

# Ramsey Quantifiers over Automatic Structures: Complexity and Applications to Verification

Pascal Bergsträßer

Department of Computer Science, TU Kaiserslautern  
Kaiserslautern, Germany

Anthony W. Lin

Department of Computer Science, TU Kaiserslautern  
Max Planck Institute for Software Systems (MPI-SWS)  
Kaiserslautern, Germany

Moses Ganardi

Max Planck Institute for Software Systems (MPI-SWS)  
Kaiserslautern, Germany

Georg Zetsche

Max Planck Institute for Software Systems (MPI-SWS)  
Kaiserslautern, Germany

## Abstract

Automatic structures are infinite structures that are finitely represented by synchronized finite-state automata. This paper concerns specifically automatic structures over finite words and trees (ranked/unranked). We investigate the “directed version” of Ramsey quantifiers, which express the existence of an infinite directed clique. This subsumes the standard “undirected version” of Ramsey quantifiers. Interesting connections between Ramsey quantifiers and two problems in verification are firstly observed: (1) reachability with Büchi and generalized Büchi conditions in regular model checking can be seen as Ramsey quantification over transitive automatic graphs (i.e., whose edge relations are transitive), (2) checking monadic decomposability (a.k.a. recognizability) of automatic relations can be viewed as Ramsey quantification over co-transitive automatic graphs (i.e., the complements of whose edge relations are transitive). We provide a comprehensive complexity landscape of Ramsey quantifiers in these three cases (general, transitive, co-transitive), all between NL and EXP. In turn, this yields a wealth of new results with precise complexity, e.g., verification of subtree/flat prefix rewriting, as well as monadic decomposability over tree-automatic relations. We also obtain substantially simpler proofs, e.g., for NL complexity for monadic decomposability over word-automatic relations (given by DFAs).

**CCS Concepts:** • Theory of computation → Logic and verification; Regular languages; Tree languages.

**Keywords:** Ramsey quantifier, automatic structures, recurrent reachability, monadic decomposability

## 1 Introduction

Automatic structures are infinite structures whose domains are *regular sets* (represented by finite automata over finite words/trees) and whose relations are *regular relations* (represented by synchronized finite word/tree automata) [4, 5].

They have been intensively studied in the logic and automata community, and have been also applied to infinite-state verification, especially the area of regular model checking. In this paper, we are interested in algorithmic aspects of the so-called *Ramsey quantifiers*, which state the existence of infinite cliques. We pinpoint a surprising connection between Ramsey quantifiers and the verification of liveness properties in regular model checking, as well as monadic decomposability of regular relations (a.k.a. finite recognizability). Exploiting this connection, we establish a comprehensive picture of the computational complexity landscape of Ramsey quantifiers over automatic structures, as well the aforementioned verification problems. We first discuss the state-of-the-art of these seemingly disconnected problems.

**Ramsey Quantifiers over Automatic Structures.** Blumensath and Grädel were the first to systematically study automatic structures [4, 5]. A fundamental fact is that, given a first-order formula  $\varphi(\mathbf{x})$  and a word/tree automatic structure  $\mathfrak{A}$  (with domain  $A$ ), one can effectively compute a synchronized word/tree automaton representing the set  $[[\varphi]]_{\mathfrak{A}} = \{\mathbf{a} \in A^{|\mathbf{x}|} : \mathfrak{A} \models \varphi(\mathbf{a})\}$  of solutions of  $\varphi$ . In other words, regular relations are *effectively closed under all first-order operations*. Consequently, first-order (FO) model checking over automatic structures is decidable.

In the seminal paper [4] on automatic structures, it was already observed that FO can be enriched with the quantifier “there exist infinitely many” — i.e.,  $\exists^\infty x \varphi(x, \mathbf{y})$ , which is true iff there exist infinitely many  $x$  such that  $\varphi(x, \mathbf{y})$  — while preserving the above effective closure property and decidability of model checking. In fact, assuming nondeterministic automata as finite representations of  $[[\varphi]]_{\mathfrak{A}}$ , one can compute  $[[\exists^\infty x \varphi]]_{\mathfrak{A}}$  in polynomial-time. A few years later, in the case of word automatic structures, Rubin [26] studied Ramsey quantifiers, which generalize  $\exists^\infty$  by enforcing that these infinitely many elements form an infinite undirected clique, and showed that Ramsey quantifiers preserve regularity as well, meaning FO extended with Ramsey quantifiers is still decidable. Upon closer inspection, Rubin’s construction runs in doubly exponential time. *Is this optimal? Does the same extend to tree-automatic structures?*

**Liveness in Regular Model Checking.** *Regular model checking (RMC)* is a generic verification framework that exploits regular languages and relations (e.g., over finite/ $\omega$ -words or trees) as symbolic representations of infinite systems [1, 2, 20]. Various flavors of automata and transducers for representing the transition relations are employed, e.g., word/tree automatic relations (or subsets thereof),  $\omega$ -automatic relations (or subsets thereof), and rational relations. Since safety and reachability are undecidable in RMC (e.g., over automatic graphs), one focus of RMC has been to develop acceleration/widening techniques, which are semi-algorithms for computing reachability sets (i.e.,  $\text{post}^*(S)$ ) and reachability relations (i.e., transitive closure  $R^*$  of the edge relation in the graph), that may terminate on many interesting cases. Some of these semi-algorithms have general completeness and termination guarantee, e.g., bounded local-depth acceleration for automatic relations are guaranteed to compute reachability relations for pushdown systems (PDS) and ground-tree rewrite systems (GTRS) [1, 19], while flattable acceleration for Presburger-definable relations is guaranteed to compute reachability relations for reversal-bounded counter systems and 2-dimensional vector addition systems with states [16].

Reachability sets/relations can be directly used to solve safety. The challenge of verifying liveness is the necessity to deal with genuinely infinite paths (with no repeated configurations). To and Libkin [29, 30] showed that one can decide liveness (in the form of recurrent reachability) over word- and tree-automatic graphs, when the transitive closure  $R^*$  of the edge relation is additionally supplied (e.g., by the aforementioned acceleration methods or otherwise). The algorithm runs in time polynomial in the size of the problem, with  $R^*$  supplied as part of inputs. Their technique uses a kind of “staircase argument” combined with Ramsey’s Theorem to construct a Büchi automaton that represents some witnessing infinite runs.

**Monadic Decomposability and Recognizability.** A classic task in the theory of finite-state transductions is the problem to decide whether a given regular relation  $R \subseteq (\Sigma^*)^k$  is *recognizable*, i.e., if it can be expressed as a finite union of cartesian products of regular languages (in symbols:  $R = \bigcup_{i=1}^n L_{i,1} \times \dots \times L_{i,k}$  for some  $n \in \mathbb{N}$  and regular sets  $L_{i,j} \subseteq \Sigma^*$ ). In the formal verification terminology [33], such a relation is said to be *monadically decomposable*, i.e., that it can be expressed as a Boolean combination of monadic predicates. The first important result was by Stearns [28] and Valiant [31]: Their algorithms for checking regularity of deterministic pushdown automata imply that given deterministic rational relation—i.e., a relation  $R \subseteq \Sigma^* \times \Sigma^*$  recognized by *deterministic asynchronous automata*, which is strictly more general than binary automatic relations—can be checked to be recognizable in doubly exponential time.

This decidability was extended to general  $k$ -ary deterministic rational relations by Carton et al. [6], which yields decidability as well for the subclass of automatic relations. As noted by Löding and Spinrath [23], the complexity of the algorithm for automatic relations in [6] runs in doubly exponential time. Using their new polynomial-time algorithm for checking regularity for deterministic visibly pushdown automata, Löding and Spinrath showed that this could be improved to single exponential time for binary automatic relations. The complexity for automatic relations was fully settled by Barceló et al. [3] by showing that this problem is NL-complete (resp. PSPACE-complete) when  $R$  is presented as a deterministic (resp. nondeterministic) automaton. The proof technique in [3] is an extremely intricate refinement and analysis of the staircase argument used by To and Libkin [29] for recurrent reachability for automatic relations.

**Contributions.** Our account of the state-of-the-art of the aforementioned three research directions seems to suggest that there might be some connections between them. To what extent are they connected? Is there a more fundamental notion that unifies them? These questions are hitherto open, but as we shall see in this paper the answer is a resounding yes. We pinpoint that the *directed Ramsey quantifiers*—which ask for the existence of infinite directed cliques (instead of infinite undirected cliques as in [26])—is a fundamental concept that underlies the above three problems, and lets us study them under the same umbrella, while inferring the optimal complexity and even new results. On the one hand, the directed Ramsey quantifiers subsume the standard Ramsey quantifiers.

On the other hand, recurrent reachability over automatic graphs [29] can be seen as a Ramsey quantifier over a transitive binary relation, whereas monadic decomposability over automatic relations [3, 6, 23] can be construed as a Ramsey quantifier over co-transitive binary relations. Our results are summarized in Table 1.

Firstly, from the proof by Barceló et al. [3], it is possible to infer that the Ramsey quantifier can be evaluated on regular relations in NL, which substantially improves the doubly exponential-time algorithm of Rubin [26]. Unfortunately, their argument relies on an intricate Ramsey argument on the transition monoid of the automaton. Our contribution is a substantially simpler argument that avoids the use of the transition monoid altogether, which we show to generalize to the case of tree-regular relations (which is not the case with the proof of [3]). More precisely, our approach divides the proof for regular relations into two steps: (i) First, we argue that one can assume infinite cliques witnessed by accepting runs that form a *comb of combs*. (ii) Then, we argue that the runs can be “merged” together so that it can be witnessed by a single run of a polynomial-size Büchi automaton. This way, we obtain the same complexity as [3].

For tree-regular relations we can easily extend step (i). The comb of combs structure of the accepting runs can be witnessed by an alternating Büchi tree automaton, which yields the complexity of EXP for the Ramsey quantifier on tree-regular relations. However, step (ii) is provably impossible over tree-regular relations, since as we show, the infinite clique problem is EXP-hard. For the special cases of transitive and co-transitive relations we need further separate arguments that enable us to evaluate the Ramsey quantifier in P. The case for transitive relations can be inferred from the proof in [29], but not so for the co-transitive case.

Finally, we apply our results to decidability and complexity of recurrent reachability with generalized Büchi conditions, and automatic structures over unranked trees. We show, for example, decidability (in fact in polynomial-time) of recurrent reachability of subtree/flat prefix rewriting, answering an open question by Löding and Spelten [22] and decidability (in fact, EXP-completeness) of recurrent reachability with generalized Büchi conditions of ground tree rewrite systems, answering an open question by Löding [21].

**Organization.** We provide a more detailed summary of our main results in Section 2. We fix notation and basic terminologies in Section 3. We then start with the word case in Section 4 and proceed to the tree case in Section 5. Applications and generalizations to unranked trees are given in, respectively, Section 6 and Section 7.

## 2 Detailed summary of main results

To improve readability, we provide a detailed summary of our main results in this section before we take a deeper dive into the proofs. Unless otherwise specified, the completeness results mentioned in this section (and Table 1) hold for NFAs and DFAs in the word case and NTAs, D↑TAs, and D↓TAs in the tree case. We define the directed Ramsey quantifier:

**Definition 2.1.** Let  $\mathfrak{A}$  be a structure with domain  $A$ . The Ramsey quantifier  $\exists^{\text{ram}}$  over an  $\mathfrak{A}$ -formula  $\varphi$  with  $k+2$  free variables is defined for all  $\mathbf{c} \in A^k$  by  $\mathfrak{A} \models \exists^{\text{ram}} x, y: \varphi(x, y, \mathbf{c})$  if and only if there is an infinite sequence  $(a_i)_{i \geq 1}$  of pairwise distinct elements  $a_i \in A$  so that  $\mathfrak{A} \models \varphi(a_i, a_j, \mathbf{c})$  for all  $1 \leq i < j$ .

We deviate from the definition of the Ramsey quantifier found in the literature, see [11], requiring  $\mathfrak{A} \models \varphi(a_i, a_j, \mathbf{c})$  for all  $i \neq j$ , in the definition above. Over (tree-)regular relations the two quantifier definitions can be simulated by each other, see Appendix A. Furthermore, there are also higher-dimensional versions  $\exists^{d\text{-ram}}$  of the Ramsey quantifier, which will not be considered in this paper.

**Evaluating Ramsey quantifiers.** If  $R$  is a binary (tree-)regular relation, then evaluating  $\exists^{\text{ram}} x, y: R(x, y)$  is the

problem of checking whether  $R$  contains an *infinite (directed) clique*, i.e., an infinite sequence  $(a_i)_{i \geq 1}$  of distinct elements of  $A$  such that  $(a_i, a_j) \in R$  for all  $1 \leq i < j$ . It follows from [3] that the infinite clique problem over word-regular relations is NL-complete. We provide a much simpler proof by considering a slightly more general setting. Instead of the infinite clique problem we consider the evaluation of the Ramsey quantifier on a  $(k+2)$ -ary (tree-)regular relation  $R \subseteq A^{k+2}$ , i.e., compute an automaton for  $[[\exists^{\text{ram}} x, y: R(x, y, \mathbf{c})]] = \{\mathbf{c} \in A^k \mid \exists^{\text{ram}} x, y: R(x, y, \mathbf{c})\}$ .

**Theorem 2.2.** *Given a regular relation  $R \subseteq (\Sigma^*)^{k+2}$  by an NFA  $\mathcal{A}^1$ , one can construct in logspace an NFA for the relation  $[[\exists^{\text{ram}} x, y: R(x, y, \mathbf{z})]]$ . In particular, the infinite clique problem over regular relations is in NL.*

We show that the complexity of the infinite clique problem increases from NL to EXP when considered over tree-regular relations given by NTAs or D↓TAs. Let  $\mathcal{T}_\Sigma$  denote the set of ranked trees over alphabet  $\Sigma$ .

**Theorem 2.3.** *The infinite clique problem over tree-regular relations  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  is EXP-complete if  $R$  is given as NTA or D↓TA, and P-complete if  $R$  is given as D↑TA.*

For the exponential lower bound, we present a reduction from intersection nonemptiness for NTAs and D↓TAs. This is surprising, because an analogue reduction in the word case does not exist: This would yield a PSPACE lower bound for the infinite clique problem over words, but the latter belongs to NL.

For the exponential upper bound of Theorem 2.3, we prove the tree analogue of Theorem 2.2. It even holds when the relation  $R$  is given as an alternating tree automaton (ATA), which allows us to apply it to recurrent reachability with generalized Büchi condition.

**Theorem 2.4.** *Given an ATA (D↑TA)  $\mathcal{A}$  for a tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^{k+2}$ , one can construct in exponential (polynomial) time an NTA for the relation  $[[\exists^{\text{ram}} x, y: R(x, y, \mathbf{z})]]$ .*

If we make further assumptions on the relation  $R$ , we obtain a better complexity for NTAs. We say that a  $(k+2)$ -ary relation  $R$  over  $A$  is *transitive* if the binary relation  $\{(a, b) \mid (a, b, \mathbf{c}) \in R\}$  is transitive for all  $\mathbf{c} \in A^k$ .

**Theorem 2.5.** *Given an NTA  $\mathcal{A}$  for a transitive tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^{k+2}$ , one can construct in polynomial time an NTA for the relation  $[[\exists^{\text{ram}} x, y: R(x, y, \mathbf{z})]]$ . In particular, the infinite clique problem over transitive tree-regular relations is in P.*

A binary relation  $R \subseteq A \times A$  is *co-transitive* if its complement  $(A \times A) \setminus R$  is a transitive relation.

**Theorem 2.6.** *The infinite clique problem over co-transitive tree-regular relations  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  given as NTA is in P.*

<sup>1</sup>In this and the following theorems, the parameter  $k$  is part of the input.

	regular relations	tree-regular relations
Automaton construction for Ramsey quantifier	logspace	exponential time poly-time for (co)transitive relations
Recurrent reachability & infinite clique	NL-complete*	EXP-complete for NTA, $D\downarrow TA$ P-complete for transitive relations* or $D\uparrow TA$
Recurrent reachability with generalized B���� condition	PSPACE-complete	EXP-complete
Monadic decomposability	NL-complete for DFA* PSPACE-complete for NFA*	P-complete for $D\uparrow TA$ , $D\downarrow TA$ EXP-complete for NTA

**Table 1.** Complexity results. Those marked with \* were known, but we provide simpler proofs. The other results are new.

In Section 7 we show by a reduction that the Ramsey quantifier can be evaluated over unranked tree-regular relations with the same complexity as in the ranked case.

