Regular Matching and Inclusion on Compressed Tree Patterns with Context Variables

Iovka Boneva[1], Joachim Niehren[2], and Momar Sakho[2]

[1] Université de Lille, France
[2] Inria Lille, France

Abstract. We study the complexity of regular matching and inclusion for compressed tree patterns extended by context variables. The addition of context variables to tree patterns permits us to properly capture compressed string patterns but also compressed patterns for unranked trees with tree and hedge variables. Regular inclusion for the latter is relevant to certain query answering on XML streams with references.[3]

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:

Pattern matching: Is a given algebraic value an instance of a given pattern?
Pattern unification: Do two given patterns have some common instance?
Regular pattern matching: Does some instance of a given pattern belong to a given regular language?
Regular pattern inclusion: Do all instances of a given pattern belong to a 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 singleton context-free grammars (also called straight-line programs), and a regular string language may be represented by nondeterministic finite automata (NFAs) or by deterministic finite automata (DFAs). The problem of string pattern matching is well-known to be NP-complete for NFAs [**?**] but in P for DFAs, with and without compression [**?**]. The more general problem of string unification is known to be PSPACE-complete [**?**].

Compressed patterns were called hyperstreams in [**?**]. Regular inclusion on compressed patterns is the problem of certain query answering on hyperstreams for queries defined by automata. This application motivated the study of regular inclusion and matching in [**?**]. For string patterns, both problems were shown to be PSPACE-complete, for DFAs and NFAs, with and without compression. See Fig. **??** 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. **??**. The

[3] We are grateful to Sylvain Salvati for pointing out and helping to solve difficulties.

	DFAs	NFAs
Regular Matching	PSPACE-c	PSPACE-c
Regular Inclusion	PSPACE-c	PSPACE-c

Fig. 1: (Compressed) string patterns.

	DFAs	NFAs
Regular Matching	P	P
Regular Inclusion	P	PSPACE-c

Fig. 2: Linear restriction.

	DTAs	NTAs
Regular Matching	NP-c	EXP-c
Regular Inclusion	CONP-c	EXP-c

Fig. 3: (Compressed) tree patterns.

	DTAs	NTAs
Regular Matching	P	P
Regular Inclusion	P	EXP-c

Fig. 4: Linear restriction.

	DTAs	NTAs
Regular Matching	EXP-c	EXP-c
Regular Inclusion	EXP-c	EXP-c

Fig. 5: Adding context variables.

	DTAs	NTAs
Regular Matching	P	P
Regular Inclusion	P	EXP-c

Fig. 6: Linear restriction.

problem which remains PSPACE-complete is regular inclusion on linear string patterns for NFAs.

The complexity landscapes of regular matching and inclusion for tree patterns look quite different to the case of string patterns, see Figs. **??** and **??**. Here, regular languages are defined by tree automata, which may either be nondeterministic (NTAs) or (bottom-up) deterministic (DTAs), while compressed descriptions of tree patterns can be obtained by singleton tree grammars. Regular matching for tree patterns against NTAs is EXP-complete with and without compression. For DTAs, however, regular matching is NP-complete and regular inclusion CONP-complete. The cases of tree pattern matching without compression were already mentioned in TATA [**?**]. Similar arguments apply to regular inclusion problems and for adding compression. For linear tree patterns, three of the 4 problems are in P except for the case of regular inclusion against NTAs.

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 and tree variable Y. The interest of adding context variables to tree patterns was already noticed when generalizing string pattern matching to context pattern matching [**?**], 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 [**?**]). Since we are here interested in a proper generalization of regular matching and inclusion from string patterns to tree patterns, we propose to study these problems for tree patterns extended with context variables.

The main contributions of the present paper are the complexity classes of regular matching and inclusion for tree patterns with context variables, which are summarized in Figs. **??** and **??**. The results are fully symmetric to those for string patterns, except that PSPACE-completeness there is to be replaced

Tree patterns	$p, p_1, \ldots, p_n \in \mathcal{P}^e_\Sigma ::= x \mid f(p_1, \ldots, p_n) \mid P@p$
Context patterns	$P \in \mathcal{P}^c_\Sigma ::= X \mid \lambda x.p$ where x occurs exactly once in p.

Fig. 7: Tree patterns context patterns, where $x \in \mathcal{V}^e$, $X \in \mathcal{V}^c$, $n \geq 0$, $f \in \Sigma^{(n)}$.

by Exp-completeness here. The main reason for this change is that the central problems of context inhabitation for Dtas and resp. Ntas are Exp-complete, while the inhabitation problems for Dfas and resp. Nfas are PSpace-complete.

Finally, we show that regular pattern matching and inclusion have the same complexity for (compressed) patterns on unranked trees with tree and hedge variables, mainly since such patterns can be encoded into (compressed ranked) tree patterns with context variables. Compressed patterns for unranked trees capture systems of Xml streams with references [**?**]. They permit to generalize the notion of hyperstreams in [**?**] from strings to unranked trees.

Outline. We introduce tree patterns with context variables in Section **??**. The inhabitation problem for Σ-algebras is defined in Section **??**. The complexity of context inhabitation for tree automata is discussed in Section **??**. Compressed tree patterns with context variables are introduced in Section **??** and then studied for regular matching and inclusion in Section **??**. Due to lack of space, the discussion of the special cases of linear patterns and patterns with context variables are delegated to the longer online version. The same for the missing proofs.

2 Tree Patterns with Context Variables

We consider the set $\mathbf{T} = \{e, c\}$ with the type e for trees and the type $c = e \multimap e$ for contexts. This linear type is inspired by linear logic. The usual nonlinear function type $e \to e$ would be written as $!e \multimap e$ there. We assume sets of tree variables $\mathcal{V}^e$ and of context variables $\mathcal{V}^c$. The tree variables are ranged over by x, y, z and the context variables by X, Y. The set of all variables is denoted by $\mathcal{V} = \cup_{\tau \in \mathbf{T}} \mathcal{V}^\tau$.

