^{unr}(\langle H\rangle^{tree})$ for any $H \in \mathcal{P}^u_\Gamma$.*

Proof. Let $H \in \mathcal{P}^u_\Gamma$ be an unranked tree pattern. The proof is by induction on the structure of H.

Case $H = a(\varepsilon)$ where $a \in \Gamma$. Then the following equalities $Inst(H) = \{a(\varepsilon)\}$ and $\langle Inst(H)\rangle^{tree} = \{\lambda y.a((\lambda y.y)@\#, y)@\#\}$ hold. This implies that $norm_\beta(\langle Inst(H)\rangle^{tree}) = \{a(\#, \#)\} = Inst^{unr}(\langle H\rangle^{tree})$, since H contains no variable to instantiate.

Case $H = Y \in \mathcal{V}^u$. Then $Inst(H) = \{a(H') \mid a \in \Gamma \text{ and } H' \in \mathcal{H}_\Gamma\}$ and $\langle Inst(H)\rangle^{tree} = \{(\lambda y.a(\langle H'\rangle^{tree}, y))@\# \mid a \in \Gamma \text{ and } H' \in \mathcal{H}_\Gamma\}$. This implies that $norm_\beta(\langle Inst(H)\rangle^{tree}) = \{a(norm_\beta(\langle H'\rangle^{tree}), \#) \mid a \in \Gamma \text{ and } H' \in \mathcal{H}_\Gamma\} = Inst^{unr}(\langle H\rangle^{tree})$ since no unranked tree $a(H') \in Inst(H)$ contains a variable to instantiate.

Case $H = b(H')$ where $b \in \Gamma$ and $H' \in \mathcal{P}^h_\Gamma$. Then $Inst(H) = \{b(H'') \mid H'' \in Inst(H')\}$ and $\langle Inst(H)\rangle^{tree} = \{(\lambda y.b(\langle H''\rangle^{tree}, y))@\# \mid H'' \in Inst(H')\}$. So $norm_\beta(\langle Inst(H)\rangle^{tree}) = \{b(norm_\beta(\langle H''\rangle^{tree}), \#) \mid H'' \in Inst(H')\}$. By the induction hypothesis, $norm_\beta(\langle Inst(H')\rangle^{tree}) = Inst^{unr}(\langle H'\rangle^{tree})$, which implies that $norm_\beta(\langle Inst(H)\rangle^{tree}) = \{b(t, \#) \mid t \in Inst^{unr}(\langle H'\rangle^{tree})\} = Inst^{unr}(\langle H\rangle^{tree})$.

□

Appendix D. Proofs for Section 9 (Linearity Restriction)

Claim 36. *$Inst(pat(G)) \cap L(A) \neq \emptyset$ if and only if $[\![pat(G)]\!]^{\Delta,\widehat{s}} \cap F \neq \emptyset$.*

Proof. For the forward direction, assume $Inst(pat(G)) \cap L(A) \neq \emptyset$. According to Lemma 25, there exists a well-typed assignment $\sigma : fv(G) \to [\![Val_\Sigma]\!]^\Delta$ such that $[\![pat(G)]\!]^{\Delta,\sigma} \cap F \neq \emptyset$. For all tree variable $x \in fv(G)$ (resp. context variable

$X \in fv(G)$), the construction of s guarantees that $\sigma(x) \subseteq s(x) = Q$ (resp. for all $q \in Q$, $\sigma(X)(q) \subseteq s(X)(q) = acc_\Delta(q)$). This implies that $[\![pat(G)]\!]^{\Delta,\hat{s}} \cap F \neq \emptyset$ too.

For the inverse direction, let $p_S \in \mathcal{P}_\Sigma^{tree}$ be such that $R(S) = p_S$ in G and assume $[\![pat(G)]\!]^{\Delta,\widehat{\sigma}} \cap F \neq \emptyset$. We prove the property by induction on the structure of p_S.

