Regular Matching and Inclusion on Compressed Tree Patterns with Context Variables

Iovka Boneva, Joachim Niehren, Momar Sakho

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.

Keywords: computational complexity, patterns, trees, tree languages and tree automata

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 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 [1] but in P for DFAs, with and without compression [5]. The more general problem of string unification is known to be PSPACE-complete [11].

Compressed patterns were called hyperstreams in [8]. 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 [2]. For string patterns, both problems were shown to be PSPACE-complete, for DFAs and NFAs, with and without compression. See Fig. 1 for an overview. When restricted to linear string patterns, the complexity

	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	PSPACE-c	EXP-c
Regular Inclusion	PSPACE-c	EXP-c

Fig. 5: Adding context variables.

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

Fig. 6: Linear restriction.

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 landscapes of regular matching and inclusion for tree patterns look 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), 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. For linear tree patterns, three of the four problems are in P except for the case of regular inclusion against NTAs. In [3], regular matching for tree patterns with neither compression nor context variables is studied as the *ground instance intersection problem*. Recently [12] studied the problem of matching compressed terms represented as singleton tree grammars, which is incomparable with regular matching that we study here.

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 [5], 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 [6]. 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.

The main contributions of the present paper are the complexity classes of regular matching and inclusion for compressed tree patterns with context variables,

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

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

which are summarized in Figs. 5 and 6. As shown in Figure 5, the problems are both PSPACE-complete when the tree automaton representing the language is deterministic, while they are EXP-complete in the case of a nondeterministic tree automaton. This complexities coincide with that of the *context inhabitation* problem for the two models.

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 XML streams with references [10]. They permit to generalize the notion of hyperstreams in [2] from strings to unranked trees.

Outline. We introduce tree patterns with context variables in Section 2. The inhabitation problem for Σ-algebras is defined in Section 3. The complexity of context inhabitation for tree automata is discussed in Section 4. Compressed tree patterns with context variables are introduced in Section 5 and then studied for regular matching and inclusion in Section 6. Due to space limitation, the discussion of the special cases of linear patterns and patterns with context variables, as well as the missing proofs, are not included in this extended abstract.

2 Tree Patterns with Context Variables

We consider the set of types $\mathbf{T} = \{tree, context\}$ with the type *tree* for trees and the type $context = tree \multimap tree$ for contexts. The latter linear type is inspired by linear logic and is different from the usual (nonlinear) function type $tree \to tree$.We assume sets $\mathcal{V}^{tree}$ of tree variables and $\mathcal{V}^{context}$ of context variables. The tree variables are ranged over by x, y, z and the context variables by X, Y. The set of all variables is $\mathcal{V} = \mathcal{V}^{tree} \cup \mathcal{V}^{context}$.

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 $\mathcal{T}_\Sigma$ 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 $x \in \mathcal{V}^{tree}$ that occurs exactly once in p. The set of all values of both types is $Val_\Sigma = \mathcal{T}_\Sigma \cup \mathcal{C}_\Sigma$.

The sets of all *tree patterns* $\mathcal{P}_\Sigma^{tree}$ and of all *context patterns* $\mathcal{P}_\Sigma^{context}$ are defined in Fig. 7. Note that both types of patterns may contain context variables. The set of all *patterns* is $\mathcal{P}_\Sigma = \mathcal{P}_\Sigma^{tree} \cup \mathcal{P}_\Sigma^{context}$. For a (tree or context) pattern π, its sets of free variables $fv(\pi)$ and of bound variables $bv(\pi)$ can be defined

as usual. 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* $t \in \mathcal{T}_\Sigma$ is a ground pattern of type *tree* and any *context* $C \in \mathcal{C}_\Sigma$ is a ground pattern of type *context*. 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 $p \in \mathcal{P}_\Sigma^{gr,tree}$ 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 $P \in \mathcal{P}_\Sigma^{gr,context}$ 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 $[\![\mathcal{P}_\Sigma^{gr,tree}]\!] = [\![\mathcal{T}_\Sigma]\!]$ is equal to $\mathcal{T}_\Sigma$ while $[\![\mathcal{P}_\Sigma^{gr,context}]\!] = [\![\mathcal{C}_\Sigma]\!]$ is a proper subset of the set of functions of type $\mathcal{T}_\Sigma \to \mathcal{T}_\Sigma$.