We fix a finite ranked signature $\Sigma = \uplus_{n \geq 0} \Sigma^{(n)}$ of function symbols $f \in \Sigma^{(n)}$ of arity n. We assume that Σ contains at least one constant and one symbol of arity at least 2. The set of trees $t \in \mathcal{T}_\Sigma$ is the least set that contains all elements $f(t_1, \ldots, t_n)$ 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 $a \in \Sigma^{(0)}$. The set of contexts $C \in \mathcal{C}_\Sigma$ is the set of all terms $\lambda x.p$ such that $p \in \mathcal{T}_{\Sigma \uplus \{x\}}$ for some tree variable $x \in \mathcal{V}^e$ that occurs exactly once in p. The set of all values of both types is $\mathit{Val}_\Sigma = \mathcal{T}_\Sigma \cup \mathcal{C}_\Sigma$.

The sets of all *tree patterns* $p \in \mathcal{P}^e_\Sigma$ and of all *context patterns* $P \in \mathcal{P}^c_\Sigma$ are defined in Fig. **??**. Note that both types of patterns may contain context variables. The set of all *patterns* is $\mathcal{P}_\Sigma = \uplus_{\tau \in \mathbf{T}} \mathcal{P}^\tau_\Sigma$. The sets of free variables $\mathit{fv}(p)$ and $\mathit{fv}(P)$ and of bound variables $\mathit{bv}(p)$ and $\mathit{bv}(P)$ can be defined as usual. The set $\mathcal{G}^\tau_\Sigma$ of ground patterns of type $\tau \in \mathbf{T}$ is the subset of patterns in $\mathcal{P}^\tau_\Sigma$ without free variables. The set of ground patterns of all types is denoted by $\mathcal{G}_\Sigma = \mathcal{G}^e_\Sigma \cup \mathcal{G}^c_\Sigma$. Clearly, any *tree* $t \in \mathcal{T}_\Sigma$ is a ground pattern of type e and any

context $C \in \mathcal{C}_\Sigma$ is a ground pattern of type c. A pattern is called *linear* if each of its free variables has at most one free occurrence.

We can apply β-reduction to both kinds of patterns. 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 apart before. Any ground tree pattern $p \in \mathcal{G}^e_\Sigma$ can be β-reduced in a linear number of steps to some tree in polynomial time, since all λ-binders are assumed to be linear. The semantics $[\![p]\!]$ of a ground pattern p is the tree obtained from p by exhaustive β-reduction. Similarly, any ground context pattern $P \in \mathcal{G}^c_\Sigma$ can be β-reduced in a linear number of steps to a unique context $\lambda x.p \in \mathcal{C}_\Sigma$. The semantics $[\![P]\!]$ is the function $[\![P]\!] : \mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$ such that $[\![P]\!](t) = p[x/t]$ for any tree t. Note that $[\![\mathcal{G}^e_\Sigma]\!] = [\![\mathcal{T}_\Sigma]\!]$ is equal to $\mathcal{T}_\Sigma$ while $[\![\mathcal{G}^c_\Sigma]\!] = [\![\mathcal{C}_\Sigma]\!]$ is a proper subset of the set of functions of type $\mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$.

A substitution $\sigma : V \to \mathcal{G}_\Sigma$ where $V \subseteq \mathcal{V}$ is called well-typed if it maps tree variables to $\mathcal{G}^e_\Sigma$ and context variables to $\mathcal{G}^c_\Sigma$. For any patterns $p \in \mathcal{P}^e_\Sigma$, the grounding $\sigma(p) \in \mathcal{G}^e_\Sigma$ is obtained by applying σ to the free variables in p. The set of all instances of p is obtained by β-normalizing all groundings:

$$\mathit{Inst}(p) = \{[\![\sigma(p)]\!] \mid \sigma : \mathit{fv}(p) \to \mathcal{G}_\Sigma \text{ well-typed}\}$$

Clearly, $\mathit{Inst}(p) \subseteq \mathcal{T}_\Sigma$. For example, consider the tree pattern $p = X@(X@a)$ and the substitution σ where $\sigma(X) = \lambda x.f(b, x)$ and $\sigma(x) = a$. Then the β-normalization of the grounding $\sigma(p) = \sigma(X)@(\sigma(X)@\sigma(x))$ is the tree $t = f(b, f(b, a))$, i.e. $t \in \mathit{Inst}(p)$. Similarly, for any $P \in \mathcal{P}^c_\Sigma$, we can define the grounding $\sigma(P) \in \mathcal{G}^c_\Sigma$. The set of instance $\mathit{Inst}(P)$ contains the semantics of all groundings of P. Clearly $\mathit{Inst}(P) \subseteq [\![\mathcal{C}_\Sigma]\!]$.

3 Inhabitation for Σ-Algebras

We recall the notion of inhabitation by trees and contexts in Val_Σ for Σ-algebras, and then relate it to the notion of pattern evaluation in Σ-algebras.

A Σ-algebra $\Delta = (\mathit{dom}^\Delta, .^\Delta)$ consists of a set $D = \mathit{dom}^\Delta$ called the domain, and a mapping $.^\Delta$ that interprets symbols $f \in \Sigma^{(n)}$ as functions $f^\Delta : D^n \to D$. In particular, the set of trees $\mathcal{T}_\Sigma$ yields a Σ-algebra, known as the *term algebra*, whose domain is $\mathcal{T}_\Sigma$ and whose interpretation satisfies $f^{\mathcal{T}_\Sigma}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. Depending on their type, we can interpret values in Val_Σ as elements of dom^Δ or as functions on dom^Δ. The *interpretation* $[\![t]\!]^\Delta$ of a tree $t = f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ is the element $[\![t]\!]^\Delta = f^\Delta([\![t_1]\!]^\Delta, \ldots, [\![t_n]\!]^\Delta)$, while the *interpretation* of a context $C = \lambda x.p \in \mathcal{C}_\Sigma$ is the function $[\![C]\!]^\Delta : D \to D$ with $[\![C]\!]^\Delta(d) = [\![p[x/d]]\!]^\Delta$ for all $d \in D$.