- Case $p_S = x \in \mathcal{V}$. Then $[\![pat(G)]\!]^{\Delta,\widehat{s}} = s(x) = Q$. Since A is reduced and that all the states of the NTA are accessible, it holds that for all $q \in [\![pat(G)]\!]^{\Delta,\widehat{s}}$ there exists a tree $t \in \mathcal{T}_\Sigma$ such that $q \in [\![t]\!]^\Delta$. Let $q_f \in [\![pat(G)]\!]^{\Delta,\widehat{s}} \cap F$. There exists a tree $t_f \in L(A)$ such that $q_f \in [\![t_f]\!]^\Delta$, hence $t_f \in Inst(pat(G)) \cap L(A)$.

- Case $p_S = X@x$. We have $[\![pat(G)]\!]^{\Delta,\widehat{s}} = \widehat{s(X)}(s(x))$. Let the state $q_f \in \widehat{s(X)}(s(x)) \cap F$ be in the intersection of $[\![pat(G)]\!]^{\Delta,\widehat{s}}$ and F. Since $s(X) = acc_\Delta$, there is a state $q_r \in Q$ such that $q_f \in acc_\Delta(q_r)$, and thus a context $\lambda x.p_f \in \mathcal{C}_\Sigma$ such that $q_f \in [\![\lambda x.p_f]\!]^\Delta(\{q_r\})$. Furthermore, $q_r \in s(x) = Q$ is an accessible state of A, and so there is a tree t_r such that $q_r \in [\![t_r]\!]^\Delta$. Notice that $q_f \in [\![(\lambda x.p_f)@t_r]\!]^\Delta = [\![\lambda x.p_f]\!]^\Delta(\{q_r\})$, and thus $(\lambda x.p_f)@t_r \in Inst(pat(G)) \cap L(A)$.

- The cases $p_S = t$ and $p_S = X@t$ where $t \in \mathcal{T}_\Sigma$ and $X \in \mathcal{V}^{context}$ are respectively special instances of the first and second cases.

- Case $p_S = f(S_1, \ldots, S_n)$ where $f \in \Sigma^{(n)}$, $S_1, \ldots, S_n \in N \setminus fv(G)$ are starting symbols for some linear compressed tree patterns $G_1, \ldots, G_n \in Lin\mathcal{P}^{comp,tree}$ and for all different $i, j \in \{1, \ldots, n\}$, $fv(G_i) \cap fv(G_j) \neq \emptyset$. Here we have assumed without loss of generality that any compressed tree pattern is built only from smaller compressed tree patterns. Thus if there were some constant symbol or free variable v occurring in p_S, one could just create a new compressed tree pattern G' from G where the occurrences of v in S are replaced by a new nonterminal S_v, and with the additional rule $S_v \to v$. But for the sake of simplicity, we sup-

pose that G is already in the form we want it to be. Thus every S_i can be considered as the start symbol of the compressed tree pattern G_i. We have that $[\![pat(G)]\!]^{\Delta,\widehat{s}} = [\![f(pat(G_1),\ldots,pat(G_n))]\!]^{\Delta,\widehat{s}} = \{q \mid \exists q_1 \in [\![pat(G_1)]\!]^{\Delta,\widehat{s}},\ldots,\exists q_n \in [\![pat(G_n)]\!]^{\Delta,\widehat{s}}.\ f(q_1,\ldots,q_n) \to q \text{ in } \Delta\}$. Let $q_f \in [\![pat(G)]\!]^{\Delta,\widehat{s}} \cap F \neq \emptyset$. Then by the induction hypothesis, there exists $t_1 \in Inst(pat(G_1)),\ldots,t_n \in Inst(pat(G_n))$ such that $q_f \in [\![f(t_1,\ldots,t_n)]\!]^{\Delta,\hat{s}}$, and thus $f(t_1,\ldots,t_n) \in Inst(pat(G)) \cap L(A)$. Hence the property holds.

□