A substitution $\sigma : V \to \mathcal{P}_\Sigma^{gr}$ where $V \subseteq \mathcal{V}$ 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 $\sigma(p) \in \mathcal{P}_\Sigma^{gr,tree}$ 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:

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

Clearly, $Inst(p) \subseteq \mathcal{T}_\Sigma$. 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. $t \in Inst(p)$. Similarly, for any $P \in \mathcal{P}_\Sigma^{context}$, we can define the grounding $\sigma(P) \in \mathcal{P}_\Sigma^{gr,context}$. The set of instances $Inst(P)$ contains the semantics of all groundings of P. Clearly $Inst(P) \subseteq [\![\mathcal{C}_\Sigma]\!]$.

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 $\Delta = (dom^\Delta, .^\Delta)$ consists of a set $D = dom^\Delta$ 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 of a tree* $t = f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$ is the domain element $[\![t]\!]^\Delta = f^\Delta([\![t_1]\!]^\Delta, \ldots, [\![t_n]\!]^\Delta)$, 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 $d \in D$.

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

Fig. 8: Algebra evaluation of patterns.

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

The subset of 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 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. 8, we define for any tree pattern p and any 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 a ground pattern $\pi \in \mathcal{P}^{gr}_\Sigma$ in Δ does not depend on the variable assignment σ. Therefore we can write $[\![\pi]\!]^{\Delta}$ instead of $[\![\pi]\!]^{\Delta,\sigma}$. Clearly, algebra evaluation restricted to values is equal to algebra interpretation. Furthermore, note that $[\![\mathit{Val}_\Sigma]\!]^{\Delta} = [\![\mathcal{P}^{gr}_\Sigma]\!]^{\Delta}$ since any ground pattern in $\mathcal{P}^{gr}_\Sigma$ 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 $\sigma : V \to \mathcal{P}^{gr}_\Sigma$. Then $[\![.]\!]^{\Delta} \circ \sigma$ is a well-typed variable assignment into $[\![\mathcal{P}^{gr}_\Sigma]\!]^{\Delta} = [\![\mathit{Val}_\Sigma]\!]^{\Delta}$, such that $[\![\sigma(p)]\!]^{\Delta} = [\![p]\!]^{\Delta,[\![.]\!]^{\Delta}\circ\sigma}$ 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 $\{[\![p]\!]^{\mathcal{T}_\Sigma,[\![.]\!]^{\mathcal{T}_\Sigma}\circ\sigma} \mid \sigma : \mathit{fv}(p) \to \mathcal{P}^{gr}_\Sigma \text{ well-typed}\}$, 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 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 $f(q_1, \ldots, q_n) \to q$. The transition Σ-algebra of the 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-hand 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 automata classes. For any automaton class $\mathcal{A}$ and any signature Σ, tree inhabitation is the following problem:

Inhab$^{tree}_\Sigma(\mathcal{A})$. *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* $\text{INHAB}^{tree}_\Sigma(\text{NTA})$ *is* EXP*-complete, while its restriction* $\text{INHAB}^{tree}_\Sigma(\text{DTA})$ *to deterministic tree automata 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 $\text{INHAB}^{tree}_\Sigma(\text{NTA})$ 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 $\text{INHAB}^{tree}_\Sigma(\text{DTA})$ is in polynomial time.

We now have to show that $\text{INHAB}^{tree}_\Sigma(\text{NTA})$ is EXP-hard. This is achieved by reduction from the problem of non-emptiness of the intersection of a sequence of DTAs, which is well known to be EXP-complete [13]. 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 1 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 2. Since union homomorphisms are determined by their images on singletons, they can be represented by functions $s : Q \to 2^Q$. Conversely, every such 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 is a succinct descriptor of a union homomorphism:

Inhab$_\Sigma^{context}(\mathcal{A})$. *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 [9,7]. However, λ-definability for orders up to 3 is decidable [14], 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 1. Thus $\hat{s}$ is Δ-inhabited. □

Theorem 2. *Context inhabitation is* EXP*-complete for* NTA*s while it is* PSPACE*-complete for* DTA*s.*

Proof. The PSPACE-hardness of INHAB$_\Sigma^{context}$(DTA) can be shown by reduction from the problem of non-emptiness of the intersection of a finite number of Deterministic Finite Automata (DFAs), while a PSPACE algorithm can be obtained for it by reduction to this problem. We next present an exponential time algorithm for INHAB$_\Sigma^{context}$(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 1 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 $\text{INHAB}_\Sigma^{context}(\text{NTA})$ can be shown by reduction from the intersection problem of DTAs. The idea of the proof is similar to that of the PSPACE-hardness proof of DFA-inhabitation (see [2]), so we omit the details. □