Definition 1. *Let Δ be a Σ-algebra. An element $d \in \mathit{dom}^\Delta$ is called Δ-inhabited, if there exists a tree $t \in \mathcal{T}_\Sigma$ such that $d = [\![t]\!]^\Delta$. A function $S : \mathit{dom}^\Delta \to \mathit{dom}^\Delta$ is called Δ-inhabited if there exists a context $C \in \mathcal{C}_\Sigma$ such that $S = [\![C]\!]^\Delta$.*

The subset all Δ-inhabited elements and functions is $[\![\mathit{Val}_\Sigma]\!]^\Delta = [\![\mathcal{T}_\Sigma]\!]^\Delta \cup [\![\mathcal{C}_\Sigma]\!]^\Delta$. We next lift algebra interpretation on values to algebra evaluation on

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

Fig. 8: Algebra evaluation of patterns.

patterns. We call a variable assignment $\sigma : V \to [\![\mathit{Val}_\Sigma]\!]^\Delta$ with $V \subseteq \mathcal{V}$ well-typed, if σ maps tree variables to $[\![\mathcal{T}_\Sigma]\!]^\Delta$ and context variables to $[\![\mathcal{C}_\Sigma]\!]^\Delta$. In Fig. **??**, we define for any tree pattern p and well-typed variable assignment $\sigma : V \to [\![\mathit{Val}_\Sigma]\!]^\Delta$ with $\mathit{fv}(p) \subseteq V$ the evaluation $[\![p]\!]^{\Delta,\sigma} \in [\![\mathcal{T}_\Sigma]\!]^\Delta$, and similarly $[\![P]\!]^{\Delta,\sigma} \in [\![\mathcal{C}_\Sigma]\!]^\Delta$ for all context patterns P with $\mathit{fv}(P) \subseteq V$. The evaluation of ground patterns $p, P \in \mathcal{G}_\Sigma$ in Δ does not depend on the variable assignment σ. Therefore we can write $[\![p]\!]^\Delta$ instead of $[\![p]\!]^{\Delta,\sigma}$ and $[\![P]\!]^\Delta$ instead of $[\![P]\!]^{\Delta,\sigma}$. Clearly, algebra evaluation restricted to values is equal to algebra interpretation. Furthermore, note that $[\![\mathit{Val}_\Sigma]\!]^\Delta = [\![\mathcal{G}_\Sigma]\!]^\Delta$ since any ground pattern in $\mathcal{G}_\Sigma$ can be β-reduced to some value in Val_Σ which has the same interpretation. Note also that the notion of Δ-inhabitation does not change when based on ground patterns instead of values.

Consider a well-typed variable assignment $\sigma : V \to \mathcal{G}_\Sigma$. Then $\sigma \circ [\![.]\!]^\Delta$ is a well-typed variable assignment into $[\![\mathcal{G}_\Sigma]\!]^\Delta = [\![\mathit{Val}_\Sigma]\!]^\Delta$, such that $[\![\sigma(p)]\!]^\Delta = [\![p]\!]^{\Delta,\sigma\circ[\![.]\!]^\Delta}$ for all tree patterns p with $\mathit{fv}(p) \subseteq V$. As a consequence for the term algebra, the set of instances $\mathit{Inst}(p)$ of a tree pattern p is equal to $\{[\![p]\!]^{\mathcal{T}_\Sigma,\sigma\circ[\![.]\!]^{\mathcal{T}_\Sigma}} \mid \sigma : \mathit{fv}(p) \to \mathcal{G}_\Sigma \text{ well-typed}\}$, and similarly for context patterns P.

4 Inhabitation for Tree Automata

We recall the notion of tree automata for recognizing *regular languages of trees* and discuss tree and context inhabitation problems for tree automata. As we will see in the following section, these inhabitation problems are closely related to regular matching and inclusion for patterns with tree and context variables.

Definition 2. *A (nondeterministic) tree automaton (*NTA*) over Σ is a tuple $A = (Q, \Sigma, F, \Delta)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and $\Delta \subseteq \cup_{n\geq 0}\Sigma^{(n)} \times Q^{n+1}$ the transition relation.*

A rule $(f, q_1, \ldots, q_n, q) \in \Delta$ is written as $f(q_1, \ldots, q_n) \to q$. The transition Σ-algebra of NTA A – that we equally denote by Δ – has as its domain 2^Q and interprets the function symbols $f \in \Sigma^{(n)}$ where $n \geq 0$ as the n-ary functions f^Δ such that for all subsets of states $Q_1 \ldots, Q_n \subseteq Q$:

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

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:

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

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

Tree Inhabitation. Let NTA_Σ be the set of all NTAs with signature Σ, and similarly DTA_Σ. We call NTA and DTA an automata class. For any automata class $\mathcal{A}$ and signature Σ, tree inhabitation is the following problem:

Inhab$^e_\Sigma(\mathcal{A})$**.** *Input:* A tree automaton $A = (Q, \Sigma, F, \Delta) \in \mathcal{A}_\Sigma$, $Q' \subseteq Q$.
Output: The truth value of whether Q' is Δ-inhabited.