**Recurrent reachability.** Since reachability in automatic graphs is in general undecidable [5], we will instead use *transitive paths*, i.e., infinite sequences  $(a_i)_{i \geq 1}$  with  $(a_i, a_j) \in R$  for all  $1 \leq i < j$ . Given sets  $L_1, \dots, L_k \subseteq A$  we write  $Rec(L_1, \dots, L_k)[R]$  for the set of all initial vertices  $a_1$  of transitive paths  $(a_i)_{i \geq 1}$  that visit each set  $L_j$  infinitely often. *Recurrent reachability with generalized B     condition* is the problem of testing  $a_1 \in Rec(L_1, \dots, L_k)[R]$  for a given (tree-)regular relation  $R \subseteq A \times A$ , (tree-)regular languages  $L_1, \dots, L_k \subseteq A$ , and an initial element  $a_1 \in A$ . If  $k = 1$  this problem is simply called *recurrent reachability*.

Since the infinite clique problem and recurrent reachability are logspace equivalent (Proposition 6.1), we obtain:

**Corollary 2.7.** *Recurrent reachability is NL-complete over regular relations. It is EXP-complete over tree-regular relations given by NTAs or  $D\downarrow TAs$ , and P-complete if the tree-regular relations are transitive or given by  $D\uparrow TAs$ .*

We also apply Theorems 2.2 and 2.4 to obtain tight upper bounds for recurrent reachability with generalized B     condition. The lower bounds result from a reduction from intersection nonemptiness.

**Theorem 2.8.** *Recurrent reachability with generalized B     condition is PSPACE-complete over regular relations, and EXP-complete over tree-regular relations.*

**Monadic Decomposability.** Recall that a relation  $R \subseteq A^k$  is *monadically decomposable* if it is of the form  $\bigcup_{i=1}^n A_{i,1} \times \dots \times A_{i,k}$  for some  $n \in \mathbb{N}$  and (tree-)regular languages  $A_{i,j}$ . The traditional approach to deciding monadic decomposability [3, 6, 9, 17, 23] is to associate with  $R$  certain equivalence relations  $\sim_j$  for  $1 \leq j \leq k$  such that  $R$  is monadically decomposable if and only if each  $\sim_j$  has finite index. An equivalence relation has infinite index if and only if there exist infinitely many elements that are pairwise in different equivalence classes which is witnessed by an infinite clique

in the complement relation. Therefore, monadic decomposability amounts to checking that  $\sim_j$ 's complement  $\not\sim_j$  does not have an infinite clique for any  $j$ . If  $R$  is given by a DFA (resp. NFA), then one can construct an NFA for each  $\not\sim_j$  in logspace (resp. in PSPACE) and thus Theorem 2.2 yields a tight upper bound:

**Corollary 2.9.** *Given a regular relation  $R \subseteq (\Sigma^*)^k$  by a DFA (resp. NFA), it is NL-complete (resp. PSPACE-complete) to decide whether  $R$  is monadically decomposable.*

While this approach yields optimal complexity for words, this is, unexpectedly, not the case for trees. For a tree-regular relation given as  $D\downarrow TA$  or  $D\uparrow TA$  (resp. NTA), one can also construct an NTA for each  $\not\sim_j$  in logspace (resp. PSPACE). Then, applying Theorem 2.4 would yield an EXP (resp. 2EXP) algorithm. However, perhaps surprisingly, monadic decomposability for trees has much lower complexity:

**Corollary 2.10.** *Given a tree-regular relation  $R \subseteq \mathcal{T}_\Sigma^k$  by a  $D\downarrow TA$  or  $D\uparrow TA$  (resp. NTA), it is P-complete (resp. EXP-complete) to decide whether  $R$  is monadically decomposable.*

To get the P (resp. EXP) algorithm, we exploit co-transitivity of each  $\not\sim_j$  and apply Theorem 2.6 instead of Theorem 2.4. This shows the importance of the co-transitivity notion: In the word case, monadic decomposability requires only the generic clique detection, but the tree case is more nuanced—we need one algorithm for the general case and a specialized algorithm for co-transitive relations.

### 3 Preliminaries

We assume familiarity with the basic models of (non)deterministic and alternating finite automata on words and trees as well as with standard complexity classes (e.g., NL, P, PSPACE, EXP). We refer the reader to the textbooks [7, 13] for more details. We often abbreviate a finite or infinite sequence of elements  $a_1, a_2, \dots$  by a boldface letter  $\mathbf{a}$ .

**Trees.** A *tree domain* is a nonempty set  $D \subseteq \mathbb{N}^*$  such that (i)  $D$  is prefix closed, i.e.,  $uv \in D$  implies  $u \in D$ , (ii) for all



$v \in \mathbb{N}^*$  and  $j \leq i$  if  $vi \in D$ , then  $vj \in D$ , and (iii) each node  $v \in D$  has only finitely many children  $vi \in D$  where  $i \in \mathbb{N}$ . An *unranked tree* over an alphabet  $\Sigma$  is a function  $t: \text{dom}(t) \rightarrow \Sigma$  where  $\text{dom}(t)$  is a finite tree domain. A *ranked alphabet* is a finite alphabet  $\Sigma$  where every symbol  $a \in \Sigma$  has a rank  $\text{rk}(a) \in \mathbb{N}$ . A *ranked tree* is an unranked tree  $t$  such that every node  $v \in \text{dom}(t)$  has  $\text{rk}(t(v))$  many children. We denote the set of all (un)ranked trees over  $\Sigma$  by  $\mathcal{T}_\Sigma$  and  $\mathcal{U}_\Sigma$ .

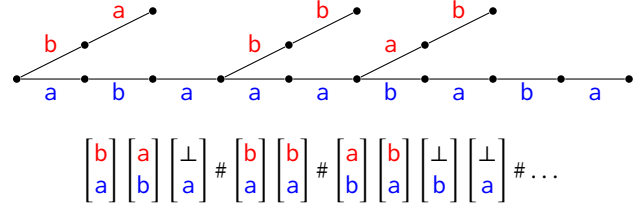
Let  $x \notin \Sigma$  be a *variable*. The set  $C_\Sigma$  of all *contexts* over  $\Sigma$  contains all unranked trees over  $\Sigma \cup \{x\}$  such that every node  $u \in \text{dom}(t)$  with  $t(u) = x$  is a leaf, called *hole*. We partition  $\text{dom}(t) = \text{nodes}(t) \cup \text{holes}(t)$  into nodes and holes. The size of a context  $t$  is  $|\text{nodes}(t)|$ . For contexts  $s, t_1, \dots, t_n$  with  $|\text{holes}(s)| = n$  we denote by  $s[t_1, \dots, t_n]$  the context obtained by replacing the  $i$ -th hole in lexicographic order by  $t_i$ . For two contexts  $s_1, s_2 \in C_\Sigma$ , we call  $s_1$  a *prefix* of  $s_2$ , denoted by  $s_1 \leq_p s_2$ , if  $s_1[t_1, \dots, t_n] = s_2$  for some contexts  $t_1, \dots, t_n$ . If  $n > 0$  and each context  $t_i$  has size at least one, then  $s_1$  is a *proper prefix* of  $s_2$ , denoted by  $s_1 <_p s_2$ .

We define an *infinite unranked tree* and an *infinite ranked tree* as in the finite case but with infinite domains. We denote the set of all finite and infinite unranked trees over the alphabet  $\Sigma$  by  $\mathcal{U}_\Sigma^\infty$  and the set of all finite and infinite ranked trees over  $\Sigma$  by  $\mathcal{T}_\Sigma^\infty$ .

**Regular and tree-regular languages.** A *nondeterministic finite automaton* (NFA) over the alphabet  $\Sigma$  is a tuple  $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$  where  $Q$  is a finite set of states,  $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$  is a transition relation,  $q_0 \in Q$  is an initial state, and  $F \subseteq Q$  is a set of final states. We denote by  $L(\mathcal{A}) \subseteq \Sigma^*$  the *regular language* recognized by  $\mathcal{A}$ . In our algorithms, the alphabet  $\Sigma$  is *not* part of the representation of an automaton. Instead, we will always work with the subalphabet of all symbols occurring in the transitions. This will be important later when the implicitly given alphabet is significantly smaller.

A *nondeterministic (top-down) tree automaton* (NTA) over the ranked alphabet  $\Sigma$  is a tuple  $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$  where  $Q$  is a finite set of states,  $q_0 \in Q$  is an initial state, and  $\Delta \subseteq \bigcup_{a \in \Sigma} Q \times \{a\} \times Q^{\text{rk}(a)}$  is a transition relation. A *run* of  $\mathcal{A}$  on a tree  $t \in \mathcal{T}_\Sigma$  is a tree  $\rho \in \mathcal{T}_Q$  with  $\text{dom}(\rho) = \text{dom}(t)$  such that  $\rho(\varepsilon) = q_0$  and  $(\rho(u), t(u), \rho(u_1), \dots, \rho(u_r)) \in \Delta$  for all nodes  $u \in \text{dom}(\rho)$  with  $\text{rk}(t(u)) = r$ . As before,  $L(\mathcal{A})$  is the set of trees recognized by  $\mathcal{A}$ , i.e., the set of all trees  $t$  such that there exists a run of  $\mathcal{A}$  on  $t$ . A set of trees is called *tree-regular* if there is an NTA that recognizes it. We will also use the notions of *deterministic finite automata* (DFA), *deterministic bottom-up* (D $\uparrow$ TA), and *deterministic top-down tree automata* (D $\downarrow$ TA). Moreover, *alternating automata* will be formally introduced in later sections.

**Regular and tree-regular relations.** Let  $\Sigma$  be a finite alphabet and let  $\Sigma_\perp = \Sigma \cup \{\perp\}$  where  $\perp \notin \Sigma$  is a fresh symbol. For words  $w_1, \dots, w_k \in \Sigma^*$  with  $w_i = a_{i,1} \dots a_{i,n_i}$



**Figure 1.** An example word comb  $ba, ababb, abaaaab, \dots$  and its encoding as an infinite word.

and  $n := \max\{n_i \mid 1 \leq i \leq k\}$  we define their convolution

$$w_1 \otimes \dots \otimes w_k := \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} := \begin{pmatrix} a'_{1,1} \\ \vdots \\ a'_{k,1} \end{pmatrix} \dots \begin{pmatrix} a'_{1,n} \\ \vdots \\ a'_{k,n} \end{pmatrix} \in (\Sigma_\perp^k)^*$$

where  $a'_{i,j} = a_{i,j}$  if  $j \leq n_i$  and  $a'_{i,j} = \perp$  otherwise. A relation  $R \subseteq (\Sigma^*)^k$  is *recognized* by an NFA  $\mathcal{A}$  if  $L(\mathcal{A}) = \{w_1 \otimes \dots \otimes w_k \mid (w_1, \dots, w_k) \in R\}$ . In that case we call  $R$  *regular*.

We extend the definitions to tree-regular relations. For an alphabet  $\Sigma$  we set again  $\Sigma_\perp = \Sigma \cup \{\perp\}$  where  $\perp \notin \Sigma$  is a fresh symbol. Let  $\varepsilon$  be the *empty tree* with  $\text{dom}(\varepsilon) := \emptyset$ . Given  $k$  trees  $t_1, \dots, t_k \in \mathcal{U}_\Sigma \cup \{\varepsilon\}$  we define their convolution  $t = t_1 \otimes \dots \otimes t_k \in \mathcal{U}_{\Sigma_\perp^k} \cup \{\varepsilon\}$  with  $\text{dom}(t) = \bigcup_{i=1}^k \text{dom}(t_i)$  and  $t(v) = (t'_1(v), \dots, t'_k(v))$  where  $t'_i(v) = t_i(v)$  if  $v \in \text{dom}(t_i)$  and  $t'_i(v) = \perp$  otherwise. Observe that the degree of a node  $v$  in  $t_1 \otimes \dots \otimes t_k$  is the maximum degree of  $v$  in a tree  $t_i$  such that  $v \in \text{dom}(t_i)$ . Similar to the word case, we also write the convolution of trees as a column vector. If all  $t_i$  are ranked trees, then also  $t$  is a ranked tree with  $\text{rk}(a_1, \dots, a_k) := \max\{\text{rk}(a_i) \mid 1 \leq i \leq k\}$  for all  $(a_1, \dots, a_k) \in \Sigma_\perp^k$  where  $\text{rk}(\perp) := 0$ . A relation  $R \subseteq \mathcal{T}_\Sigma^k$  is *recognized* by an NTA  $\mathcal{A}$  if the tree language  $\{t_1 \otimes \dots \otimes t_k \mid (t_1, \dots, t_k) \in R\}$  is recognized by  $\mathcal{A}$ . In that case we call  $R$  *tree-regular*.

Regular and tree-regular relations are effectively closed under first-order operations (Boolean operations and projections). A relational structure  $\mathfrak{A}$  is *automatic (tree-automatic)* if its universe and all its relations are regular (tree-regular).

## 4 Word-automatic structures

We first consider Ramsey quantifiers over word-regular relations. We show that if  $R \subseteq (\Sigma^*)^{k+2}$  is a regular relation, then an automaton for  $\{c \mid \exists^{\text{ram}} x, y: R(x, y, c)\}$  can be constructed in logarithmic space (Theorem 2.2).

**Word combs.** The first step is to observe that, when looking for infinite cliques in  $R$ , one can restrict to combs: An infinite sequence  $v$  of words is called a *comb* if there exist infinite sequences  $\alpha$  and  $\beta$  of words with  $v_i = \beta_1 \dots \beta_{i-1} \alpha_i$  and  $1 \leq |\alpha_i| \leq |\beta_i|$  for all  $i \geq 1$ . The pair  $(\alpha, \beta)$  is called a *generator* of  $v$ . We remark that the choice of the generator is not unique. Any infinite subsequence of a comb is again

a comb. In fact, the following lemma is well-known, see [15, Lemma 5.1].

In contrast to arbitrary infinite sequences of words, combs can be encoded naturally by infinite words. If  $(\alpha, \beta)$  is a generator we call the infinite word

an encoding of  $(\alpha, \beta)$ , or also an encoding of  $v$ .

Consider a comb  $v$  with generator  $(\alpha, \beta)$ . First observe that the convolutions  $v_i \otimes v_j$  can be written as

and can hence be arranged in a trie displayed in Figure 2, that we call *comb of combs*. The next insight is that we can ensure that the accepting runs  $\rho(v_i, v_j)$  on the convolutions  $v_i \otimes v_j$  match this comb of combs structure, after replacing  $v$  by an infinite subsequence. Roughly speaking, the runs look like as if the automaton for  $R$  would be *deterministic*. For example, all runs  $\rho(v_1, v_j)$  for  $j > 1$  share a common prefix which is a run on  $\alpha_1 \otimes \beta_1$ , and all runs  $\rho(v_i, v_j)$  for  $1 < i < j$  share a common prefix which is a run on  $\beta_1 \otimes \beta_1$ .

We define a *decomposition* of a word  $w \in \Sigma^*$  as  $w = u_1 \dots u_n$  where  $u_i \in \Sigma^*$ . The decomposition in Equation (1) is called the  $(\alpha, \beta)$ -decomposition of  $v_i \otimes v_j$ . We say that a decomposition of a run  $\rho = \rho_1 \dots \rho_n$  of an NFA is *compatible* with a decomposition  $w = u_1 \dots u_n$  of a word if  $\rho_i$  is a run on  $u_i$  for all  $i \in [1, n]$ .

**Lemma 4.2.** *Let  $\mathbf{w}$  be a comb generated by  $(\gamma, \delta)$  that forms an infinite clique in a regular relation  $R \subseteq \Sigma^* \times \Sigma^*$  given as an NFA  $\mathcal{A}$ . There exist a coarsening  $(\alpha, \beta)$  of  $(\gamma, \delta)$  that generates a comb  $\mathbf{v}$ , accepting runs  $\rho(v_i, v_j)$  of  $\mathcal{A}$  on  $v_i \otimes v_j$ , and runs  $\kappa_i, \lambda_i, \mu_i, \nu_{i,j}$  such that*

$$\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} v_{i,j}$$

is a decomposition compatible with the  $(\alpha, \beta)$ -decomposition of  $v_i \otimes v_j$  for all  $i < j$ .

*Proof.* Let  $\mathbf{w}$  be a comb generated by  $(\alpha, \beta)$  that forms an infinite clique in  $R$  and  $\rho(w_i, w_j)$  be an accepting run of  $\mathcal{A}$  on  $w_i \otimes w_j$  for all  $1 \leq i < j$ . We establish the run structure as illustrated in Figure 3 column-wise.