5 Compressed Tree Patterns

We now recall compressed tree patterns with context variables that are defined by 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 $c\mathcal{P}_\Sigma^\tau$.*

For instance, consider the compressed tree pattern $G \in c\mathcal{P}_\Sigma^{tree}$ with the nonterminals $N = \{x, X, Y, Z, y\}$, with $S = x$ and with 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 the grammar G is $\emptyset$. What interests us instead is the tree pattern $pat(G) = (\lambda x.Z@a(x, y))@a((\lambda x.Z@a(x, y))@b, Y@c)$ that G represents in a compressed manner. By exhaustive β-reduction of $pat(G)$ we obtain the tree pattern with context variables $[\![pat(G)]\!] = Z@a(a(Z@a(b, y), Y@c), y)$.

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 $dom(R)$ and the bound variables on the right-hand sides of these rules.

In what follows we will identify any tree pattern $p \in \mathcal{P}_\Sigma^{tree}$ with the compressed tree pattern in $c\mathcal{P}_\Sigma^{tree}$ that has a single rule mapping a new start symbol to p. This compressed tree pattern has no compression. In this sense, $\mathcal{P}_\Sigma^{tree} \subseteq c\mathcal{P}_\Sigma^{tree}$. A compressed tree pattern G 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 $G = (N, \Sigma, R, S) \in c\mathcal{P}_\Sigma^{tree}$ 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; 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 $[\![pat(G_v)]\!]^{\Delta,\hat{\sigma}}$ 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 $\mathcal{G}$ of compressed tree patterns with context variables such as $\mathcal{P}^{tree}$ and $c\mathcal{P}^{tree}$.

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

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

Regular pattern matching $\textsc{Match}_\Sigma(\mathcal{G}, \mathcal{A})$. *Input: A compressed tree pattern $G \in \mathcal{G}_\Sigma$ and a tree automaton $A \in \mathcal{A}_\Sigma$.*
Output: 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*, $p \in \mathcal{P}^{tree}_\Sigma$ 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). $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ *is* PSPACE*-hard while* $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Nta})$ *is* EXP*-hard.*

Proof. We reduce $\textsc{Inhab}^{context}_\Sigma(\textsc{Dta})$ to $\textsc{Match}_\Sigma(\mathcal{P}^{tree}, \textsc{Dta})$ in polynomial time, then the PSPACE-hardness of the latter problem follows from Theorem 2. Let $A = (Q, \Sigma, F, \Delta)$ be a complete DTA– otherwise it is completed – and $s : Q \to 2^Q$ a function. We set $Q = \{q_1, \ldots, q_n\}$ and consider a new symbol $\# \notin \Sigma$ of arity n and a new state $q_\#$. Since A is deterministic and complete, s must map states to singletons in order to be inhabited. Let $\gamma_1, \ldots, \gamma_n \in Q$ be the states such that $s(q_1) = \{\gamma_1\}, \ldots, s(q_n) = \{\gamma_n\}$. From this we build a new DTA $\tilde{A} = (\tilde{Q}, \tilde{\Sigma}, \tilde{F}, \tilde{\Delta})$ where $\tilde{Q} = Q \cup \{q_\#\}$, $\tilde{\Sigma} = \Sigma \cup \{\#\} \cup Q$, $\tilde{F} = \{q_\#\}$ and $\tilde{\Delta} = \Delta \cup \{\#(\gamma_1, \ldots, \gamma_n) \to q_\#\}$. Let $X \in \mathcal{V}^{context}$ and $p = \#(X@q_1, \ldots, X@q_n) \in \mathcal{P}^{tree}_{\tilde{\Sigma}}$. The reduction is induced by the following claim, whose technical proof is based on Lemma 4 without any special tricks.

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

For the EXP-hardness of $\text{MATCH}_\Sigma(\mathcal{P}^{tree}, \text{NTA})$, it is sufficient to notice that the ground instance intersection problem [4], which is an EXP-hard problem in the case of NTAs, is equivalent to regular matching with only first-order variables for NTAs. The latter is a special case of regular matching with context variables.

Lemma 5 (Complementation). *Regular inclusion and matching are complementary problems for deterministic automata: For any class of compressed tree patterns* $\mathcal{G}$, $\text{INCL}_\Sigma(\mathcal{G}, \text{DTA})$ *and* $\text{coMATCH}_\Sigma(\mathcal{G}, \text{DTA})$ *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 it requires determinization.