Theorem 1 (Folklore). *Tree inhabitation* $\text{INHAB}^e_\Sigma(\text{NTA})$ *is* EXP*-complete, while its restriction by deterministic tree automata* $\text{INHAB}^e_\Sigma(\text{DTA})$ *is in* P*.*

Proof. Let $A = (Q, \Sigma, F, \Delta)$ be an NTA and $Q' \subseteq Q$. By definition, Q' is Δ-inhabited iff there exists a tree $p \in \mathcal{T}_\Sigma$ such that $[\![p]\!]^\Delta = Q'$, which is equivalent to that $[\![p]\!]^{\det(\Delta)} = \{Q'\}$. Thus Q' is Δ-inhabited iff Q' is accessible in the tree automaton $\det(A)$. This can be tested in polynomial time from $\det(A)$ which is computed in exponential time. Thus $\text{INHAB}^e_\Sigma(\text{NTA})$ is in EXP. If A is a DTA, then there is no need to determinize it and Q' is a singleton. It is thus sufficient to test whether Q' is accessible in A. Hence $\text{INHAB}^e_\Sigma(\text{DTA})$ is in polynomial time.

For the lower bound, we have to show that $\text{INHAB}^e_\Sigma(\text{NTA})$ is EXP-hard. We do so by reduction from the problem of non-emptiness of the intersection of a sequence of DTAs, which is well known to be EXP-complete [**?**]. Let $A_1, \ldots, A_n$ be a sequence of DTAs with alphabet Σ. Suppose that $A_i = (Q^i, \Sigma, \Delta^i, F^i)$. Without loss of generality, we can assume that each of them has a single final state $F^i = \{q^i_f\}$. 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 $F = \{q^1_f, \ldots, q^n_f\}$. Since all A_i are deterministic, we can then show that $t \in \cap_{i=1}^n L(A_i)$ iff F is Δ-inhabited by t.

Context Inhabitation. Contexts evaluate to very particular functions in transition algebras of tree automata, since they use their bound variable once.

Definition 3. *A union homomorphism on* 2^Q *is a function* $S : 2^Q \to 2^Q$ *such that* $S(\emptyset) = \emptyset$ *and for all* $Q', Q'' \subseteq Q$, $S(Q' \cup Q'') = S(Q') \cup S(Q'')$*.*

Lemma 1 (Folklore). *For any context* $C \in \mathcal{C}_\Sigma$ *and* NTA $A = (Q, \Sigma, F, \Delta)$ *the semantics* $[\![C]\!]^\Delta$ *is a union homomorphism on* 2^Q*.*

The main reason to restrict ourselves to contexts, is that Lemma **??** would fail for nonlinear λ-terms such as $N = \lambda x.f(x, x)$. In order to see this, consider 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

that $[\![N]\!]^{\Delta}$ is not a union homomorphism and cannot be represented by a function $s : Q :\to 2^Q$ as stated in Lemma **??**. Since union homomorphisms are determined by their images on singletons, they can be represented by functions $s : Q \to 2^Q$. Conversely such a function defines the union homomorphism $\hat{s} : 2^Q \to 2^Q$ such that for any $Q' \subseteq Q$: $\hat{s}(Q') = \cup_{q \in Q'} s(q)$.

Lemma 2. *If $S : 2^Q \to 2^Q$ is a union homomorphism then $S = \hat{s}$ where $s : Q \to 2^Q$ is the function with $s(q) = S(\{q\})$ for all $q \in Q$.*

We next consider the problem of context-inhabitation for tree automata. Here, the input here will be a succinct descriptor of a union homomorphism:

Inhab$^c_\Sigma(\mathcal{A})$. *Input:* An automaton $A = (Q, \Sigma, F, \Delta) \in \mathcal{A}_\Sigma$, $s : Q \to 2^Q$.
Output: The truth value of whether $\hat{s}$ is Δ-inhabited.

Context inhabitation is a restriction of the more general λ-*definability* problem, which is undecidable [**?**,**?**]. However, λ-definability for orders up to 3 is decidable [**?**], and context-inhabitation is a special case of second-order λ-definability. Its precise complexity, however, has not been studied so far to the best of our knowledge.

Proposition 1. *Let $A = (Q, \Sigma, F, \Delta)$ be an* NTA *and $s : Q \to 2^Q$. 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 backwards 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 $\hat{s}(Q') = \cup_{q \in Q'} s(q) = \cup_{q \in Q'} [\![C]\!]^{\Delta}(\{q\}) = [\![C]\!]^{\Delta}(Q')$ since $[\![C]\!]^{\Delta}$ is a union-homomorphism by Lemma **??**. Thus $\hat{s}$ is Δ-inhabited.

Theorem 2. *For both classes of tree automata $\mathcal{A} \in \{\text{NTA}, \text{DTA}\}$ the context-inhabitation problem* INHAB$^c_\Sigma(\mathcal{A})$ *is* EXP*-complete.*

Proof. Since INHAB$^c_\Sigma$(DFA) is EXP-complete, a naive exponential time reduction from INHAB$^c_\Sigma$(NFA) to INHAB$^c_\Sigma$(DFA) would lead to a doubly exponential time algorithm. Nevertheless, we will present a single exponential time algorithm for INHAB$^c_\Sigma$(NTA) based on determinization. Let $A = (Q, \Sigma, F, \Delta)$ be an NTA where $Q = \{q_1, \ldots, q_n\}$ and $s : Q \to 2^Q$. We fix a variable $x \in \mathcal{V}^e$ arbitrarily. For each $i \in \{1, \ldots, n\}$, let $A_i = (Q, \Sigma \uplus \{x\}, \Delta \cup \{x \to q_i\}, F)$. For any context $C = \lambda x.p$, $[\![p]\!]^{A_i}$ is the set of states to which C can be evaluated when starting at the hole marker x with state q_i. Let $\tilde{A}$ be the product DTA $\tilde{A} = \det(A_1) \times \ldots \times \det(A_n)$. Note that the number of states of $\tilde{A}$ is at most $(2^n)^n = 2^{n^2}$, which is exponential. Furthermore, the tuple $(s(q_1), \ldots, s(q_n))$ is an accessible state of $\tilde{A}$ if and only if there is a context $\lambda x.p \in \mathcal{C}_\Sigma$ such that for all $1 \leq i \leq n$, $[\![\lambda x.p]\!]^{\Delta}(\{q_i\}) = s(q_i)$. By Proposition **??** this is equivalent to that $\hat{s}$ is Δ-inhabited. Testing whether $(s(q_1), \ldots, s(q_n))$ is accessible in $\tilde{A}$ is in polynomial time in the size of $\tilde{A}$ and thus in exponential time too.