Assume that we already defined  $(\kappa_i)_{i < n}$ ,  $(\lambda_i)_{i < n}$ ,  $(\mu_{i,j})_{i < j < n}$ , and  $(v_{i,j})_{i < j < n}$  for some  $n \geq 1$  such that

$$\begin{aligned}\rho(w_i, w_j) &= \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,n-1} \tau(w_i, w_j) \\ \rho(w_{i'}, w_{j'}) &= \kappa_1 \dots \kappa_{n-1} \sigma(w_{i'}, w_{j'})\end{aligned}$$

for runs  $\tau(w_i, w_j)$ ,  $\sigma(w_{i'}, w_{j'})$  for all  $1 \leq i \leq n-1 < j$  and  $n-1 < i' < j'$ . For all  $1 \leq i < n$  define  $v_{i,n} := \tau(w_i, w_n)$ .

We now define  $\mu_{i,n}$  successively for each  $1 \leq i < n$ . In step  $i$  we apply the pigeonhole principle to get an infinite subsequence  $\nu$  of  $\mathbf{w}$  starting with  $w_1, \dots, w_n$  such that all runs  $\tau(v_i, v_j)$  for  $j > n$  have a common prefix  $\mu_{i,n}$  which is a run on  $\varepsilon \otimes \beta_n$ . At the end of step  $i$  we replace  $\mathbf{w}$  by  $\nu$  and we replace  $(\alpha, \beta)$  by the coarsening defined by  $\nu$ .

Next we define  $\lambda_n$ . By the pigeonhole principle there exists an infinite subsequence  $\mathbf{v}$  of  $\mathbf{w}$  starting with  $w_1, \dots, w_n$  such that all runs  $\sigma(v_n, v_j)$  for  $j > n$  have a common prefix  $\lambda_n$  which is a run on  $\alpha_n \otimes \beta_n$ . Again we replace  $\mathbf{w}$  by  $\mathbf{v}$  and  $(\alpha, \beta)$  by the coarsening defined by  $\mathbf{v}$ .

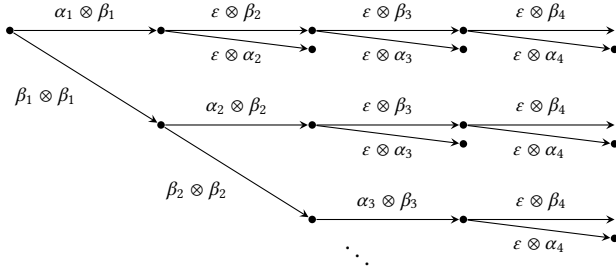
Finally, by Ramsey's theorem there is an infinite subsequence  $\mathbf{v}$  of  $\mathbf{w}$  starting with  $w_1, \dots, w_n$  such that all runs  $\sigma(v_i, v_j)$  for  $n < i < j$  have a common prefix  $\kappa_n$  which is a run on  $\beta_n \otimes \beta_n$ . We replace  $\mathbf{w}$  by  $\mathbf{v}$  and  $(\alpha, \beta)$  by the coarsening defined by  $\mathbf{v}$ .

In the limit we obtain the desired decomposition of  $\rho(v_i, v_j)$  and the generator  $(\alpha, \beta)$  of a comb  $v$  that is coarser than the initial generator of  $w$ .  $\square$

It is not hard to see that such a comb of combs structure can be simulated by an alternating Büchi automaton, which would only yield a PSPACE-solution for the infinite clique problem. The following key lemma states that the runs  $\mu_{i,j}$ ,  $\nu_{i,j}$  from [Lemma 4.2](#) can be chosen independently from  $i$ , which reduces the width of the run dag of the alternating automaton to a constant.

**Lemma 4.3.** *If  $\mathbf{w}$  is an infinite clique in a regular relation  $R \subseteq \Sigma^* \times \Sigma^*$  given as an NFA  $\mathcal{A}$ , then there exist a generator  $(\alpha, \beta)$  for a subsequence  $\mathbf{v}$  of  $\mathbf{w}$ , accepting runs  $\rho(v_i, v_j)$  of  $\mathcal{A}$  on  $v_i \otimes v_j$ , and runs  $\kappa_i, \lambda_i, \mu_j, \nu_j$  such that*

$$\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i+1} \dots \mu_{j-1} v_j$$



**Figure 2.** If  $v$  is a comb of the form  $v_i = \beta_1 \dots \beta_{i-1} \alpha_i$  then the convolutions  $v_i \otimes v_j$  for all  $i < j$  form a *comb of combs*.

is a decomposition compatible with the  $(\alpha, \beta)$ -decomposition of  $v_i \otimes v_j$  for all  $i < j$ .

*Proof.* Suppose that  $R$  has an infinite clique. Then there exist an infinite clique  $w$  in  $R$  generated by  $(\alpha, \beta)$  and runs  $\kappa_i, \lambda_i, \mu_{i,j}, v_{i,j}$  for  $i < j$  as in [Lemma 4.2](#).

It remains to ensure that  $\mu_{i,j} = \mu_{i',j}$  and  $v_{i,j} = v_{i',j}$  for all  $i < i' < j$ . To do so, consider the initial state of the run  $\mu_{i,j+1}$ . By Ramsey's theorem there exist indices  $k_1 < k_2 < \dots$  such that all runs  $\mu_{k_i, k_j+1}$  have the same initial state. We define  $v_i = w_{k_i}$  for all  $i \geq 1$ , and

$$\begin{aligned} \tilde{\kappa}_i &= \kappa_{k_{i-1}+1} \dots \kappa_{k_i} & \tilde{\lambda}_i &= \kappa_{k_{i-1}+1} \dots \kappa_{k_i-1} \lambda_{k_i}, \\ \tilde{\mu}_{i+1} &= \mu_{k_1, k_i+1} \dots \mu_{k_i, k_{i+1}} & \tilde{v}_{i+1} &= \mu_{k_1, k_i+1} \dots \mu_{k_i, k_{i+1}-1} v_{k_i, k_{i+1}}, \end{aligned}$$

for all  $i \geq 1$  where  $k_0 := 0$ . Observe that the composition  $\tilde{\lambda}_i \tilde{\mu}_{i+1}$  forms a valid run since  $\mu_{k_i, k_i+1}$  and  $\mu_{k_1, k_i+1}$  have the same initial state. Then  $\tilde{\kappa}_1 \dots \tilde{\kappa}_{i-1} \tilde{\lambda}_i \tilde{\mu}_{i+1} \dots \tilde{\mu}_{j-1} \tilde{v}_j$  is an accepting run on  $v_i \otimes v_j$ . Furthermore, this run decomposition is compatible with the  $(\delta, \gamma)$ -decomposition of  $v_i \otimes v_j$  where the generator  $(\delta, \gamma)$  is defined as

$$\begin{aligned} \delta_i &= \beta_{k_{i-1}+1} \dots \beta_{k_i} \\ \gamma_i &= \beta_{k_{i-1}+1} \dots \beta_{k_i-1} \alpha_{k_i} \end{aligned}$$

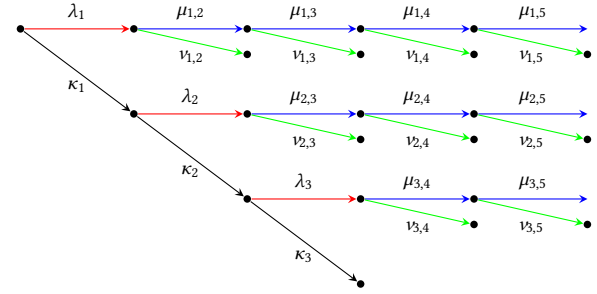
for all  $i \geq 1$  where  $k_0 := 0$ . This concludes the proof.  $\square$

A *nondeterministic Büchi automaton* (NBA) has the same format as an NFA  $\mathcal{B} = (Q, \Sigma, \Delta, q_0, F)$ . An infinite word  $w \in \Sigma^\omega$  is *accepted* by  $\mathcal{B}$  if there exists an accepting run  $(q_0, a_1, q_1)(q_1, a_2, q_2) \dots \in \Delta^\omega$  with  $w = a_1 a_2 \dots$  and  $q_i \in F$  for infinitely many  $i \geq 0$ .

**Proposition 4.4.** *Given an NFA  $\mathcal{A}$  for a relation  $R \subseteq (\Sigma^*)^2$ , one can construct in logarithmic space a Büchi automaton  $\mathcal{B}$  over the alphabet  $(\Sigma_\perp \times \Sigma) \cup \{\#\}$  such that:*

- If  $w$  is an infinite clique in  $R$  then  $\mathcal{B}$  accepts an encoding of a comb  $v$  which is a subsequence of  $w$ .
- If  $\mathcal{B}$  accepts an encoding of a comb  $w$  then  $w$  is an infinite clique in  $R$ .

For the proof we construct a Büchi automaton  $\mathcal{B}$  that simulates the runs  $\kappa_j, \lambda_j, \mu_j, v_j$  from [Lemma 4.3](#) in four components. We refer to [Appendix B](#) for the detailed construction.



**Figure 3.** One can always find an infinite comb clique  $v$  with accepting runs which can be decomposed in the form  $\rho(v_i, v_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} v_{i,j}$ .

*Proof of Theorem 2.2.* First observe that we can construct in log-space an NFA  $C$  over  $\Sigma$  such that (i) for every infinite clique  $w$  of  $R$  some element  $w_i$  is accepted by  $C$ , and (ii) if  $w$  is accepted by  $C$  then  $w$  belongs to an infinite clique of  $R$ . To be more precise,  $C$  accepts  $\alpha_1 \in \Sigma^*$  if and only if some encoding  $\text{enc}(\alpha, \beta)$  is accepted by the Büchi automaton  $\mathcal{B}$  from [Proposition 4.4](#). This can be done in log-space as follows: First we construct a Büchi automaton  $\hat{C}$  over  $\Sigma \cup \{\#\}$  which accepts all words of the form  $\alpha_1 \#^\omega$  such that an encoding  $\text{enc}(\alpha, \beta)$  is accepted by  $\mathcal{B}$ . Then  $\hat{C}$  is turned into an NFA  $C$  (which does not read the suffix  $\#^\omega$ ) by replacing  $\#$ -transitions by  $\epsilon$ -transitions. Furthermore  $C$  tracks the number of final states visited so far and accepts if and only if this number exceeds the number of states in  $\hat{C}$ .

Given an NFA  $\mathcal{A}$  for  $R \subseteq (\Sigma^*)^{k+2}$ . We first construct an NFA  $\mathcal{A}'$  over  $\Sigma_\perp^{2k+2}$  which accepts the regular binary relation

$$R' = \{(u \otimes c_1 \otimes \dots \otimes c_k, v \otimes c_1 \otimes \dots \otimes c_k) \mid (u, v, c) \in R\}.$$

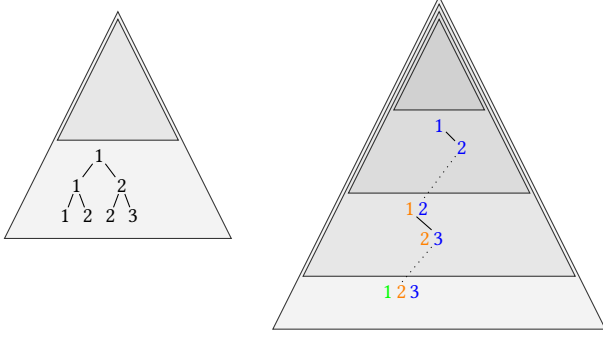
Note that  $\mathcal{A}'$  can be constructed in logspace since it is obtained by taking each transition of  $\mathcal{A}$  and duplicating the  $c$ -coordinates and moving the  $v$ -coordinate. Let  $C'$  be the NFA described above which accepts at least one word from each infinite  $R'$ -clique and only accepts elements of infinite  $R'$ -cliques. Projecting away the first component yields the desired NFA for  $\{c \in (\Sigma^*)^k \mid \exists^{\text{ram}} x, y: R(x, y, c)\}$ .  $\square$

## 5 Tree-automatic structures

### 5.1 EXP-hardness

In this section we prove the exponential lower bound from [Theorem 2.3](#) for the infinite clique problem over trees. This lower bound is surprising since over words, the infinite clique problem can be reduced to the emptiness of (word) Büchi automata, which is NL-complete. This is not the case in the tree case since emptiness of Büchi tree automata is P-complete.

We start with an intuitive explanation of the lower bound. To prove the upper bound in the word case, we used the fact that we can assume cliques  $v$  whose runs  $\rho(v_i, v_j)$  can



**Figure 4.** Left: Illustration of the tree automaton  $\mathcal{T}$ , tracking the number of right directions modulo  $n$ . Right: A path on which the runs of  $\mathcal{T}$  are disjoint.

be merged into a *single* global run (Lemma 4.3). Over tree regular relations this is not the case anymore. Consider a deterministic top-down tree automaton  $\mathcal{T}$ , which behaves as follows on the convolution  $t \otimes t'$  of two binary trees  $t, t'$  with  $\text{dom}(t) \subsetneq \text{dom}(t')$ : Starting from every node on the fringe of  $t$ , the automaton tracks the number of times it moves to a right child, modulo some number  $n$ ; see Figure 4 for a depiction. Now consider an increasing sequence of binary trees  $t_1, t_2, \dots$ , and the unique runs  $\rho_{i,j}$  of  $\mathcal{T}$  on  $t_i \otimes t_j$ . Figure 4 illustrates that we can always find a path on which the runs  $\rho_{1,n}, \rho_{2,n}, \dots, \rho_{n-1,n}$  are disjoint. This behavior indicates that it is difficult to witness the existence of infinite cliques by a polynomially-sized B  chi tree automaton.

We extend this idea to a reduction from the intersection non-emptiness problem for tree automata, which is known to be EXP-complete [8]: Given an NTA  $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$  and states  $q_1, \dots, q_n \in Q$ , decide whether  $\bigcap_{i=1}^n L(\mathcal{A}_{q_i})$  is non-empty. Here,  $\mathcal{A}_{q_i}$  denotes the NTA  $\mathcal{A}$  with initial state  $q_i$ .

We construct a relation  $R$  on decorated trees, which are obtained from a binary tree by attaching to every inner node  $u$  a ranked tree  $\delta(u)$  over  $\Sigma$ . Let  $\Gamma := \Sigma \cup \{a, c\}$  be a ranked alphabet with  $\text{rk}(a) = 3$ ,  $\text{rk}(c) = 0$ . A *decorated tree* is a tree  $t \in \mathcal{T}_\Gamma$  such that  $t(\varepsilon) \in \{a, c\}$  and for all  $u \in \text{dom}(t)$  we have

- $t(u1) = t(u2) \in \{a, c\}$  and  $t(u3) \in \Sigma$  if  $t(u) = a$  and
- $t(ui) \in \Sigma$  for all  $i \in [1, \text{rk}(t(u))]$  if  $t(u) \in \Sigma$ .

We denote by  $a(t) := \{u \in \text{dom}(t) \mid t(u) = a\}$  the nodes of  $t$  labeled with  $a$  and by  $ac(t) := \{u \in \text{dom}(t) \mid t(u) \in \{a, c\}\}$  the nodes labeled with  $a$  or  $c$ . The *decoration* of  $t$  is a function  $\delta_t: a(t) \rightarrow \mathcal{T}_\Sigma$  such that  $\delta_t(u) = t_{\downarrow u3}$  where  $t_{\downarrow v}$  denotes the subtree of  $t$  rooted in  $v \in \text{dom}(t)$ .

Let  $\mathcal{A}' = (Q', \Gamma, \Delta', p_1)$  be the NTA where  $Q' := Q \cup \{p_1, \dots, p_n\}$  and  $\Delta'$  contains all transitions from  $\Delta$  and the transitions

$$p_i \xrightarrow{a} (p_i, p_{i+1}, q_i) \text{ for all } i \in [1, n] \text{ where } p_{n+1} := p_1,$$

$$p_i \xrightarrow{c} () \text{ for all } i \in [1, n].$$

We define the tree-regular relation  $R \subseteq \mathcal{T}_\Gamma \times \mathcal{T}_\Gamma$  such that  $(s, t) \in R$  if and only if  $s$  and  $t$  are decorated trees with  $ac(s) \subseteq ac(t)$  and  $\mathcal{A}'$  accepts  $t_{\downarrow u}$  for all  $u \in \min(a(t) \setminus a(s))$ . Here, the minimum is defined with respect to prefix ordering. It is easy to construct an NTA that recognizes  $R$  in logspace.