Proposition 3 (Lower Bound Inclusion). $\text{INCL}_\Sigma(\mathcal{P}^{tree}, \text{DTA})$ *is* PSPACE*-hard while* $\text{INCL}_\Sigma(\mathcal{P}^{tree}, \text{NTA})$ *is* EXP*-hard.*

Proof. Lemma 5 states that $\text{INCL}_\Sigma(\mathcal{P}^{tree}, \text{DTA}) = \text{coMATCH}_\Sigma(\mathcal{P}^{tree}, \text{DTA})$ modulo P. By Proposition 2, $\text{MATCH}_\Sigma(\mathcal{P}^{tree}, \text{DTA})$ is PSPACE-hard and since PSPACE is closed by complement, $\text{coMATCH}_\Sigma(\mathcal{P}^{tree}, \text{DTA})$ is PSPACE-hard too. It then holds that $\text{INCL}_\Sigma(\mathcal{P}^{tree}, \text{DTA})$ is PSPACE-hard. For the EXP-hardness of $\text{INCL}_\Sigma(\mathcal{P}^{tree}, \text{NTA})$, we make a straightforward reduction from the problem of universality of NTAs, which is EXPTIME-hard. □

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

Proposition 4 (Upper Bounds). $\text{MATCH}_\Sigma(c\mathcal{P}^{tree}, \text{DTA})$ *and* $\text{INCL}_\Sigma(c\mathcal{P}^{tree}, \text{DTA})$ *are in* PSPACE*, while* $\text{MATCH}_\Sigma(c\mathcal{P}^{tree}, \text{NTA})$ *and* $\text{INCL}_\Sigma(c\mathcal{P}^{tree}, \text{NTA})$ *are in* EXP*.*

Proof. Let $G \in c\mathcal{P}^{tree}_\Sigma$ be a compressed tree pattern with start symbol $S \in \mathcal{V}^{tree}$ and set of nonterminals N, and $A = (Q, \Sigma, F, \Delta)$ be a tree automaton. According to Lemma 4, 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 iterates over all such σ, tests the inhabitation of $\widehat{\sigma}(v)$ for all $v \in N$, and checks that $\widehat{\sigma}(pat(G)) \cap F \neq \emptyset$. It is successful if the test succeeds for some σ. The number of iterations is at most $2^{|Q|^2 \cdot |fv(G)|}$, and can be done in a polynomial space. Moreover inhabitation can be tested in a polynomial space for if A is a DTA, while it requires at least time an exponential time if A is an NTA– Theorems 1 and 2. We can also compute $\widehat{\sigma}(pat(G))$ in polynomial time from A, G, and σ by Lemma 3. Thus the algorithm is in PSPACE for DTAs and EXP for NTAs. For $\text{INCL}_\Sigma(c\mathcal{P}^{tree}, \text{DTA})$ and $\text{INCL}_\Sigma(c\mathcal{P}^{tree}, \text{NTA})$, 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. □

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

Fig. 9: 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.

7 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. 9. 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 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. 9 how to encode hedge patterns into (ranked) context patterns over the signature $\Sigma = \Sigma^{(2)} \uplus \Sigma^{(0)}$ where $\Sigma^{(2)} = \Gamma$ and $\Sigma^{(0)} = \{\#\}$ for $\#$ is a fresh symbol not in Γ. 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 $\mathcal{V}^{context} = \mathcal{V}^u \uplus \mathcal{V}^h$, while the set $\mathcal{V}^{tree}$ of tree variables is left arbitrary. 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 6 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 $Inst^{\text{unr}}(p) = \{[\![\sigma(p)]\!] \mid \sigma : fv(p) \to Val_\Sigma$ well-typed and unranked$\}$ and similarly for $Inst^{\text{unr}}(P)$.

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

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

Let $c\mathcal{P}^u_\Gamma$ be the set of compressed unranked trees over Γ, defined in an analogous way as compressed tree patterns. For a class of automata $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$, the problem $\text{MATCH}_\Gamma(c\mathcal{P}^u, \mathcal{A})$ of regular matching of compressed unranked tree patterns takes as input an unranked tree pattern $H \in c\mathcal{P}^u$ and an automaton A in class $\mathcal{A}$, and outputs the truth value of whether $Inst^{\text{unr}}(\langle H\rangle^{tree}) \cap L(A) \neq \emptyset$. The problem INCL_Γ $(c\mathcal{P}^u, \mathcal{A})$ of regular inclusion for compressed patterns of unranked trees is defined in an analogous way. Note that using tree automata in the above definitions is not a restriction, as it is well known [3] that for any unranked tree language L recognizable by a hedge automaton, there exists a tree automaton that recognizes the encodings as ranked trees of the trees in L.