The EXP-hardness of INHAB$^c_\Sigma$(DFA) can be shown by reduction to the intersection problem of DTAs. The idea of the proof is similar to that of the PSPACE-hardness proof of DFA-inhabitation (see [**?**]), so we omit the details.

5 Compressed Tree Patterns

We now recall compressed tree patterns with context variables that are defined by partial singleton tree grammars.

Definition 4. *A* compressed pattern (with context variables) *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,* $S \in N \cap \mathcal{V}^\tau$ *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 tree patterns of type* τ *is denoted by* $c\mathcal{P}^\tau_\Sigma$*.*

For instance, consider the compressed tree pattern $G \in c\mathcal{P}^e_\Sigma$ with the nonterminals $N = \{x, X, Y, Z, y\}$ where $S = x$ and two rules $R(x) = X@a(X@b, Y@c)$, and $R(X) = \lambda x.Z@a(x, y)$. This grammar is acyclic, in that no variable on the left hand side of some rule can appear in any subsequent rule. It should be noticed that the tree language of grammar G is $\emptyset$. What we are interested instead is the tree pattern $pat(G) = (\lambda x.Z@a(x, y))@a((\lambda x.Z@a(x, y))@b, Y@c)$ that this compressed tree pattern represents in a compressed manner. By exhaustive β-reduction, we obtain the tree pattern with context variables $[\![pat(G)]\!] = Z@a(a(Z@a(b, y), Y@c), y)$. The free variables of a compressed tree pattern G are the free variables of $pat(G)$. Its bound variables are the nonterminals in $dom(R)$ and the bound variables on the right-hand sides of these rules.

In what follows we will identify any tree pattern in $\mathcal{P}^e_\Sigma$ with the compressed tree pattern in $c\mathcal{P}^e_\Sigma$ that has a single rule mapping a new start symbol to the tree pattern. This compressed tree patterns has no compression. In this sense, $\mathcal{P}^e_\Sigma \subseteq c\mathcal{P}^e_\Sigma$. A compressed tree pattern $G \in c\mathcal{P}^\tau_\Sigma$ where $\tau \in \mathbf{T}$ is called *linear* if its tree pattern $pat(G)$ is linear.

Let $A = (Q, \Sigma, F, \Delta)$ be an NTA, $V \subseteq \mathcal{V}$ a finite subset of variables, and σ a function with domain V that maps any tree variable $x \in V$ to $\sigma(x) \subseteq Q$ and any context variable $X \in V$ to a function $\sigma(X) : Q \to 2^Q$. Note that $\sigma(X)$ represents the union homomorphism $\widehat{\sigma(x)} : 2^Q \to 2^Q$. Let $\hat{\sigma}$ be such that $\hat{\sigma}(x) = \sigma(x)$ for all $x \in V$ and $\hat{\sigma}(X) = \widehat{\sigma(X)}$ for all $X \in V$.

Lemma 3. *For any* $G = (N, \Sigma, R, S) \in c\mathcal{P}^e_\Sigma$ *with* $fv(G) \subseteq V$ *we can compute* $[\![pat(G)]\!]^{\Delta,\hat{\sigma}}$ *in polynomial time from* A*,* G*, and* σ*.*

Proof. The algorithm evaluates the pattern inductively along the partial order on the nonterminals of G. 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 $[\![pat(G_v)]\!]^{\Delta,\hat{\sigma}}$ can be computed in polynomial time from A, G, and σ. In particular this holds for $[\![pat(G)]\!]^{\Delta,\hat{\sigma}} = [\![pat(G_S)]\!]^{\Delta,\hat{\sigma}}$.

6 Regular Matching and Inclusion

We now study the complexity of regular matching and inclusion for classes $\mathcal{H}$ of compressed tree patterns with context variables such as $\mathcal{P}^e$ and $c\mathcal{P}^e$.

Definition 5. *For any classes $\mathcal{H}$ of compressed tree patterns and $\mathcal{A}$ of* NTA*s, and for any ranked alphabet Σ we define two decision problems:*

Regular pattern inclusion $\textsc{Incl}_\Sigma(\mathcal{H}, \mathcal{A})$. *Input: A compressed tree pattern $G \in \mathcal{H}_\Sigma$ and a tree automaton $A \in \mathcal{A}_\Sigma$.*
Output: The truth value of whether $Inst(pat(G)) \subseteq L(A)$.

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

The following characterization of regular matching induces a decision procedure by reduction to context inhabitation, and is useful in the hardness proof.

Lemma 4. *Let $A = (Q, \Sigma, F, \Delta)$ be an* NTA*, $p \in \mathcal{P}^e_\Sigma$ be a tree pattern. Then $Inst(p) \cap L(A) \neq \emptyset$ if and only if there exists a well-typed assignment into Δ-inhabited subset of states and union-homomorphisms $\sigma : fv(p) \to [\![Val_\Sigma]\!]^\Delta$ such that $[\![p]\!]^{\Delta,\sigma} \cap F \neq \emptyset$.*

Proposition 2 (Lower Bound Matching). $\textsc{Match}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ *is* EXP*-hard.*

Proof. We reduce $\textsc{Inhab}^c_\Sigma(\textsc{Dta})$ to $\textsc{Match}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ in polynomial time. As $\textsc{Inhab}^c_\Sigma(\textsc{Dta})$ was shown to be EXP-complete in Theorem **??**, it follows that $\textsc{Match}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ is EXP-hard.