It remains to show that  $\bigcap_{i=1}^n L(\mathcal{A}_{q_i}) \neq \emptyset$  if and only if  $R$  contains an infinite clique. For the “only if” direction let  $t \in \mathcal{T}_\Sigma$  be a tree that is accepted by  $\mathcal{A}_{q_i}$  for all  $i \in [1, n]$ . For all  $i \geq 0$  we define the decorated tree  $t_i \in \mathcal{T}_\Gamma$  such that  $ac(t_i) = \bigcup_{j=0}^i \{1, 2\}^j$  and  $\delta_{t_i}(u) = t$  for all  $u \in a(t_i)$ . It is easy to verify that  $(t_i, t_j) \in R$  for all  $i < j$ .

Conversely, consider a sequence of decorated trees  $t_i \in \mathcal{T}_\Gamma$  for  $i \geq 0$  with  $(t_i, t_j) \in R$  for all  $i < j$ . We define nodes  $v_1, \dots, v_n$  with

- $v_i \in \min(a(t_i) \setminus a(t_{i-1}))$  for all  $i \in [1, n]$  and
- $v_{i+1} = v_i 21^{k_i}$  for all  $i \in [1, n-1]$  and some  $k_i \geq 0$ .

We can choose  $v_1 \in \min(a(t_1) \setminus a(t_0))$  arbitrary which defines  $v_2, \dots, v_n$  uniquely. Since  $(t_{i-1}, t_n) \in R$  and  $v_i \in \min(a(t_n) \setminus a(t_{i-1}))$ , the subtree  $t_{n \downarrow v_i}$  is accepted by  $\mathcal{A}'$  for all  $i \in [1, n]$ . By definition of  $\mathcal{A}'$  there exist accepting runs on  $t_{n \downarrow v_n}$  starting from  $p_1, \dots, p_n$ . Therefore, the tree  $\delta_{t_n}(v_n) \in \mathcal{T}_\Sigma$  is accepted by  $\mathcal{A}$  starting from all states  $q_1, \dots, q_n$ .

We note that EXP-hardness already holds if  $R$  is given by a D  TA since intersection nonemptiness is EXP-hard already for D  TAs [27], and if the automaton  $\mathcal{A}$  is a D  TA, then the constructed relation  $R$  from the proof can also be recognized by a D  TA. Moreover, the reduction can be adapted to recurrent reachability by setting the target set to  $\mathcal{T}_\Sigma$ , which proves the exponential lower bound in Corollary 2.7.

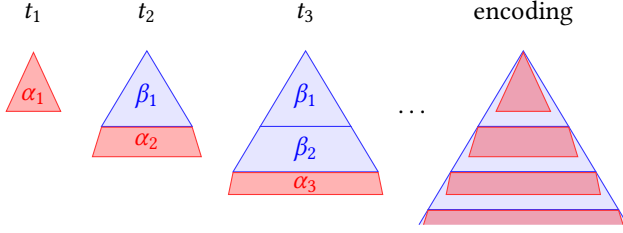
## 5.2 Tree combs

To prove the upper bounds for tree-regular relations, we extend the notion of combs to the tree case. Here a tree  $t_i$  is decomposed vertically in the form  $t_i = \beta_1 \dots \beta_{i-1} \alpha_i$  where  $\beta_1$  is a context,  $\beta_2, \dots, \beta_{i-1}$  are forests of contexts, and  $\alpha_i$  is a forest of trees, see Figure 5 for an abstract illustration.

A *context forest* of width  $n$  is a finite sequence  $\tau = (c_i)_{1 \leq i \leq n}$  of contexts  $c_i \in C_\Sigma$ . Context forests of width 1 are regarded as contexts. We say that  $\tau$  is *nontrivial* if  $n \geq 1$  and  $|c_i| \geq 1$  for all  $1 \leq i \leq n$ . If the  $c_i$  are trees in  $\mathcal{U}_\Sigma$ , we call  $\tau$  just a *forest*. We define the concatenation of a context  $\tau_1 \in C_\Sigma$  of width  $n$  with a context forest  $\tau_2 = (c_1, \dots, c_n) \in \mathcal{F}_\Sigma$  by  $\tau_1 \tau_2 := \tau_1[c_1, \dots, c_n]$ . Assuming left-associativity we write  $\tau_1 \tau_2 \dots \tau_n$  for a context  $\tau_1$  and context forests  $\tau_2, \dots, \tau_n$ . Here we implicitly assume that the width of  $\tau_i$  matches  $|\text{holes}(\tau_1 \dots \tau_{i-1})|$ . If  $t$  is a tree and  $s$  is a context, we write  $t \triangleleft s$  if  $\text{dom}(t) \cap \text{holes}(s) = \emptyset$ . For a forest  $\alpha = (t_i)_{i \leq n}$  and a context forest  $\beta = (s_i)_{i \leq n}$  we also write  $\alpha \triangleleft \beta$  if  $t_i \triangleleft s_i$  for all  $1 \leq i \leq n$ .

An infinite sequence  $\mathbf{t} = (t_i)_{i \geq 1}$  of ranked trees is called a *comb* if there is a sequence of forests  $\boldsymbol{\alpha} = (\alpha_i)_{i \geq 1}$  and a





**Figure 5.** An example tree comb and its encoding as an infinite tree. In this example the generator satisfies  $\text{dom}(\alpha_i) \subseteq \text{dom}(\beta_i)$  whereas general generators only satisfy  $\alpha_i \triangleleft \beta_i$ .

sequence of nontrivial context forests  $\beta = (\beta_i)_{i \geq 1}$  such that for all  $i \geq 1$  we have  $t_i = \beta_1 \dots \beta_{i-1} \alpha_i$  and  $\alpha_i \triangleleft \beta_i$ . The pair  $(\alpha, \beta)$  is called *generator* of the comb. Since the trees  $t_i$  are ranked, also the trees in  $\alpha_i$  and the contexts in  $\beta_i$  are ranked.

The property  $\alpha_i \triangleleft \beta_i$  should be compared to the property  $|\alpha_i| \leq |\beta_i|$  in word combs. It ensures that every forest  $\alpha_i$  does not touch any context forest  $\alpha_j, \beta_j$  for  $j \neq i$ .

**Lemma 5.1** (Combs lemma over trees). *Any sequence  $t$  of pairwise distinct ranked trees  $t_i \in \mathcal{T}_\Sigma$  over a finite ranked alphabet  $\Sigma$  contains a comb as a subsequence.*

*Proof.* It suffices to show that for any infinite set  $T \subseteq \mathcal{T}_\Sigma$  there exists a comb over  $T$ . Consider the following finitely branching infinite tree whose nodes are contexts from  $C_\Sigma$ . The root is the context  $x$ . The children of a context  $s$  are the contexts of the form  $s[t_1, \dots, t_n]$  where each  $t_i$  is a context of size one. Observe that all trees in  $\mathcal{T}_\Sigma$  occur as nodes in the infinite tree. The set of all ancestors of trees in  $T$  form an infinite subtree, which contains an infinite path  $s_0 <_p s_1 <_p s_2 <_p \dots$  by König's Lemma. For all  $i \geq 1$  there exists a tree  $t_i \in T$  which contains  $s_{i-1}$  as a prefix.

Since the minimal level of a hole in  $s_i$  is strictly increasing, for every  $i \geq 1$  there exists a  $j \geq i$  with  $t_i \triangleleft s_j$ . Hence one can inductively construct indices  $1 = k_1 < k_2 < \dots$  such that  $t_{k_{i+1}} \triangleleft s_{k_{i+1}}$  for all  $i \geq 1$ . Then  $(t_{k_{i+1}})_{i \geq 1}$  is a comb where the generator  $(\alpha, \beta)$  is defined such that  $s_{k_{i+1}} = \beta_1 \dots \beta_i$  and  $t_{k_{i+1}} = \beta_1 \dots \beta_{i-1} \alpha_i$  for  $i \geq 1$ . The comb property  $\alpha_i \triangleleft \beta_i$  follows from  $t_{k_{i+1}} \triangleleft s_{k_{i+1}}$ .  $\square$

To define the encoding  $\text{enc}(\alpha, \beta)$  of a comb generator we need a few more definitions. For a tree  $t$  and context  $s$  with  $t \triangleleft s$  we define the convolution  $t \otimes s$  as before but every  $(\perp, x)$  is replaced by  $x$ . That is,  $t \otimes s$  is again a context. We extend the convolution in a natural way to forests and context forests of the same width. If  $\tau = (c_1, \dots, c_n)$  is a context forest let  $\bar{\tau}$  be obtained from  $\tau$  by attaching a new  $\#$ -labeled root to each of the  $n$  contexts  $c_i$ . We can now define the *encoding* of a comb with generator  $(\alpha, \beta)$  as the infinite tree

$$\text{enc}(\alpha, \beta) := (\alpha_1 \otimes \beta_1) (\overline{\alpha_2 \otimes \beta_2}) (\overline{\alpha_3 \otimes \beta_3}) \dots$$

over the ranked alphabet  $\Omega := \Sigma_\perp^2 \cup \{\#\}$ . See Figure 5 for an illustration of the encoding. Here, the forests  $\alpha_i$  are colored

red and the context forests  $\beta_i$  are colored blue. It is not hard to see that the set  $\text{Enc}_\Sigma$  of all comb encodings is a regular language of infinite trees.

### 5.3 Arbitrary relations

An *alternating tree automaton* (ATA) over the ranked alphabet  $\Sigma$  is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0)$  where  $Q$  is a set of states,  $q_0 \in Q$  is an initial state, and  $\delta: Q \times \Sigma \rightarrow \mathcal{B}^+(Q \times \mathbb{N})$  is a transition function  $\delta(q, a) \in \mathcal{B}^+(Q \times \{1, \dots, \text{rk}(a)\})$  for all  $q \in Q, a \in \Sigma$ . Here,  $\mathcal{B}^+(Q \times \mathbb{N})$  denotes the set of positive propositional formulas over the set of variables  $Q \times \mathbb{N}$ . For a set  $S$  of variables and formula  $\varphi$  we denote by  $S \models \varphi$  that if the variables in  $S$  are set to true and the variables not in  $S$  are set to false, then  $\varphi$  is satisfied.

We will use a nonstandard definition of runs of ATAs. Firstly, we consider runs on both trees and contexts. Secondly, each node in the run also carries the labels of its children, with the purpose of predetermining the states in the context holes. A *run* of  $\mathcal{A}$  on a nontrivial ranked context  $t \in C_\Sigma$  is a context  $\rho$  over the alphabet  $\mathbb{N}^* \times Q \times 2^{Q \times \mathbb{N}}$  such that  $\rho(\varepsilon) = (\varepsilon, q_0, S)$  for some  $S \subseteq Q \times \mathbb{N}$  and for each node  $u \in \text{nodes}(\rho)$  with  $\rho(u) = (w, q, S)$  and  $S = \{(q_1, c_1), \dots, (q_r, c_r)\}$  we have that  $S \models \delta(q, t(w))$  and  $u$  has  $r \geq 0$  children such that  $\rho(ui) = (wc_i, q_i, S_i)$  for some  $S_i \subseteq Q \times \mathbb{N}$  if  $wc_i \in \text{nodes}(t)$ , and  $\rho(ui) = x$ , otherwise. Note that an NTA can be seen as a special ATA where for all  $q \in Q$  and  $a \in \Sigma$ , the transition formula  $\delta(q, a)$  is a disjunction of conjunctions  $\bigwedge_{i=1}^{\text{rk}(a)} (q_i, i)$ .

We define a *decomposition* of a context  $t$  as  $t = \tau_1 \dots \tau_n$  where the  $\tau_i$  are context forests. For a generator  $(\alpha, \beta)$  of a comb  $s$  we define the  $(\alpha, \beta)$ -decomposition of  $s_i \otimes s_j$  as in the word case. We say that a decomposition of a run  $\rho = \rho_1 \dots \rho_n$  of an ATA is *compatible* with a decomposition  $t = \tau_1 \dots \tau_n$  of a context if  $\rho_1 \dots \rho_i$  is a run on  $\tau_1 \dots \tau_i$  for all  $i \in [1, n]$ . Note that the above definition of a run ensures that  $\rho_1 \dots \rho_i$  already determines the first two components of the root labels of  $\rho_{i+1}$  for all  $i < n$ .

We say that a generator  $(\alpha, \beta)$  of a comb  $s$  is *coarser* than a generator  $(\gamma, \delta)$  of a comb  $t$  if there exist indices  $k_1 < k_2 < \dots$  such that  $s_i = t_{k_i}$  and  $\beta_1 \dots \beta_i = \delta_1 \dots \delta_{k_i}$  for all  $i \geq 1$ . In this case we also say that  $(\alpha, \beta)$  is the *coarsening* of  $(\gamma, \delta)$  defined by the subsequence  $s$  of  $t$ .

**Lemma 5.2.** *Let  $t$  be a comb generated by  $(\gamma, \delta)$  that forms an infinite clique in a tree-regular relation  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  given as an ATA  $\mathcal{A}$ . There exist a coarsening  $(\alpha, \beta)$  of  $(\gamma, \delta)$  that generates a comb  $s$ , runs  $\rho(s_i, s_j)$  of  $\mathcal{A}$  on  $s_i \otimes s_j$ , and context forests  $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$  such that*

$$\rho(s_i, s_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} \nu_{i,j}$$

*is a decomposition compatible with the  $(\alpha, \beta)$ -decomposition of  $s_i \otimes s_j$  for all  $i < j$ .*

*Proof.* The proof is similar to the proof of Lemma 4.2 in the word case. We emphasize that the pigeonhole principle and

Ramsey's theorem can be applied as in the word case since the unique prefixes of the runs of the ATA that are runs on a given context have bounded size. Further note that since a run on a forest is not defined, instead of considering only the suffixes  $\tau(t_i, t_j)$  and  $\sigma(t_i, t_j)$  in the inductive step, we have to consider the whole run  $\rho(t_i, t_j)$  and extend the common prefix that is already fixed.  $\square$

An *alternating B uchi tree automaton* (ABTA) over the ranked alphabet  $\Sigma$  is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  where  $Q$ ,  $\Sigma$ ,  $\delta$ , and  $q_0$  are as in the definition of an ATA and  $F \subseteq Q$  is a set of final states. A *run* of  $\mathcal{A}$  on a finite or infinite tree  $t \in \mathcal{T}_\Sigma^\infty$  is a tree  $\rho \in \mathcal{U}_{\mathbb{N}^* \times Q}^\infty$  such that  $\rho(\varepsilon) = (\varepsilon, q_0)$  and for all  $u \in \text{dom}(\rho)$  with  $\rho(u) = (w, q)$  and children  $u_1, \dots, u_r$  there is a satisfying assignment  $S = \{(q_1, c_1), \dots, (q_r, c_r)\} \models \delta(q, t(w))$  of pairwise distinct  $(q_i, c_i)$  such that  $\rho(ui) = (wc_i, q_i)$  for all  $i \in [1, r]$ . A run  $\rho$  is *accepting* if every infinite path of  $\rho$  contains infinitely many nodes with labels in  $\mathbb{N}^* \times F$ .

**Proposition 5.3.** *Given an ATA  $\mathcal{A}$  for a tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^\infty$ , one can construct in polynomial time an ABTA  $\mathcal{B}$  over the ranked alphabet  $\Omega = \Sigma_\perp^2 \cup \{\#\}$  such that we have:*

- *If  $t$  is an infinite clique in  $R$ , then  $\mathcal{B}$  accepts an encoding of a comb  $s$  which is a subsequence of  $t$ .*
- *If  $\mathcal{B}$  accepts  $t \in \mathcal{T}_\Omega^\infty$ , then  $t$  is an encoding of a comb  $t$  that is an infinite clique in  $R$ .*

*Proof.* Let  $\mathcal{A} = (Q, \Sigma_\perp^2, \delta, q_0)$  be the ATA that recognizes  $R$ . We construct an ABTA  $\mathcal{B}$  over  $\Omega$  which accepts precisely all comb encodings  $\text{enc}(\alpha, \beta)$  with the properties from Lemma 5.2. The state set of  $\mathcal{B}$  is  $Q \times \{1, 2, 3, 4\}$ , representing four different modes. In the first mode it simulates  $\kappa_j$  on  $\beta_j \otimes \beta_j$ , in the second mode it simulates  $\lambda_j$  on  $\alpha_j \otimes \beta_j$ , in the third mode it simulates  $\mu_{i,j}$  on  $\varepsilon \otimes \beta_j$ , and in the fourth mode it simulates  $v_{i,j}$  on  $\varepsilon \otimes \alpha_j$ . Figure 3 illustrates the simulation.