Proposition 5. *For any* $\mathcal{A} \in \{\text{DTA}, \text{NTA}\}$ *there exist polynomial time reductions from* $\text{MATCH}_\Gamma(c\mathcal{P}^u, \mathcal{A})$ *to* $\text{MATCH}_{\Sigma'}(c\mathcal{P}^{tree}, \mathcal{A})$ *and from* $\text{INCL}_\Gamma(c\mathcal{P}^u, \mathcal{A})$ *to* $Incl_{\Sigma'}(c\mathcal{P}^{tree}, \mathcal{A})$ *for some signature* Σ' *derived from* Σ.

Proof idea. The basic idea is to use Lemma 6, but we also need to constrain the variable assignments for the encoded patterns to be unranked. We illustrate how this works on an example for the case of regular matching. Consider the unranked tree pattern $H = a(Z) \in \mathcal{P}^u_\Gamma$, a language $\mathcal{L} \subseteq \mathcal{U}_\Gamma$ and an NTA A over Σ with $L(A) = \langle\mathcal{L}\rangle^{tree}$. From $\langle H\rangle^{tree} = a(Z@\#, \#)$, we build the compressed tree pattern $p_H = a(root_Z(Z@hole_Z(\#)), \#)$, where $root_Z$ and $hole_Z$ are new unary symbols. We also construct a NTA A' from A so that $Inst^{\text{unr}}(\langle H\rangle^{tree}) \cap L(A) \neq \emptyset$ iff $Inst(p_H) \cap L(A') \neq \emptyset$. Basically in p_H any variable Z is "enclosed" between the $root_Z$ and $hole_Z$ symbols, and A' tests that any context between $root_Z$ and $hole_Z$ is a correct encoding of an unranked hedge.

Theorem 3. $\text{MATCH}_\Gamma(c\mathcal{P}^u, \text{DTA})$ *and* $\text{INCL}_\Gamma(c\mathcal{P}^u, \text{DTA})$ *are* PSPACE*-complete while* $\text{MATCH}_\Gamma(c\mathcal{P}^u, \text{NTA})$ *and* $\text{INCL}_\Gamma(c\mathcal{P}^u, \text{NTA})$ *are* EXP*-complete.*

Proof. The upper bounds follow via the polynomial time reduction from Proposition 5 and the complexities in Proposition 4. 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 2 and 3. □

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).

Acknowledgments. We are grateful to Sylvain Salvati for pointing out and helping to solve difficulties. 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. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
2. I. Boneva, J. Niehren, and 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*, pages 117–132, 2018.
3. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available online since 1997: `http://tata.gforge.inria.fr`, Oct. 2007.
4. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications., 2007. http://www.grappa.univ-lille3.fr/tata.
5. A. Gascón, G. Godoy, and 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*, pages 93–102. IEEE Computer Society, 2008.
6. A. Jez. Context unification is in PSPACE. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2014.
7. T. Joly. Encoding of the halting problem into the monster type & applications. In M. Hofmann, editor, *Typed Lambda Calculi and Applications, 6th International Conference, TLCA 2003, Valencia, Spain, June 10-12, 2003, Proceedings.*, volume 2701 of *Lecture Notes in Computer Science*, pages 153–166. Springer, 2003.
8. P. Labath and J. Niehren. A functional language for hyperstreaming XSLT. Technical report, INRIA Lille, 2013.
9. R. Loader. The undecidability of λ-definability. In Z. M. Anderson C.A., editor, *Logic, Meaning and Computation*, volume 305. Springer, 2001.
10. S. Maneth, A. O. Pereira, and H. Seidl. Transforming XML streams with references. In C. S. Iliopoulos, S. J. Puglisi, and E. Yilmaz, editors, *String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings*, volume 9309 of *Lecture Notes in Computer Science*, pages 33–45. Springer, 2015.
11. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004.
12. M. Schmidt-Schauß. Linear pattern matching of compressed terms and polynomial rewriting. *Mathematical Structures in Computer Science*, 28(8):1415–1450, 2018.
13. H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3):424–437, 1990.
14. M. Zaionc. Probabilistic approach to the lambda definability for fourth order types. *Electr. Notes Theor. Comput. Sci.*, 140:41–54, 2005.