Let $A = (Q, \Sigma, F, \Delta)$ be a DTA and $s : Q \to 2^Q$ be a function. We set $Q = \{q_1, \ldots, q_n\}$ and consider a new symbol $\# \notin \Sigma$ of arity n and a new state $q_\#$. From this we build a new DTA $\tilde{A} = (\tilde{Q}, \tilde{\Sigma}, \tilde{F}, \tilde{\Delta})$ where $\tilde{Q} = Q \cup \{q_\#\} \cup \{s(q_i) \mid 1 \leq i \leq n\}$, $\tilde{\Sigma} = \Sigma \cup \{\#\} \cup Q$, $\tilde{F} = \{q_\#\}$ and $\tilde{\Delta} = \Delta \cup \{\#(s(q_1), \ldots, s(q_n)) \to q_\#\}$. Let $X \in \mathcal{V}^c$ and $p = \#(X@q_1, \ldots, X@q_n) \in \mathcal{P}^e_{\tilde{\Sigma}}$. The reduction is induced by the following claim, whose technical proof can be based on Lemma **??** without any special tricks.

Claim. The function $\hat{s}$ is Δ-inhabited if and only if $Inst(p) \cap L(\tilde{A}) \neq \emptyset$. □

Lemma 5 (Complementation). *Regular inclusion and matching are complementary problems for deterministic automata: For any class of compressed tree patterns $\mathcal{H}$,* $\textsc{Incl}_\Sigma(\mathcal{H}, \textsc{Dta})$ *and* $\textsc{coMatch}_\Sigma(\mathcal{H}, \textsc{Dta})$ *are equivalent modulo* P*.*

Proof. For a compressed tree pattern G and an NTA A, $Inst(p) \subseteq L(A)$ iff $Inst(p) \cap \overline{L(A)} = \emptyset$ iff $= Inst(p) \cap L(\overline{A}) = \emptyset$, and the complementation operation is polynomial for DTAs and exponential for NTAs– since determinization is needed.

Proposition 3 (Lower Bound Inclusion). $\textsc{Incl}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ *is* EXP*-hard.*

Proof. Lemma **??** states that $\textsc{Incl}_\Sigma(\mathcal{P}^e, \textsc{Dta}) = \textsc{coMatch}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ modulo P. By Proposition **??**, $\textsc{Match}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ is EXP-hard and since EXP is closed by complement, $\textsc{coMatch}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ is EXP-hard too. It then holds that $\textsc{Incl}_\Sigma(\mathcal{P}^e, \textsc{Dta})$ is EXP-hard.

We next reduce the problems of regular matching and inclusion to context inhabitation for tree automata in order to upper complexity bounds.

Proposition 4 (Upper Bounds). $\textsc{Match}_\Sigma(c\mathcal{P}^e, \textsc{Nta})$ *and* $\textsc{Incl}_\Sigma(c\mathcal{P}^e, \textsc{Nta})$ *are in* Exp.

Proof. Let $G \in c\mathcal{P}^e_\Sigma$ be a compressed tree pattern with start symbol $S \in \mathcal{V}^e$ and set of nonterminals N, and $A = (Q, \Sigma, F, \Delta)$ be an Nta. According to Lemma **??**, in order to decide whether $pat(G)$ matches $L(A)$ it is sufficient to find a well-typed assignment σ with domain $fv(G)$ such that $\sigma(x) \subseteq Q$ for all $x \in fv(G)$ and $\sigma(X) : Q \to 2^Q$ for all $X \in fv(G)$. Furthermore, $\widehat{\sigma} : fv(G) \to [\![Val_\Sigma]\!]$ must map to Δ-inhabited subsets of Q and Δ-inhabited union-homomorphisms of type $2^Q \to 2^Q$ such that $\widehat{\sigma}(pat(G)) \cap F \neq \emptyset$. Thus the algorithm will iterate over all such σ, test the inhabitation of $\widehat{\sigma}(v)$ for all $v \in N$, before checking that $\widehat{\sigma}(pat(G)) \cap F \neq \emptyset$. It is successful if the test succeeds for some σ. The number of iterations of the algorithm is at most $2^{|Q|^2 \cdot |fv(G)|}$. Moreover inhabitation can be tested in time $O(2^{|Q|^2})$ by Theorems **??** and **??** while $\widehat{\sigma}(pat(G))$ is computed in polynomial time from A, G, and σ by Lemma **??**. Thus the algorithm is in Exp. For $\textsc{Incl}_\Sigma(c\mathcal{P}^e, \textsc{Nta})$, the algorithm is similar except that the condition $\widehat{\sigma}(pat(G)) \cap F \neq \emptyset$ must hold for all $\widehat{\sigma}$ mapping $fv(G)$ to Δ-inhabited sets of states and functions.

7 Encoding Patterns for Unranked Trees

7.1 Patterns of Unranked Trees and Hedges Without Context Variables

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 indeed be reduced to the corresponding problems of (ranked) tree patterns with context variables.