For all  $q \in Q$  and  $\binom{a}{b} \in \Sigma_\perp^2$  we set

$$\begin{aligned} \delta_{\mathcal{B}}((q, 1), \binom{a}{b}) &:= \delta_1(q, \binom{a}{b}) & \delta_{\mathcal{B}}((q, 2), \binom{a}{b}) &:= \delta_2(q, \binom{a}{b}) \\ \delta_{\mathcal{B}}((q, 3), \binom{a}{b}) &:= \delta_3(q, \binom{a}{b}) & \delta_{\mathcal{B}}((q, 4), \binom{a}{b}) &:= \delta_4(q, \binom{a}{b}) \end{aligned}$$

where  $\delta_i(q, \sigma)$  is the formula  $\delta(q, \sigma)$  where each variable  $(p, c)$  is replaced by  $((p, i), c)$ . At the holes of  $\alpha_j \otimes \beta_j$  labeled with  $\#$ , i.e., the points where  $\alpha_{j+1} \otimes \beta_{j+1}$  starts, the simulations in modes 1, 2, and 3 split up. For all  $q \in Q$  we define

$$\begin{aligned} \delta_{\mathcal{B}}((q, 1), \#) &:= ((q, 1), 1) \wedge ((q, 2), 1) \\ \delta_{\mathcal{B}}((q, 2), \#) &:= \delta_{\mathcal{B}}((q, 3), \#) := ((q, 3), 1) \wedge ((q, 4), 1) \end{aligned}$$

Finally, we add a new initial state  $q_0^{\mathcal{B}}$  which spawns simulations of  $\mathcal{A}$  in mode 1 and 2, i.e., for all  $q \in Q$  and  $\binom{a}{b} \in \Sigma_\perp^2$  we define

$$\delta_{\mathcal{B}}(q_0^{\mathcal{B}}, \binom{a}{b}) := \delta_1(q_0, \binom{a}{b}) \wedge \delta_2(q_0, \binom{a}{b}).$$

Finally, we intersect  $L(\mathcal{B})$  with the tree-regular language  $\text{Enc}_\Sigma$ , which concludes the proof.  $\square$

If  $R$  is given by a  $D\uparrow$ TA we can even compute in polynomial-time a *nondeterministic* B uchi tree automaton (NBTA) for the representation of infinite cliques. The proof idea is that the runs  $\mu_{i,j}$  and  $v_{i,j}$  in Lemma 5.2 only depend on  $j$ .

**Proposition 5.4.** *Given a  $D\uparrow$ TA  $\mathcal{A}$  for a tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^\infty$ , one can construct in polynomial time an NBTA  $\mathcal{B}$  over the ranked alphabet  $\Omega = \Sigma_\perp^2 \cup \{\#\}$  such that we have:*

- *If  $t$  is an infinite clique in  $R$ , then  $\mathcal{B}$  accepts an encoding of a comb  $s$  which is a subsequence of  $t$ .*
- *If  $\mathcal{B}$  accepts  $t \in \mathcal{T}_\Omega^\infty$ , then  $t$  is an encoding of a comb  $t$  that is an infinite clique in  $R$ .*

We are ready to prove Theorem 2.4. If  $\mathcal{A}$  is an ATA, we use the ABTA  $\mathcal{B}$  from Proposition 5.3 and transform it into an NBTA in exponential time [24, Theorem 1.2]. If  $\mathcal{A}$  is a  $D\uparrow$ TA  $\mathcal{A}$  we use the NBTA  $\mathcal{B}$  from Proposition 5.4. We then construct an NTA  $\mathcal{C}$  over  $\Sigma$  which accepts  $t_1 \in \mathcal{T}_\Sigma$  if and only if the encoding of some comb  $(t_i)_{i \geq 1}$  is accepted by  $\mathcal{B}$ . The rest of the proof is analogous to the proof of Theorem 2.2, see Appendix B for details.

#### 5.4 Transitive relations

In this section we show that if we assume that  $R$  is transitive, then the Ramsey quantifier can be evaluated in polynomial time (Theorem 2.5).

**Proposition 5.5.** *Given an NTA  $\mathcal{A}$  for a transitive tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^\infty$ , one can construct in polynomial time an NBTA  $\mathcal{B}$  over the ranked alphabet  $\Omega$  such that:*

- *If  $t$  is an infinite clique in  $R$ , then  $\mathcal{B}$  accepts an encoding of a comb  $s$  which is a subsequence of  $t$ .*
- *If  $\mathcal{B}$  accepts  $t \in \mathcal{T}_\Omega^\infty$ , then  $t$  is an encoding of a comb  $t$  that is an infinite clique in  $R$ .*

For the proof we view  $\mathcal{A}$  as an ATA and construct the ABTA  $\mathcal{B}$  as in the proof of Proposition 5.3 which accepts precisely all comb encodings  $\text{enc}(\alpha, \beta)$  with the properties from Lemma 5.2. By transitivity of  $R$  we can omit all states in mode 3. Then any run contains for each node  $v$  of the input tree at most three run nodes referring to  $v$ . Thus, we can apply a standard powerset construction to convert  $\mathcal{C}$  into an equivalent NBTA of polynomial size by restricting to subsets of states of size at most three. We refer to Appendix B for more details. Using the polynomially-sized NBTA  $\mathcal{B}$  we can prove Theorem 2.5 analogously to Theorem 2.4.

#### 5.5 Co-transitive relations

Recall that a binary relation  $R \subseteq A \times A$  is *co-transitive* if its complement  $\bar{R} = (A \times A) \setminus R$  is transitive. Next we show Theorem 2.6.

A context  $\beta$  is called *monadic* if it has exactly one hole. We will show that, if a co-transitive relation has an infinite clique, then there exists one which is a comb generated by a *monadic generator*  $(\alpha, \beta)$  in which all forests  $\beta_i$  are monadic contexts. This also implies that all  $\alpha_i$  are trees.

**Lemma 5.6.** *If a co-transitive tree-regular relation  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  has an infinite clique over a tree-regular language  $L \subseteq \mathcal{T}_\Sigma$ , then there exists an infinite clique  $t$  of  $R$  over  $L$  and a nontrivial monadic context  $\beta$  with  $t_1 \triangleleft \beta$  and  $\beta \leq_p t_i$  for all  $i \geq 2$ .*

*Proof.* Let  $\mathcal{A} = (Q, \Sigma_\perp^\perp, \Delta, F)$  and  $\mathcal{B} = (P, \Sigma, \Lambda, G)$  be  $\text{D}\uparrow\text{TAS}$  for  $R$  and  $L$ , respectively. Suppose that  $t$  is an infinite clique in  $R$  over  $L$ . For  $j \geq 2$  let  $c_j$  be the unique context with  $\text{nodes}(c_j) = \text{nodes}(t_1) \cap \text{nodes}(t_j)$  and  $c_j \leq_p t_j$ . Notice that  $c_j$  is nontrivial since  $t_1$  and  $t_j$  contain the root. Furthermore we have  $t_1 \triangleleft c_j$  since any hole  $u \in \text{holes}(c_j)$  is contained in  $\text{nodes}(t_j) \setminus \text{nodes}(c_j)$ . Since there are only finitely many such choices for  $c_j$ , by reducing  $t$  to a subsequence which starts with  $t_1$  we can assume that  $c_j = c$  for all  $j \geq 2$  for some context  $c$ . Suppose that  $v^1, \dots, v^n$  are the holes of  $c$  in lexicographical order, and  $t_j = c[t_j^1, \dots, t_j^n]$  for some trees  $t_j^k$ . For all  $i < j$  we have

$$t_i \otimes t_j = \begin{cases} (t_i \otimes c)[\varepsilon \otimes t_j^1, \dots, \varepsilon \otimes t_j^n], & 1 = i < j, \\ (c \otimes c)[t_i^1 \otimes t_j^1, \dots, t_i^n \otimes t_j^n], & 1 < i < j, \end{cases}$$

where  $t_1 \otimes c$  and  $c \otimes c$  are naturally viewed as contexts with  $n$  holes. For  $j \geq 2$  consider the accepting run  $\rho_j$  of  $\mathcal{A}$  on  $t_1 \otimes t_j$  and the accepting run  $\pi_j$  of  $\mathcal{B}$  on  $t_j$ , and color each index  $j$  by the tuple  $(\rho_j(v^1), \dots, \rho_j(v^n), \pi_j(v^1), \dots, \pi_j(v^n))$ . By the pigeonhole principle we can pick numbers  $1 = \ell_1 < \ell_2 < \dots$  such that  $\{\ell_2, \ell_3, \dots\}$  is monochromatic. We then replace  $t$  by  $(t_{\ell_i})_{i \geq 1}$ . Hence, the accepting runs of  $\mathcal{A}$  on  $t_1 \otimes t_j$  ( $j \geq 2$ ) visit the same states  $r^1, \dots, r^n$  in the nodes  $v^1, \dots, v^n$ . Similarly, the accepting runs of  $\mathcal{B}$  on the trees  $t_j$  visit the same states  $p^1, \dots, p^n$  in the nodes  $v^1, \dots, v^n$ . Therefore

$$\begin{aligned} (t_1, c[t_{j_1}^1, \dots, t_{j_n}^n]) &\in R && \text{for any } j_1, \dots, j_n \geq 2 \text{ and} \\ c[t_{j_1}^1, \dots, t_{j_n}^n] &\in L && \text{for any } j_1, \dots, j_n \geq 2. \end{aligned}$$

For  $1 < i < j$  consider an accepting run of  $\mathcal{A}$  on  $t_i \otimes t_j$  and let  $q_{i,j}^k$  be the state reached in node  $v^k$ . By Ramsey's theorem we can assume that there exist states  $q^1, \dots, q^n \in Q$  such that  $q_{i,j}^k = q^k$  for all  $1 < i < j$  (again, after replacing  $t$  by a subsequence starting with  $t_1$ ). Observe that  $\mathcal{A}$  accepts the context  $c \otimes c$  if it starts in nodes  $v^1, \dots, v^k$  with the states  $q^1, \dots, q^k$ , respectively.

For every  $0 \leq k \leq n$  define the tree

$$s_k = c[t_3^1, \dots, t_3^{k-1}, t_2^{k+1}, \dots, t_2^n].$$

We have  $(s_0, s_n) = (t_2, t_3) \in R$ . There must be an index  $1 \leq k \leq n$  with  $(s_{k-1}, s_k) \in R$  since otherwise by transitivity of  $\bar{R}$  we would have  $(s_0, s_n) \notin R$ . Define the context  $\beta = c[t_3^1, \dots, t_3^{k-1}, x, t_2^{k+1}, \dots, t_2^n]$ . Then we have that  $(\beta[t_2^k], \beta[t_3^k]) \in R$ . This is witnessed by an accepting run on

their convolution, which reaches state  $q^k$  at node  $v^k$ . This implies that  $(\beta[t_i^k], \beta[t_j^k]) \in R$  for all  $i < j$ , since the run of  $\mathcal{A}$  on  $t_i^k \otimes t_j^k$  also reaches  $q^k$ . Hence, the context  $\beta$  together with the trees  $t_1$  and  $\beta[t_i^k]$  for  $i \geq 2$  satisfy the claim. Moreover,  $t_1 \triangleleft c$  and  $c \leq_p \beta$  implies  $t_1 \triangleleft \beta$ .  $\square$

Repeated applications of [Lemma 5.6](#) yields the desired infinite clique:

**Lemma 5.7.** *If a co-transitive tree-regular relation  $R$  has an infinite clique then there exists an infinite clique  $t$  of  $R$  generated by a monadic generator  $(\alpha, \beta)$ .*

*Proof.* Let  $n \in \mathbb{N}$  and suppose we have inductively constructed trees  $\alpha_1, \dots, \alpha_n$ , and nontrivial monadic contexts  $\beta_1, \dots, \beta_n$ , with  $\alpha_i \triangleleft \beta_i$  for all  $1 \leq i \leq n$ , such that there exist trees  $(t'_i)_{i > n}$  such that  $(t_i)_{i \geq 1}$  is an infinite clique in  $R$  where  $t_i = \beta_1 \beta_2 \dots \beta_{i-1} \alpha_i$  if  $i \leq n$ , and  $t_i = \beta_1 \beta_2 \dots \beta_n t'_i$  if  $i > n$ . Let  $\beta := \beta_1 \beta_2 \dots \beta_n$ . Then  $(t'_i)_{i > n}$  is an infinite clique in the relation  $R' = \{(s', t') \mid (\beta[s'], \beta[t']) \in R\}$ . It is easy to see that  $R'$  is again tree-regular and co-transitive. Furthermore all trees  $t'_i$  for  $i > n$  belong to the tree-regular language  $L = \bigcap_{i=1}^n \{t \mid (t_i, \beta[t]) \in R\}$ . We can apply [Lemma 5.6](#) and obtain a tree  $\alpha_{n+1}$ , a nontrivial monadic context  $\beta_{n+1}$  with  $\alpha_{n+1} \triangleleft \beta_{n+1}$ , and trees  $(t''_i)_{i > n+1}$  such that  $\alpha_{n+1}$  together with  $\beta_{n+1}[t''_i]$  for  $i > n+1$  form an infinite clique in  $R'$ . Furthermore all trees  $\beta_{n+1}[t''_i]$  for  $i > n+1$  belong to  $L$ . Hence  $t_1, \dots, t_n$  together with  $\beta[\alpha_{n+1}]$  and  $\beta[\beta_{n+1}[t''_i]]$  for  $i > n+1$  form an infinite clique in  $R$ . By induction we then obtain the desired sequences  $\alpha, \beta$ .  $\square$

We can now prove [Theorem 2.6](#). Given an NTA  $\mathcal{A}$  for a co-transitive relation  $R$ . Using [Lemmas 5.2](#) and [5.7](#), we can prove a statement similar to [Lemma 4.3](#) for tree combs which are generated by a monadic generator. In particular, all context forests  $\kappa_i, \lambda_i, \mu_j$  have exactly one hole, and hence  $\kappa_i, \lambda_i, \mu_j, v_j$  are in fact contexts.

Now we can construct in polynomial time a Büchi tree automaton  $\mathcal{B}$  which accepts all comb encodings  $\text{enc}(\alpha, \beta)$  of a monadic generator  $(\alpha, \beta)$  for which runs of the form  $\kappa_j, \lambda_j, \mu_j, v_j$  as above exist. To this end,  $\mathcal{B}$  consists of four components in which the runs  $\kappa_j, \lambda_j, \mu_j, v_j$  are simulated. The detailed construction can be found in [Appendix B](#).

## 6 Applications

### 6.1 Recurrent reachability with generalized Büchi condition

The proof of the following Proposition can be found in [Appendix C](#).

**Proposition 6.1.** *The infinite clique problem and recurrent reachability are logspace equivalent over (tree-)regular relations. Moreover, the logspace reduction from recurrent reachability to the infinite clique problem preserves transitivity of relations and determinism of automata.*

Using the proposition we obtain tight bounds on the complexity of recurrent reachability over (transitive) (tree-)regular relations. We can even compute an automaton for the set  $\text{Rec}(L)[R]$  of initial elements given automata for  $R$  and  $L$ .

**Corollary 6.2.** *If  $R$  is a binary (tree-)regular relation and  $L$  is a (tree-)regular language given by NFAs (NTAs), then one can construct an NFA (NTA) for  $\text{Rec}(L)[R]$  in logspace (exponential time). The construction works in polynomial time if  $R$  and  $L$  are given by  $D\uparrow$ TAs or if  $R$  is transitive.*

*Proof.* We can define  $\text{Rec}(L)[R]$  by the formula

$$\begin{aligned} \varphi(x) = & \exists^{\text{ram}} y, z: R(x, y) \wedge L(y) \wedge R(y, z) \\ & \vee \exists y: R(x, y) \wedge L(y) \wedge R(y, y). \end{aligned}$$

If  $R$  and  $L$  are given by NFAs, we can construct in logspace an NFA for  $\text{Rec}(L)[R]$  using the closure properties of regular relations and [Theorem 2.2](#). Over trees, we use [Theorems 2.4](#) and [2.5](#) to construct an NTA for  $\text{Rec}(L)[R]$  in exponential or polynomial time depending on whether  $R$  is transitive and how  $R, L$  are given.  $\square$

For recurrent reachability with generalized B      condition we show that over words the complexity increases from NL to PSPACE, while over trees it stays in EXP ([Theorem 2.8](#)).