Unlike ranked trees, unranked trees are constructed from symbols without fixed arities. Let Γ be a finite set of such symbols. The set of hedges $\mathcal{H}_\Gamma$ is the least set that contains all words of hedges on ${\mathcal{H}_\Gamma}^*$ 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. **??**. 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 unranked tree 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 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 how to encode hedge patterns into (ranked) context patterns. For instance, we will encode the hedge pattern $H_0 = a(ZbcY)$ into the context

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

Fig. 9: Encoding hedge patterns $H \in \mathcal{P}^h_\Gamma$ into context patterns $\langle H\rangle^c \in \mathcal{P}^c_\Sigma$, where $Y \in \mathcal{V}^u$, $Z \in \mathcal{V}^h$, $a \in \Gamma$, and ε the empty word.

pattern $\langle H_0\rangle^c = \lambda y.a(Z @ (b(\#, c(\#, Y @ \#))), y)$. The concatenation operation on hedges is simulated by the application operation of contexts. When given type u of unranked trees rather than the h of hedges, H_0 can be encoded to the (ranked) tree pattern $\langle H_0\rangle^e = \langle H_0\rangle^c @ \#$. The ranked signature we need for this encoding is $\Sigma = \Sigma^{(2)} \uplus \Sigma^{(0)}$ where $\Sigma^{(2)} = \Gamma$ and $\Sigma^{(0)} = \{\#\}$. As set of context variables we use $\mathcal{V}^c = \mathcal{V}^u \uplus \mathcal{V}^h$. The set $\mathcal{V}^e$ of tree variables is left arbitrary. The encoding in Fig. **??** maps hedge patterns $H \in \mathcal{P}^h_\Gamma$ to (ranked) context patterns $\langle H\rangle^c \in \mathcal{P}^c_\Sigma$. We finally encode patterns for unranked trees $H \in \mathcal{P}^u_\Gamma$ as (ranked) tree patterns with context variables $\langle H\rangle^e \in \mathcal{P}^e_\Sigma$ such that $\langle H\rangle^e = \langle H\rangle^c @ \#$.

In order to show the soundness of this encoding, we have to consider assignments $\sigma : V \to \mathit{Val}_\Sigma$ that map unranked tree variables to $\langle \mathcal{U}_\Gamma\rangle^c$ and hedge variables to $\langle \mathcal{H}_\Gamma\rangle^c$. We call such assignments *unranked*. The instance sets but now restricted to unranked assignments are denoted by $\mathit{Inst}^{\mathrm{unr}}(p) = \{[\![\sigma(p)]\!] \mid \sigma : \mathit{fv}(p) \to \mathit{Val}_\Sigma$ well-typed and unranked$\}$ and $\mathit{Inst}^{\mathrm{unr}}(P)$ similarly. Notice that unr can be seen as an instantiation constraint that associates every free tree variable to a Dta that recognizes the language of unranked tree encodings (into ranked trees) and every free context variable to a Dta that recognizes the language of hedge encodings.

Lemma 6. $[\![\langle \mathit{Inst}(H)\rangle^e]\!] = \mathit{Inst}^{unr}(\langle H\rangle^e)$ *for any* $H \in \mathcal{P}^u_\Gamma$.

Proof. We can prove for any $H \in \mathcal{P}^u_\Gamma$ that $[\![\langle \mathit{Inst}(H)\rangle^c]\!] = \mathit{Inst}^{\mathrm{unr}}(\langle H\rangle^c)$ by induction of the structure of H. This claim implies the lemma.

The problems of regular matching $\textsc{Match}_\Gamma(\mathcal{P}^u, \mathcal{A})$ and regular inclusion $\textsc{Incl}_\Gamma(\mathcal{P}^u, \mathcal{A})$ can now be defined for patterns on unranked trees in the analogous way as for ranked trees, except that now matching is modulo encoding, i.e., the tree automata of class $\mathcal{A}_\Sigma$ recognize subsets of encodings of unranked trees in $\langle \mathcal{U}_\Gamma\rangle^e$.

Proposition 5. *For any* $\mathcal{A} \in \{\textsc{Dta}, \textsc{Nta}\}$ *we have polynomial time reductions from* $\textsc{Match}_\Gamma(\mathcal{P}^u, \mathcal{A})$ *to* $\textsc{Match}_{\Sigma'}(\mathcal{P}^e, \mathcal{A})$ *and* $\textsc{Incl}_\Gamma(\mathcal{P}^u, \mathcal{A})$ *to* $\mathit{Incl}_{\Sigma'}(\mathcal{P}^e, \mathcal{A})$ *for some signature* Σ' *derived from* Σ.

Proof. The basic idea is to use Lemma **??**, but for this we also need to constrain the assignments for the encoded patterns to be unranked. When considering regular matching and inclusion for hedge patterns, this can be done by changing both the pattern and the tree automaton. We illustrate how this works on an example for the case of regular matching. Generalizing the example to a full

Unranked tree patterns with context variables	$\alpha \in \mathcal{P}^{uc}_\Gamma ::= \kappa \mid X_1@\alpha \mid X_2@\beta$
Hedge patterns with context variables	$\beta \in \mathcal{P}^{hc}_\Gamma ::= H \mid X_3@\alpha \mid X_4@\beta$

Fig. 10: Sets of unranked tree and hedges patterns, with $\kappa \in \mathcal{P}^u_\Gamma$, $H \in \mathcal{P}^h_\Gamma$, $X_1 \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{uc}}$, $X_2 \in \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{uc}}$, $X_3 \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}}$ and $X_4 \in \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{hc}}$.

proof is straightforward. We fix a letter $c \in \Gamma = \Sigma^{(2)}$ arbitrarily. Clearly, the language $L^h = \{[\![C@y]\!] \in \mathcal{T}_{\Sigma\cup\{y\}} \mid C \in \langle\mathcal{H}_\Gamma\rangle^c\}$ can then be recognized by some $A^h \in \text{DTA}_{\Sigma\cup\{y\}}$ that can be computed in polynomial time from Γ. Now consider the example of whether the unranked tree pattern $H = a(Z, b(\#, \#))$ matches the language of a given $A \in \text{DTA}_\Sigma$. This is equivalent to that the pattern $c(Z@y, H)$ matches the language $c(L^h, L(A))$. This language can be recognized by a $\text{DTA}_{\Sigma\cup\{y\}}$ that can be computed in polynomial time from A and A^h.