For both the word and the tree case we reduce the generalized version to the classical version. First observe that  $a_0 \in \text{Rec}(L_1, \dots, L_k)[R]$  if and only if there is a sequence  $\mathbf{a}$  such that  $(a_i, a_j) \in R$  for all  $0 \leq i < j$  and  $a_i \in L_{((i-1) \bmod k)+1}$  for all  $i \geq 1$ . We define a (tree-)regular relation  $R' \subseteq A^k \times A^k$  that checks if a tuple  $(a_1, \dots, a_{2k})$  forms a clique of size  $2k + 1$  in  $R$  starting with  $a_0$  such that  $a_i \in L_i$  for all  $i \in [1, k]$ . In the word case the NFA  $\mathcal{A}'$  for  $R'$  can be constructed in PSPACE using a product construction. In the tree case we can avoid the exponential blow-up for the product automaton by using ATAs. To make this work, we have to reduce the size of the alphabet for the ATA  $\mathcal{A}'$ . This can be achieved by encoding a tuple  $(\sigma_1, \dots, \sigma_k) \in \Sigma_{\perp}^k$  of symbols by a path  $\sigma_1(\sigma_2(\dots \sigma_k(\#_m) \dots))$  where  $\#_m$  is used as delimiter symbol of rank  $m := \max\{\text{rk}(\sigma_i) \mid 1 \leq i \leq k\}$ . Then the ATA  $\mathcal{A}'$  can be constructed in polynomial time. Now it holds that  $a_0 \in \text{Rec}(L_1, \dots, L_k)[R]$  if and only if  $\exists^{\text{ram}} x, y: R'(x, y) \vee \exists x: R'(x, x)$  is valid. By [Theorem 2.2](#) (resp. [Theorem 2.4](#)) validity of the first disjunct of  $\varphi$  can be checked in nondeterministic logspace (resp. exponential time) given  $\mathcal{A}'$ . It is easy to see that validity of the second disjunct of  $\varphi$  can also be checked in nondeterministic logspace (resp. exponential time) given  $\mathcal{A}'$ . This yields a PSPACE-algorithm in the word case and an EXP-algorithm in the tree case. Details are in [Appendix D](#).

For the lower bounds we reduce from the intersection non-emptiness problem of (tree-)regular languages  $L_1, \dots, L_k \subseteq A$ , which is known to be PSPACE-complete

over words [[12](#)] and EXP-complete over trees [[7](#), [Theorem 11](#)]. We define the (tree-)regular relation  $R \subseteq A \times A$  such that  $(a, b) \in R$  if and only if  $a = c$  or  $a = b$  where  $c \in A$  is some fixed element. Then  $L_1 \cap \dots \cap L_k \neq \emptyset$  if and only if  $c \in \text{Rec}(L_1, \dots, L_k)[R]$ .

For  $k = 1$ , the previous construction yields a reduction from nonemptiness for  $D\uparrow$ TAs, which is P-complete, to recurrent reachability over transitive tree-regular relations given by  $D\uparrow$ TAs, proving the P-hardness in [Theorem 2.3](#) and [Corollary 2.7](#).

In [[21](#)] L      shows that the reachability relation  $\rightarrow^*$  for regular ground tree rewrite systems (RGTRS) is tree-regular and an NTA for  $\rightarrow^*$  can be constructed in polynomial time. Hence, by [Theorem 2.8](#) recurrent reachability with generalized B      condition is EXP-complete for RGTRSs where hardness for GTRSs can be shown by a similar reduction as above from intersection nonemptiness.

**Corollary 6.3.** *For an RGTRS and tree-regular languages  $L_1, \dots, L_k$  given by NTAs, one can construct in exponential time an NTA recognizing  $\text{Rec}(L_1, \dots, L_k)[\rightarrow^*]$ .*

## 6.2 Monadic Decomposability

In the following we reduce monadic decomposability to the infinite clique problem over co-transitive relations, proving [Corollary 2.9](#) and [Corollary 2.10](#). A  $k$ -ary relation  $R$  over words or trees is monadically decomposable if and only if for all  $1 \leq j \leq k$  the equivalence relations  $\sim_j$  on  $(\Sigma^*)^k$  (or  $\mathcal{T}_{\Sigma}^k$ ) have finite index, where two tuples  $\mathbf{u} = (u_1, \dots, u_j)$ ,  $\mathbf{v} = (v_1, \dots, v_j)$  are  $\sim_j$ -equivalent if and only if

$$\forall \mathbf{w} = (w_{j+1}, \dots, w_k): [(\mathbf{u}, \mathbf{w}) \in R \iff (\mathbf{v}, \mathbf{w}) \in R],$$

see for example [[6](#), [Proof of Proposition 3.9](#)]. If the given automaton  $\mathcal{A}$  for  $R$  is a DFA,  $D\uparrow$ TA, or  $D\downarrow$ TA, then one can compute automata  $\mathcal{A}_{\sim_j}$  for the complements  $\sim_j$  of  $\sim_j$  in logspace, using the fact that  $\mathbf{u} \sim_j \mathbf{v}$  is equivalent to

$$\exists \mathbf{w}: ((\mathbf{u}, \mathbf{w}) \in R \wedge (\mathbf{v}, \mathbf{w}) \notin R) \vee ((\mathbf{u}, \mathbf{w}) \notin R \wedge (\mathbf{v}, \mathbf{w}) \in R).$$

If  $\mathcal{A}$  is an NFA or NTA, then this is possible in polynomial space, by determinizing  $\mathcal{A}$  and using closure properties of regular relations. Then, apply [Theorem 2.2](#) ([Theorem 2.6](#)) to  $\mathcal{A}_{\sim_j}$  to check in NL (resp. P) for an infinite clique in  $\sim_j$ .

The lower bounds are shown in [Appendix E](#) by a reduction from the universality problem for DFAs, NFAs,  $D\uparrow$ TAs, and NTAs, and the emptiness problem for  $D\downarrow$ TAs.

## 7 Unranked tree-automatic structures

In this section we consider the unranked tree analogue of [Theorems 2.4](#) and [2.5](#). Furthermore, we consider an application of the results to recurrent reachability in subtree and flat prefix rewriting systems. *Unranked tree-regular* languages and relations are recognized by nondeterministic unranked tree automata (NUTAs), see [Appendix F](#).



**Theorem 7.1.** *Given an unranked tree-regular  $R \subseteq (\mathcal{U}_\Sigma)^{k+2}$  by an NUTA  $\mathcal{A}$ , one can construct an NUTA for the relation  $[[\exists^{\text{ram}} x, y: R(x, y, z)]]$  in polynomial time if  $R$  is transitive and in exponential time otherwise. Hence, the infinite clique problem over (transitive) unranked tree-regular relations is in EXP (P).*

The proof can be found in [Appendix F.1](#). It uses the *first-child next-sibling encoding*, a standard regularity-preserving transformation from unranked trees to binary trees (see e.g., [10, 18, 25]). As over ranked trees, [Theorem 7.1](#) implies:

**Corollary 7.2.** *For a binary unranked tree-regular relation  $R$  and an unranked tree-regular language  $L$  given by NUTAs, one can construct an NUTA recognizing  $\text{Rec}(L)[R]$  in polynomial time if  $R$  is transitive and in exponential time otherwise.*

In [22] Löding and Spelten introduce tree rewriting systems over unranked trees called subtree and flat prefix rewriting systems (SFPRS). We refer to [Appendix F.2](#) for the definition. In [22] it is shown that the reachability relation  $\rightarrow^*$  for (regular) SFPRSs is an unranked tree-regular relation. Moreover, it can be observed that the NUTA for  $\rightarrow^*$  can be constructed in polynomial time. Since  $\rightarrow^*$  is transitive, we can apply [Corollary 7.2](#) to obtain that recurrent reachability for (regular) SFPRSs is decidable in polynomial time:

**Corollary 7.3.** *For a (regular) SFPRS and an unranked tree-regular language  $L$  given as NUTA, one can construct an NUTA recognizing  $\text{Rec}(L)[\rightarrow^*]$  in polynomial time.*

Let  $\text{FO}(\text{SFPRS})$  be the first-order theory over unranked trees with the reachability relation  $\rightarrow^*$  and the one-step reachability relation  $\rightarrow$  for (regular) SFPRSs. In [22] it is shown that the structure of  $\text{FO}(\text{SFPRS})$  is unranked tree-automatic which means that  $\text{FO}(\text{SFPRS})$  is decidable.

Let  $\text{FO}(\text{SFPRS} + \text{Rec})$  be the theory  $\text{FO}(\text{SFPRS})$  enriched by the recurrent reachability operator: For a formula  $\varphi$  in  $\text{FO}(\text{SFPRS} + \text{Rec})$  with one free variable we define the recurrent reachability operator  $\text{Rec}(\varphi)$  as formula with one free variable such that  $\text{Rec}(\varphi)(t)$  is true if and only if  $t \in \text{Rec}(L)[\rightarrow^*]$  for any  $t \in \mathcal{U}_\Sigma$  where  $L$  is the unranked tree-regular language defined by  $\varphi$ .

**Corollary 7.4.** *The theory  $\text{FO}(\text{SFPRS} + \text{Rec})$  is decidable.*

## 8 Conclusion and Future Works

We have identified directed Ramsey quantifiers as a fundamental notion that underlies the standard notion of Ramsey quantifiers, recurrent reachability, and monadic decomposability. We have also shown that the notion of *comb of combs* can be used to obtain substantially simpler proofs in case of word-automatic relations, and can be generalized to tree-automatic relations, allowing us to derive new results for Ramsey quantifiers, recurrent reachability and monadic decomposability (with applications to generalized Büchi conditions and unranked tree-automatic relations). There are

many natural research directions. In particular, we pinpoint that Ramsey quantifiers over  $\omega$ -automatic relations, as well as recurrent reachability over transitive  $\omega$ -automatic relations, is still a major open problem [14], although monadic decomposability is known to be decidable [23]. One possible approach is to consider the subclass of  $\omega$ -automatic relations that are definable over the theory of mixed integer-real linear arithmetic  $\langle \mathbb{R}; \mathbb{Z}, 1, 0, <, + \rangle$ , for which the problem of Ramsey quantifiers and recurrent reachability, to be the best of our knowledge, is still an open problem.

## Acknowledgments

Pascal Bergsträsser and Anthony Lin are supported by the ERC Starting Grant 759969 (AV-SMP).

## References

- [1] Parosh Aziz Abdulla, Bengt Jonsson, Pritha Mahata, and Julien d’Orso. 2002. Regular Tree Model Checking. In *Computer Aided Verification, 14th International Conference, CAV 2002, Copenhagen, Denmark, July 27-31, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2404)*, Ed Brinksma and Kim Guldstrand Larsen (Eds.). Springer, 555–568. [https://doi.org/10.1007/3-540-45657-0\\_47](https://doi.org/10.1007/3-540-45657-0_47)
- [2] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saxena. 2004. A Survey of Regular Model Checking. In *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3170)*, Philippa Gardner and Nobuko Yoshida (Eds.). Springer, 35–48. [https://doi.org/10.1007/978-3-540-28644-8\\_3](https://doi.org/10.1007/978-3-540-28644-8_3)
- [3] Pablo Barceló, Chih-Duo Hong, Xuan Bach Le, Anthony W. Lin, and Reino Niskanen. 2019. Monadic Decomposability of Regular Relations. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 103:1–103:14. <https://doi.org/10.4230/LIPIcs.ICALP.2019.103>
- [4] Achim Blumensath and Erich Grädel. 2000. Automatic Structures. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*. IEEE Computer Society, 51–62. <https://doi.org/10.1109/LICS.2000.855755>
- [5] Achim Blumensath and Erich Grädel. 2004. Finite Presentations of Infinite Structures: Automata and Interpretations. *Theory Comput. Syst.* 37, 6 (2004), 641–674. <https://doi.org/10.1007/s00224-004-1133-y>
- [6] Olivier Carton, Christian Choffrut, and Serge Grigorieff. 2006. Decision problems among the main subfamilies of rational relations. *RAIRO Theor. Informatics Appl.* 40, 2 (2006), 255–275. <https://doi.org/10.1051/ita:2006005>
- [7] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. 1997. *Tree Automata Techniques and Applications*. (1997).
- [8] Thom W. Frühwirth, Ehud Shapiro, Moshe Y. Vardi, and Eyal Yardeni. 1991. Logic Programs as Types for Logic Programs. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS ’91)*, Amsterdam, The Netherlands, July 15-18, 1991. IEEE Computer Society, 300–309. <https://doi.org/10.1109/LICS.1991.151654>
- [9] Seymour Ginsburg and Edwin H Spanier. 1966. Bounded regular sets. *Proc. Amer. Math. Soc.* 17, 5 (1966), 1043–1049. <https://doi.org/10.1090/S0002-9939-1966-0201310-3>
- [10] Georg Gottlob, Christoph Koch, Reinhard Pichler, and Luc Segoufin. 2005. The complexity of XPath query evaluation and XML typing. *J. ACM* 52, 2 (2005), 284–335. <https://doi.org/10.1145/1059513.1059520>

- [11] Erich Grädel. 2020. Automatic Structures: Twenty Years Later. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Holger Hermanns, Li-jun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 21–34. <https://doi.org/10.1145/3373718.3394734>
- [12] Dexter Kozen. 1977. Lower Bounds for Natural Proof Systems. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 254–266. <https://doi.org/10.1109/SFCS.1977.16>
- [13] Dexter Kozen. 1997. *Automata and computability*. Springer.
- [14] Dietrich Kuske. 2010. Is Ramsey’s Theorem omega-automatic?. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France (LIPIcs, Vol. 5)*, Jean-Yves Marion and Thomas Schwentick (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 537–548. <https://doi.org/10.4230/LIPIcs.STACS.2010.2483>
- [15] Dietrich Kuske and Markus Lohrey. 2010. Some natural decision problems in automatic graphs. *J. Symb. Log.* 75, 2 (2010), 678–710. <https://doi.org/10.2178/jsl/1268917499>
- [16] Jérôme Leroux and Grégoire Sutre. 2006. Flat counter automata almost everywhere!. In *Software Verification: Infinite-State Model Checking and Static Program Analysis, 19.02. - 24.02.2006 (Dagstuhl Seminar Proceedings, Vol. 06081)*, Parosh Aziz Abdulla, Ahmed Bouajjani, and Markus Müller-Olm (Eds.). Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. <http://drops.dagstuhl.de/opus/volltexte/2006/729>
- [17] Leonid Libkin. 2003. Variable independence for first-order definable constraints. *ACM Trans. Comput. Log.* 4, 4 (2003), 431–451. <https://doi.org/10.1145/937555.937557>
- [18] Leonid Libkin. 2005. Logics for Unranked Trees: An Overview. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3580)*, Luis Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung (Eds.). Springer, 35–50. [https://doi.org/10.1007/11523468\\_4](https://doi.org/10.1007/11523468_4)
- [19] Anthony Widjaja Lin. 2012. Accelerating tree-automatic relations. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*. 313–324. <https://doi.org/10.4230/LIPIcs.FSTTCS.2012.313>
- [20] Anthony W. Lin and Philipp Rümmer. 2021. Regular Model Checking Revisited. In *Model Checking, Synthesis, and Learning - Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday*. 97–114. [https://doi.org/10.1007/978-3-030-91384-7\\_6](https://doi.org/10.1007/978-3-030-91384-7_6)
- [21] Christof Löding. 2006. Reachability Problems on Regular Ground Tree Rewriting Graphs. *Theory Comput. Syst.* 39, 2 (2006), 347–383. <https://doi.org/10.1007/s00224-004-1170-6>
- [22] Christof Löding and Alex Spelten. 2007. Transition Graphs of Rewriting Systems over Unranked Trees. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Český Krumlov, Czech Republic, August 26-31, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4708)*, Ludek Kucera and Antonín Kucera (Eds.). Springer, 67–77. [https://doi.org/10.1007/978-3-540-74456-6\\_8](https://doi.org/10.1007/978-3-540-74456-6_8)
- [23] Christof Löding and Christopher Spinrath. 2019. Decision Problems for Subclasses of Rational Relations over Finite and Infinite Words. *Discret. Math. Theor. Comput. Sci.* 21, 3 (2019). <http://dmtdcs.episciences.org/5141>
- [24] David E. Muller and Paul E. Schupp. 1995. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.* 141, 1&2 (1995), 69–107. [https://doi.org/10.1016/0304-3975\(94\)00214-4](https://doi.org/10.1016/0304-3975(94)00214-4)
- [25] Frank Neven. 2002. Automata, Logic, and XML. In *Computer Science Logic, 16th International Workshop, CSL 2002, 11th Annual Conference of the EACSL, Edinburgh, Scotland, UK, September 22-25, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2471)*, Julian C. Bradfield (Ed.). Springer, 2–26. [https://doi.org/10.1007/3-540-45793-3\\_2](https://doi.org/10.1007/3-540-45793-3_2)
- [26] Sasha Rubin. 2008. Automata Presenting Structures: A Survey of the Finite String Case. *Bull. Symb. Log.* 14, 2 (2008), 169–209. <https://doi.org/10.2178/bsl/1208442827>
- [27] Helmut Seidl. 1994. Haskell Overloading is DEXPTIME-Complete. *Inf. Process. Lett.* 52, 2 (1994), 57–60. [https://doi.org/10.1016/0020-0190\(94\)00130-8](https://doi.org/10.1016/0020-0190(94)00130-8)
- [28] Richard Edwin Stearns. 1967. A Regularity Test for Pushdown Machines. *Inf. Control.* 11, 3 (1967), 323–340. [https://doi.org/10.1016/S0019-9958\(67\)90591-8](https://doi.org/10.1016/S0019-9958(67)90591-8)
- [29] Anthony Widjaja To and Leonid Libkin. 2008. Recurrent Reachability Analysis in Regular Model Checking. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*. 198–213. [https://doi.org/10.1007/978-3-540-89439-1\\_15](https://doi.org/10.1007/978-3-540-89439-1_15)
- [30] Anthony Widjaja To and Leonid Libkin. 2010. Algorithmic Metatheorems for Decidable LTL Model Checking over Infinite Systems. In *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*. 221–236. [https://doi.org/10.1007/978-3-642-12032-9\\_16](https://doi.org/10.1007/978-3-642-12032-9_16)
- [31] Leslie G. Valiant. 1975. Regularity and Related Problems for Deterministic Pushdown Automata. *J. ACM* 22, 1 (1975), 1–10. <https://doi.org/10.1145/321864.321865>
- [32] Margus Veanes. 1997. *On computational complexity of basic decision problems of finite tree automata*. Technical Report. UPMail Technical Report 133, Uppsala University, Computing Science Department.
- [33] Margus Veanes, Nikolaj Bjørner, Lev Nachmanson, and Sergey Bereg. 2017. Monadic Decomposition. *J. ACM* 64, 2 (2017), 14:1–14:28. <https://doi.org/10.1145/3040488>