Theorem 3. *For any class of tree automata $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$ the problems $\text{MATCH}_\Gamma(\mathcal{P}^u, \mathcal{A})$ and $\text{INCL}_\Gamma(\mathcal{P}^u, \mathcal{A})$ are* EXP*-complete*

Proof. The upper bounds for $\text{MATCH}_\Sigma(\mathcal{P}^u, \text{NTA})$ and $\text{INCL}_\Sigma(\mathcal{P}^u, \text{NTA})$ follow via the polynomial time reduction from Proposition **??** from the upper bounds for $\text{MATCH}_\Sigma(\mathcal{P}^e, \text{NTA})$ and $\text{INCL}_\Sigma(\mathcal{P}^e, \text{NTA})$ in Proposition **??**. The EXP-hardness of $\text{MATCH}_\Gamma(\mathcal{P}^u, \text{DTA})$ and thus of the other 3 problems can be shown in analogy to the proof of Proposition **??**.

7.2 Patterns of Unranked Trees and Hedges With Context Variables

As for ranked trees, one could consider patterns of unranked trees and hedges with context variables. We shall show in the following that regular matching and inclusion with such kind of patterns can also be reduced to the case of patterns of ranked trees with context variables. We denote the set of patterns of unranked trees with context variables by $\mathcal{P}^{uc}_\Gamma$ and the set of patterns of hedges with context variables by $\mathcal{P}^{hc}_\Gamma$. We also assume that the set of context variables $\mathcal{V}^c$ is divided into four distinct subsets, so that $\mathcal{V}^c = \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{uc}} \uplus \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{uc}} \uplus \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}} \uplus \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{hc}}$ where $\mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{uc}}, \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{uc}}, \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}}$ and $\mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{hc}}$ are the sets of context variables that represent functions of type respectively $\mathcal{P}^{uc}_\Gamma \to \mathcal{P}^{uc}_\Gamma$, $\mathcal{P}^{hc}_\Gamma \to \mathcal{P}^{uc}_\Gamma$, $\mathcal{P}^{uc}_\Gamma \to \mathcal{P}^{hc}_\Gamma$ and $\mathcal{P}^{hc}_\Gamma \to \mathcal{P}^{hc}_\Gamma$ for some unranked alphabet Γ. Their abstract syntaxes are given in Fig. **??**.

The rules of encoding into ranked patterns are the same, except for the additional rules in Fig. **??**. Notice that the tree variables in $\mathcal{V}^e$ are encoded by convention to context variables in $\mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{uc}}$. We generalize the unr instantiation constraint to the different kind of context variables so that any $X \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{uc}} \cup \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}}$ is mapped to a DTA recognizing the set of tree encodings and any $X' \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}} \cup \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{hc}}$ to a DTA recognizing the set of hedge encodings.

$$\langle X@\alpha\rangle^e = \lambda y.(X@(\langle\alpha\rangle^e@y)) \text{ where } X \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{uc}} \text{ and } \alpha \in \mathcal{P}_\Gamma^{uc}$$
$$\langle X@\beta\rangle^e = \lambda y.(X@(\langle\beta\rangle^c@y)) \text{ where } X \in \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{uc}} \text{ and } \beta \in \mathcal{P}_\Gamma^{hc}$$
$$\langle X@\alpha\rangle^c = \lambda y.(X@(\langle\alpha\rangle^e@y)) \text{ where } X \in \mathcal{V}^{\mathcal{P}^{uc}\to\mathcal{P}^{hc}} \text{ and } \alpha \in \mathcal{P}_\Gamma^{uc}$$
$$\langle X@\beta\rangle^c = \lambda y.(X@(\langle\beta\rangle^c@y)) \text{ where } X \in \mathcal{V}^{\mathcal{P}^{hc}\to\mathcal{P}^{hc}} \text{ and } \beta \in \mathcal{P}_\Gamma^{hc}$$

Fig. 11: Additional rules for encoding patterns of unranked trees and hedges with context into patterns of ranked trees

Conclusion. We have shown that regular matching and inclusion for ranked tree patterns with context variables is Exp-complete with and without compression. The complexity goes down to P for linear compressed tree patterns in 3 of 4 cases. The same result holds 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).

References

1. D. Angluin. Finding patterns common to a set of strings. *JCSS*, 21:46–62, 1980.
2. I. Boneva, J. Niehren, and M. Sakho. Certain query answering on compressed string patterns: From streams to hyperstreams. *RP 2018, France*, volume 11123 of *LNCS*, pages 117–132. Springer, 2018.
3. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. TATA. `http://tata.gforge.inria.fr`, Oct. 2007.
4. A. Gascón, G. Godoy, and M. Schmidt-Schauß. Context matching for compressed terms. *LICS 2008, USA*, pages 93–102. IEEE CS, 2008.
5. A. Jez. Context unification is in PSPACE. *ICALP 2014, Denmark, Proceedings, Part II*, volume 8573 of *LNCS*, pages 244–255. Springer, 2014.
6. P. Labath and J. Niehren. A functional language for hyperstreaming XSLT. Technical report, INRIA Lille, 2013.
7. S. Maneth, A. O. Pereira, and H. Seidl. Transforming XML streams with references. *SPIRE 2015, UK*, volume 9309 of *LNCS*, pages 33–45. Springer, 2015.
8. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004.
9. H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3):424–437, 1990.
10. R. Loader. The Undecidability of Lambda-Definability. *Logic, Meaning and Computation. Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science)*, vol 305. Springer, Dordrecht.
11. T. Joly. Encoding of the halting problem into the monster type and applications. *TLCA'03, Spain*, pages 153–166. Springer-Verlag Berlin 2003.
12. M. Zaionc. Probabilistic approach to the lambda definability for fourth order types. *Electr. Notes Theor. Comput. Sci.*, 140:41–54, 2005.