## A Directed vs undirected cliques

**Theorem 2.2** and **Theorem 2.4** also hold if we alternatively define the Ramsey quantifier  $\exists^{\text{ram}}x, y: \varphi(x, y, z)$  using *infinite undirected cliques*, i.e., there exists an infinite set  $X$  such that  $\varphi(a, b, z)$  holds for all  $a, b \in X$  with  $a \neq b$ , since we can replace  $\varphi(x, y, z)$  by  $\varphi(x, y, z) \wedge \varphi(y, x, z)$ .

Furthermore, the NL-lower bound in the word case (**Corollary 2.9**) and the EXP-lower bound in the tree case (**Theorem 2.3**) also hold for undirected cliques:

**Proposition A.1.** *The infinite clique problems for directed and undirected cliques are logspace equivalent over (tree-)regular relations.*

*Proof.* We first reduce the undirected version to the directed version. Let  $R \subseteq A \times A$  be given by an NFA (resp. NTA)  $\mathcal{A}$ . Then we define the relation  $R' := \{(a, b) \in A \times A \mid (a, b) \in R \wedge (b, a) \in R\}$ . Clearly,  $R'$  is a (tree-)regular relation and an NFA (resp. NTA) recognizing  $R'$  can be constructed in logspace from  $\mathcal{A}$ . Moreover, we have that  $R$  has an infinite undirected clique if and only if  $R'$  has an infinite directed clique.

For the reverse reduction, let  $R \subseteq A \times A$  be given by an NFA (resp. NTA)  $\mathcal{A}$ . We define the relation

$$R' := \{((a, i), (b, j)) \in (A \times \mathbb{N})^2 \mid a \neq b \wedge ((a, b) \in R \wedge i < j \vee (b, a) \in R \wedge j < i)\}.$$

It is easy to see that  $R'$  can be encoded as a (tree-)regular relation and an NFA (resp. NTA) recognizing this relation can be constructed in logspace from  $\mathcal{A}$ . It holds that  $R$  has an infinite directed clique if and only if  $R'$  has an infinite undirected clique. Indeed, if  $R$  has an infinite directed clique  $(a_i)_{i \geq 1}$ , then we can number the elements and get an infinite undirected clique  $((a_i, i))_{i \geq 1}$  in  $R'$ . Conversely, if  $R'$  has an infinite undirected clique  $((a_i, n_i))_{i \geq 1}$ , then  $(a_{i_j})_{j \geq 1}$  with  $n_{i_j} < n_{i_{j'}}$  for  $j < j'$  is an infinite directed clique in  $R$ .  $\square$

## B Constructions of Büchi automata

**Proposition 4.4.** *Given an NFA  $\mathcal{A}$  for a relation  $R \subseteq (\Sigma^*)^2$ , one can construct in logarithmic space a Büchi automaton  $\mathcal{B}$  over the alphabet  $(\Sigma_\perp \times \Sigma) \cup \{\#\}$  such that:*

- If  $\mathbf{w}$  is an infinite clique in  $R$  then  $\mathcal{B}$  accepts an encoding of a comb  $\mathbf{v}$  which is a subsequence of  $\mathbf{w}$ .
- If  $\mathcal{B}$  accepts an encoding of a comb  $\mathbf{w}$  then  $\mathbf{w}$  is an infinite clique in  $R$ .

*Proof.* Given an NFA  $\mathcal{A} = (Q, \Sigma_\perp^2, q_{\text{in}}, \Delta, F)$ , we add to  $\mathcal{A}$  a state  $\perp$  and transitions  $\perp \xrightarrow{(a,b)} q$  for all  $(a, b) \in \Sigma_\perp^2$  and  $q \in Q_\perp$ .

The Büchi automaton  $\mathcal{B} = (Q_\perp^4, \Sigma_\perp^2, q_{\text{in}}^\mathcal{B}, \Delta^\mathcal{B}, Q_\perp^4)$  simulates the runs  $\kappa_j, \lambda_j, \mu_j, \nu_j$  from Lemma 4.3 in the four components. Its initial state is  $q_{\text{in}}^\mathcal{B} = (q_{\text{in}}, q_{\text{in}}, \perp, \perp)$  and it contains the following transitions:

- $(p, s, q, t) \xrightarrow{(a,b)}_\mathcal{B} (p', s', q', t')$  for all  $p \xrightarrow{(b,b)}_\mathcal{A} p'$ ,  $s \xrightarrow{(a,b)}_\mathcal{A} s'$ ,  $q \xrightarrow{(\perp,b)}_\mathcal{A} q'$ ,  $t \xrightarrow{(\perp,a)}_\mathcal{A} t'$ ,
- $(p, q, q, t) \xrightarrow{\#}_\mathcal{B} (p, p, q, q)$  for all  $p, q \in Q, t \in F$ ,
- $(p_1, p_2, p_3, p_4) \xrightarrow{\varepsilon}_\mathcal{B} (q_1, q_2, q_3, q_4)$  whenever  $p_i = q_i$  or  $p_i \xrightarrow{\varepsilon}_\mathcal{A} q_i$  for all  $i$ .

The desired Büchi automaton is a product automaton of  $\mathcal{B}$  and a Büchi automaton which verifies that the input word is a valid comb encoding  $\text{enc}(\alpha, \beta)$ .  $\square$

**Proposition 5.4.** *Given a  $D\uparrow$ TA  $\mathcal{A}$  for a tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^2$ , one can construct in polynomial time an NBTA  $\mathcal{B}$  over the ranked alphabet  $\Omega = \Sigma_\perp^2 \cup \{\#\}$  such that we have:*

- If  $\mathbf{t}$  is an infinite clique in  $R$ , then  $\mathcal{B}$  accepts an encoding of a comb  $\mathbf{s}$  which is a subsequence of  $\mathbf{t}$ .
- If  $\mathcal{B}$  accepts  $\mathbf{t} \in \mathcal{T}_\Omega^\infty$ , then  $\mathbf{t}$  is an encoding of a comb  $\mathbf{t}$  that is an infinite clique in  $R$ .

*Proof.* Let  $\mathcal{A}' = (Q, \Sigma_\perp^2, \Delta, q_{\text{in}})$  be the NTA that is obtained by reverting the transitions of the  $D\uparrow$ TA  $\mathcal{A}$ . Clearly,  $\mathcal{A}'$  has the same runs as  $\mathcal{A}$  on trees. Let  $\mathbf{t}$  be a comb that forms an infinite clique in  $R$ . Since any NTA is a special ATA, we

can apply Lemma 5.2 on  $\mathcal{A}'$  and  $\mathbf{t}$  to get a subcomb  $\mathbf{s}$  of  $\mathbf{t}$  generated by  $(\alpha, \beta)$ , runs  $\rho(s_i, s_j)$  of  $\mathcal{A}'$  on  $s_i \otimes s_j$ , and context forests  $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$  such that

$$\rho(s_i, s_j) = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} \nu_{i,j}$$

is a decomposition compatible with the  $(\alpha, \beta)$ -decomposition of  $s_i \otimes s_j$  for all  $i < j$ . Moreover, we have that  $\mu_{i,j}$  and  $\nu_{i,j}$  only depend on  $j$  since there are unique runs of  $\mathcal{A}$  on  $(\varepsilon \otimes \beta_j)(\varepsilon \otimes \alpha_{j+1})$  and  $\varepsilon \otimes \alpha_j$ . Thus, we can just write  $\mu_j$  and  $\nu_j$  for all  $j > 1$ .

We now construct an NBTA  $\mathcal{B}$  over the alphabet  $\Omega$  which accepts precisely all comb encodings  $\text{enc}(\alpha, \beta)$  of a generator  $(\alpha, \beta)$  with the above properties. Since the set of all comb encodings is regular, we can assume that the input tree is already a valid comb encoding. A state in  $\mathcal{B}$  consists of four components in which  $\kappa_j, \lambda_j, \mu_j, \nu_j$  are simulated. To handle the special case where only  $\kappa_0, \lambda_0$  are simulated, we add a state  $\perp$  to  $\mathcal{A}'$  with transitions  $\perp \xrightarrow{(a,b)} q$  for all symbols  $(a, b) \in \Sigma_\perp^2$  of rank  $r$ , and  $q \in Q_\perp^r$ . The NBTA  $\mathcal{B}$  has the state set  $Q_\perp^4$ , initial state  $(q_{\text{in}}, q_{\text{in}}, \perp, \perp)$ , and the transitions

- $(p, s, q, t) \xrightarrow{(a,b)} p \otimes s \otimes q \otimes t$  if  $\mathcal{A}'$  contains the transitions  $p \xrightarrow{(b,b)} p, s \xrightarrow{(a,b)} s, q \xrightarrow{(\perp,b)} q, t \xrightarrow{(\perp,a)} t$ ,
- $(p, q, q, \perp) \xrightarrow{\#} (p, p, q, q)$  for all  $p, q \in Q$ .

Correctness follows from the previous observations.  $\square$

**Theorem 2.4.** *Given an ATA ( $D\uparrow$ TA)  $\mathcal{A}$  for a tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^{k+2}$ , one can construct in exponential (polynomial) time an NTA for the relation  $[\exists^{\text{ram}} x, y: R(x, y, z)]$ .*

*Proof.* First observe that from the ABTA  $\mathcal{B}$  in Proposition 5.3 we can construct in exponential time an NTA  $\mathcal{C}$  over  $\Sigma$  which accepts  $t_1 \in \mathcal{T}_\Sigma$  if and only if the encoding of some comb  $(t_i)_{i \geq 1}$  is accepted by  $\mathcal{B}$ . Indeed, we first transform  $\mathcal{B}$  into an NBTA  $\mathcal{D} = (Q_\mathcal{D}, \Omega, \Delta_\mathcal{D}, q_0^\mathcal{D}, F_\mathcal{D})$  which can be done in exponential time [24, Theorem 1.2]. From  $\mathcal{D}$  we construct  $\mathcal{C} = (Q_\mathcal{C}, \Sigma, \Delta_\mathcal{C}, q_0^\mathcal{C})$  such that for all  $q \in Q_\mathcal{D}, a \in \Sigma$ , and  $p_i \in Q_\mathcal{D}$  for  $1 \leq i \leq \text{rk}(a)$  we let  $(q, a, (p_i)_{i \leq \text{rk}(a)}) \in \Delta_\mathcal{C}$  if and only if there exist  $b \in \Sigma_\perp$  and  $p_i \in Q_\mathcal{D}$  for  $\text{rk}(a) < i \leq \text{rk}(b)$  such that

$$(q, (b), (p_i)_{i \leq \text{rk}(b)}) \in \Delta_\mathcal{D}$$

and  $\mathcal{D}$  accepts some tree from state  $p_i$  for all  $i > \text{rk}(a)$ . Note that  $\mathcal{C}$  can be constructed in polynomial time given  $\mathcal{D}$  since we need to perform a polynomial number of non-emptiness checks on  $\mathcal{D}$ , each of which takes quadratic time. The NTA  $\mathcal{C}$  satisfies that (i) for every infinite clique  $\mathbf{t}$  of  $R$  some element  $t_i$  is accepted by  $\mathcal{C}$  and (ii) if  $\mathbf{t}$  is accepted by  $\mathcal{C}$ , then  $\mathbf{t}$  belongs to an infinite clique of  $R$ .

Given an NTA  $\mathcal{A}$  for  $R \subseteq (\mathcal{T}_\Sigma)^{k+2}$ . We first construct an NTA  $\mathcal{A}'$  over  $\Sigma_\perp^{2k+2}$  accepting the binary relation

$$R' = \{(s \otimes c_1 \otimes \dots \otimes c_k, t \otimes c_1 \otimes \dots \otimes c_k) \mid (s, t, \mathbf{c}) \in R\}.$$

Let  $\mathcal{C}'$  be the NTA described above that accepts at least one tree from each infinite  $R'$ -clique and only accepts elements



of infinite  $R'$ -cliques. Projecting away the first component yields the desired NTA for  $\{c \in (\mathcal{T}_\Sigma)^k \mid \exists^{\text{ram}} x, y: R(x, y, c)\}$ .

If  $R$  is given as  $D\uparrow\text{TA}$ , then one can construct an NBTA in polynomial time instead of an ABTA using [Proposition 5.4](#). Then  $C'$  can be constructed in polynomial time.  $\square$

**Proposition 5.5.** *Given an NTA  $\mathcal{A}$  for a transitive tree-regular relation  $R \subseteq (\mathcal{T}_\Sigma)^2$ , one can construct in polynomial time an NBTA  $\mathcal{B}$  over the ranked alphabet  $\Omega$  such that:*

- *If  $t$  is an infinite clique in  $R$ , then  $\mathcal{B}$  accepts an encoding of a comb  $s$  which is a subsequence of  $t$ .*
- *If  $\mathcal{B}$  accepts  $t \in \mathcal{T}_\Omega^\infty$ , then  $t$  is an encoding of a comb  $t$  that is an infinite clique in  $R$ .*

*Proof.* We view the NTA  $\mathcal{A}$  for  $R$  as an ATA and construct the ABTA  $\mathcal{B}$  as in the proof of [Proposition 5.3](#) which accepts precisely all comb encodings  $\text{enc}(\alpha, \beta)$  with the properties from [Lemma 5.2](#). Then, we omit all states in mode 3. More formally, let  $C = (Q_C, \Omega, \delta_C, q_0^C, Q_C)$  be the ABTA with state set  $Q_C = \{q_0^C\} \cup (Q \times \{1, 2, 4\})$ , and the same transitions as  $\mathcal{B}$  except for  $\delta_C((q, 2), \#) := ((q, 4), 1)$  for all  $q \in Q$ . Clearly, all comb encodings with the properties from [Lemma 5.2](#) are still accepted by  $C$ . Conversely, if the encoding  $\text{enc}(\alpha, \beta)$  of a comb  $t$  is accepted by  $C$  then for all  $i \geq 1$ ,  $C$  simulates a run of  $\mathcal{A}$  on  $t_i \otimes t_{i+1}$  as argued in [Proposition 5.3](#). By transitivity we obtain  $(t_i, t_j) \in R$  for all  $i < j$ .

It remains to convert  $C$  into an NBTA of polynomial size. Observe that  $C$  only universally branches in the root and at  $\#$ -nodes into a state of mode 1 and 2. Furthermore, states of mode 2 transition to mode 4 when reading  $\#$ . Hence, any run contains for each node  $v$  of the input tree at most three run nodes referring to  $v$ . Thus, we apply a standard power-set construction to convert  $C$  into an equivalent NBTA  $\mathcal{D}$ , where we restrict to subsets of  $Q_C$  of size at most three. We make all states in  $\mathcal{D}$  final, since  $C$  accepts any tree with a run. Finally, we take the product construction of  $\mathcal{D}$  with an NBTA for  $\text{Enc}_\Sigma$ , to obtain the desired NBTA in polynomial time.  $\square$

**Theorem 2.6.** *The infinite clique problem over co-transitive tree-regular relations  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  given as NTA is in P.*

*Proof.* Given an NTA  $\mathcal{A} = (Q, \Sigma_1^2, \Delta, q_{\text{in}})$  for a co-transitive relation  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$ . We can prove a statement similar to [Lemma 4.3](#) for tree combs which are generated by a monadic generator.

Suppose that  $R$  has an infinite clique. By [Lemma 5.7](#) and [Lemma 5.2](#) there exist an infinite clique  $t$  in  $R$  with a monadic generator  $(\alpha, \beta)$ , and context forests  $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$  for  $i < j$  such that  $\rho_{i,j} = \kappa_1 \dots \kappa_{i-1} \lambda_i \mu_{i,i+1} \dots \mu_{i,j-1} \nu_{i,j}$  is a decomposition compatible with the  $(\alpha, \beta)$ -decomposition of  $t_i \otimes t_j$ . In particular, all context forests  $\kappa_i, \lambda_i, \mu_{i,j}$  have exactly one hole, and hence  $\kappa_i, \lambda_i, \mu_{i,j}, \nu_{i,j}$  are in fact contexts.

Moreover, we can ensure that  $\mu_{i,j} = \mu_{i',j}$  and  $\nu_{i,j} = \nu_{i',j}$  for all  $i < i' < j$  and can therefore just write  $\mu_j$  and  $\nu_j$  for all

$j > 1$ , respectively. For the proof we can reason similarly as in [Lemma 4.3](#) by applying Ramsey's theorem to ensure that all contexts  $\mu_{i,j}$  carry the same state in the root. This allows us to verify the runs using a polynomially sized NBTA on the comb encoding.

Finally, we can construct in polynomial time a B  chi tree automaton  $\mathcal{B}$  over the alphabet  $\Omega = \Sigma_\perp^2 \cup \{\#\}$  which accepts all comb encodings  $\text{enc}(\alpha, \beta)$  of a monadic generator  $(\alpha, \beta)$  for which runs of the form  $\kappa_j, \lambda_j, \mu_j, \nu_j$  as above exist. Since the set of all monadic comb encodings is regular, we can assume that the input tree is already a valid monadic comb encoding. A state in  $\mathcal{B}$  consists of four components in which the runs  $\kappa_j, \lambda_j, \mu_j, \nu_j$  are simulated. To handle the special case where only  $\kappa_0, \lambda_0$  are simulated, we add a state  $\perp$  to  $\mathcal{A}$  with transitions  $\perp \xrightarrow{(a,b)} q$  for all symbols  $(a, b) \in \Sigma_\perp^2$  of rank  $r$ , and  $q \in Q_\perp^r$ . The B  chi tree automaton  $\mathcal{B}$  has the state set  $Q_\perp^4$ , initial state  $(q_{\text{in}}, q_{\text{in}}, \perp, \perp)$ , and the transitions

- $(p, s, q, t) \xrightarrow{(a,b)} p \otimes s \otimes q \otimes t$  if  $\mathcal{A}$  contains the transitions  $p \xrightarrow{(b,b)} p, s \xrightarrow{(a,b)} s, q \xrightarrow{(\perp,b)} q, t \xrightarrow{(\perp,a)} t$ ,
- $(p, q, q, \perp) \xrightarrow{\#} (p, p, q, q)$  for all  $p, q \in Q$ .

Correctness follows from the previous observations.  $\square$

## C Proof of Proposition 6.1

**Lemma C.1.** *The infinite clique problem is logspace reducible to recurrent reachability over (tree-)regular relations.*

*Proof.* The word case is easy. Let  $R \subseteq \Sigma^* \times \Sigma^*$  be given by an NFA  $\mathcal{A}$ . We have that  $R$  has an infinite clique if and only if there exists a sequence  $(w_i)_{i \geq 1}$  of words such that  $(w_i, w_j) \in R$  and  $|w_i| < |w_j|$  for all  $1 \leq i < j$ . We define the relation  $R' \subseteq \Sigma^* \times \Sigma^*$  such that

- $(\varepsilon, w) \in R'$  for all  $w \in \Sigma^+$  and
- $(v, w) \in R'$  iff  $(v, w) \in R$  and  $|v| < |w|$  for all  $v, w \in \Sigma^+$ .

Clearly, the relation  $R'$  is regular and an NFA that accepts  $R'$  is implicitly logspace computable. Since a path in  $R'$  cannot visit a word more than once, it holds that  $R$  has an infinite clique if and only if  $\varepsilon \in \text{Rec}(\Sigma^*)[R']$ .

In the tree case we use a similar idea as in the word case for one path of the trees. Let  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  be given by an NTA  $\mathcal{A}$ . We have that  $R$  has an infinite clique if and only if there exists a sequence  $(t_i)_{i \geq 1}$  of trees such that  $(t_i, t_j) \in R$  for all  $1 \leq i < j$  and the domain of one path strictly grows indefinitely in the sequence. Such a sequence exists since the union of the domains of all  $t_i$  is an infinite ranked tree domain which by K  nig's lemma has an infinite path. Let  $t_0: \{\varepsilon\} \rightarrow \Sigma$  with  $t_0(\varepsilon) := a$  for some  $a \in \Sigma$ . We define the relation  $R' \subseteq \mathcal{T}_{\Sigma'} \times \mathcal{T}_{\Sigma'}$  with  $\Sigma' = \Sigma \cup \{a' \mid a \in \Sigma\}$  such that for all  $s, t \in \mathcal{T}_{\Sigma'} \setminus \{t_0\}$  we have

- $(t_0, t) \in R'$  and
- $(s, t) \in R'$  iff the non-primed versions of  $s$  and  $t$  are in relation in  $R$  and there exists exactly one path  $p$  from



the root to a leaf in  $s$  labeled with primed symbols, exactly one path  $q$  from the root to a leaf in  $t$  labeled with primed symbols, and the domain of  $p$  is a strict subpath of the domain of  $q$ .

The relation  $R'$  is tree-regular and an NTA for  $R'$  is implicitly logspace computable. Indeed, the NTA for  $R'$  nondeterministically guesses the path of primed labels in the convolution and verifies that all other paths have non-primed labels. Moreover, it can check if the path in  $s$  is padded and therefore a strict subpath of the path in  $t$ . Finally, it can simulate the automaton  $\mathcal{A}$  for  $R$  on the non-primed versions of  $s$  and  $t$  by just treating a symbol  $a'$  as  $a$ .

We claim that  $R$  has an infinite clique if and only if  $t_0 \in \text{Rec}(\mathcal{T}_\Sigma)[R']$ . We already argued the “only if” direction. For the “if” direction note that the path property prevents the witnessing sequence from visiting the non-primed version of a tree more than once.  $\square$

**Lemma C.2.** *Recurrent reachability is logspace reducible to the infinite clique problem over (tree-)regular relations.*

*Proof.* We use the same idea for both the word and tree case. Let  $R \subseteq A \times A$  be a (tree-)regular relation given by an NFA (resp. NTA)  $\mathcal{A}$  and  $L \subseteq A$  be a (tree-)regular language given by an NFA (resp. NTA)  $\mathcal{B}$ . Furthermore, let  $a_0 \in A$  be the initial word (resp. tree). We define the relation  $R' \subseteq (A \times \mathbb{N}) \times (A \times \mathbb{N})$  such that  $((a, m), (b, n)) \in R'$  if and only if

- $(a_0, a) \in R$ ,
- $(a, b) \in R$ , and
- $a \in L$ .

Intuitively, we create infinitely many copies of every word (resp. tree) by taking the direct product with the integers. This allows the witnessing sequence of the infinite clique to visit a word (resp. tree) several times. Furthermore, in  $R'$  we only consider the words (resp. trees) that are in relation with  $a_0$  to ensure that  $a_0$  fulfills the conditions of the initial word (resp. tree). With the third condition we ensure that every word (resp. tree) of the infinite clique is contained in  $L$ . Thus,  $a_0 \in \text{Rec}(L)[R]$  if and only if  $R'$  has an infinite clique.

Note that  $R'$  is (tree-)regular and an NFA (resp. NTA) for it is implicitly logspace computable. To this end, we represent the integers in unary as words (resp. paths) and take the convolution with the corresponding word (resp. tree). The first condition can be checked by hardwiring  $a_0$  into the automaton. The second and third conditions can be ensured by simulating  $\mathcal{A}$  and  $\mathcal{B}$ , respectively.

Note that if  $R$  is transitive, then so is  $R'$ . Moreover, if  $\mathcal{A}$  and  $\mathcal{B}$  are deterministic, then so is the automaton for  $R'$ .  $\square$

## D Recurrent reachability with generalized Büchi condition

**Proposition D.1.** *Recurrent reachability with generalized Büchi condition is decidable in polynomial space over words.*

*Proof.* We give a PSPACE-reduction from the generalized version to the classical version. Let the relation  $R \subseteq \Sigma^* \times \Sigma^*$  be given by an NFA  $\mathcal{A}$ , the languages  $L_1, \dots, L_k \subseteq \Sigma^*$  be given by NFAs  $\mathcal{A}_1, \dots, \mathcal{A}_k$ , and  $s_0 \in \Sigma^*$  be the initial word. First observe that  $s_0 \in \text{Rec}(L_1, \dots, L_k)[R]$  if and only if there is a sequence of words  $(s_i)_{i \geq 1}$  such that  $(s_i, s_j) \in R$  for all  $0 \leq i < j$  and  $s_i \in L_{((i-1) \bmod k)+1}$  for all  $i \geq 1$ . We define the relation

$$R_i := \{(w_1, \dots, w_{2k}) \in (\Sigma^*)^{2k} \mid w_i \in L_i\}$$

for all  $i \in [1, k]$ . Moreover, for all  $1 \leq i < j \leq 2k$  let

$$R_{i,j} := \{(w_1, \dots, w_{2k}) \in (\Sigma^*)^{2k} \mid (w_i, w_j) \in R\}.$$

Finally, we define the relation

$$R_{s_0,i} := \{(w_1, \dots, w_{2k}) \in (\Sigma^*)^{2k} \mid (s_0, w_i) \in R\}$$

for all  $i \in [1, k]$ . Then  $s_0 \in \text{Rec}(L_1, \dots, L_k)[R]$  if and only if

$$\varphi := \exists^{\text{ram}} x, y: R'(x, y) \vee \exists x: R'(x, x)$$

is valid where

$$R' := \bigcap_{i=1}^k R_i \cap \bigcap_{1 \leq i < j \leq 2k} R_{i,j} \cap \bigcap_{i=1}^k R_{s_0,i} \subseteq (\Sigma^*)^k \times (\Sigma^*)^k.$$

Note that the NFAs for  $R_i$ ,  $R_{i,j}$ , and  $R_{s_0,i}$  can be easily constructed in polynomial time. Hence, the product automaton  $\mathcal{A}'$  that recognizes  $R'$  can be constructed in PSPACE. By [Theorem 2.2](#) validity of the first disjunct of  $\varphi$  can be checked in nondeterministic logspace given  $\mathcal{A}'$ . It is easy to see that validity of the second disjunct of  $\varphi$  can also be checked in nondeterministic logspace given  $\mathcal{A}'$ . This yields a PSPACE-algorithm in total.  $\square$

**Proposition D.2.** *Recurrent reachability with generalized Büchi condition is decidable in exponential time over trees.*

*Proof.* We proceed similarly to the word case but we use ATAs to avoid the exponential blow-up for the product automaton. Let the relation  $R \subseteq \mathcal{T}_\Sigma \times \mathcal{T}_\Sigma$  be given by an ATA  $\mathcal{A} = (Q, \Sigma_\perp^2, \delta, q_0)$ , the languages  $L_1, \dots, L_k \subseteq \mathcal{T}_\Sigma$  be given by ATAs  $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, q_0^i)$  for all  $i \in [1, k]$ , and  $s_0 \in \mathcal{T}_\Sigma$  be the initial tree. Note that we may assume that the relation and the languages are given by alternating automata since an NTA can be easily converted into an ATA in polynomial time. Let  $r := \max\{\text{rk}(a) \mid a \in \Sigma\}$  and  $\Omega := \Sigma_\perp \cup \{\#_i \mid 0 \leq i \leq r\}$  be a new ranked alphabet with  $\text{rk}(a) := 1$  for all  $a \in \Sigma_\perp$  and  $\text{rk}(\#_i) := i$  for all  $i \in [1, r]$ . For trees  $t_1, \dots, t_n \in \mathcal{T}_\Sigma$  we define  $p(t_1, \dots, t_n) \in \mathcal{T}_\Omega$  to be the tree  $t_1 \otimes \dots \otimes t_n$  where each node labeled with  $(a_1, \dots, a_n) \in \Sigma_\perp^n$  is replaced by a path  $a_1(a_2(\dots a_n(\#_m) \dots))$  where  $m := \max\{\text{rk}(a_i) \mid 1 \leq i \leq n\}$ .

Let

$$R_p := \{(s, t) \in \mathcal{T}_\Omega^2 \mid s = p(t_1, \dots, t_k) \wedge t = p(t_{k+1}, \dots, t_{2k})\}$$

be the binary relation that checks if the trees are images under  $p$ . Note that an ATA for  $R_p$  can easily be constructed. We define ATAs recognizing relations  $R_i$  for all  $i \in [1, k]$

and  $R_{i,j}$  for all  $1 \leq i < j \leq 2k$  with a similar meaning as in the word case. We start with the construction of the ATA

$$\mathcal{B}_i = (Q_i^{\mathcal{B}}, \Omega^2, \delta_i^{\mathcal{B}}, (q_0, 0))$$

for  $R_i$ . Intuitively,  $\mathcal{B}_i$  checks if in  $p(t_1, \dots, t_k) \otimes p(t_{k+1}, \dots, t_{2k})$  we have that  $t_i \in L_i$  for all  $i \in [1, k]$ . The set of states of  $\mathcal{B}_i$  is defined as

$$Q_i^{\mathcal{B}} := Q_i \times \{0, \dots, i-1\} \cup Q_i \times \Sigma \times \{i, \dots, k\}.$$

For all  $q \in Q_i$ ,  $j \in [0, k]$ ,  $a, b \in \Sigma_{\perp}$ , and  $c \in \Sigma$  we let

$$\begin{aligned} \delta_i^{\mathcal{B}}((q, j), (\frac{a}{b})) &:= ((q, j+1), 1), \text{ if } j \leq i-2 \\ \delta_i^{\mathcal{B}}((q, i-1), (\frac{c}{b})) &:= ((q, c, i), 1) \\ \delta_i^{\mathcal{B}}((q, c, j), (\frac{a}{b})) &:= ((q, c, j+1), 1), \text{ if } i \leq j < k \end{aligned}$$

and for all  $r_1, r_2 \in [0, r]$  with  $\text{rk}(c) \leq \max\{r_1, r_2\}$  let

$$\delta_i^{\mathcal{B}}((q, c, k), (\frac{\#r_1}{\#r_2})) := \delta'_i(q, c)$$

where  $\delta'_i(q, c)$  is the formula  $\delta_i(q, c)$  in which each variable  $(p, \ell)$  is replaced by  $((p, 0), \ell)$ .

We now construct the ATA

$$\mathcal{A}_{k_1, k_2} = (Q_{k_1, k_2}, \Omega^2, \delta_{k_1, k_2}, (q_0, 0))$$

for  $R_{k_1, k_2}$ . Intuitively,  $\mathcal{A}_{k_1, k_2}$  checks if  $(t_{k_1}, t_{k_2}) \in R$  holds in  $p(t_1, \dots, t_k) \otimes p(t_{k+1}, \dots, t_{2k})$ . We only show the construction for the case  $1 \leq k_1 \leq k < k_2 \leq 2k$  and  $k_1 < k_2 - k$  and note that the other cases work analogously. The set of states of  $\mathcal{A}_{k_1, k_2}$  is defined as

$$\begin{aligned} Q_{k_1, k_2} &:= Q \times \{0, \dots, k_1 - 1\} \cup \\ &\quad Q \times \Sigma_{\perp} \times \{k_1, \dots, k_2 - k - 1\} \cup \\ &\quad Q \times \Sigma_{\perp}^2 \times \{k_2 - k, \dots, k\}. \end{aligned}$$

We now define the transition function. For all  $q \in Q$ ,  $j \in [0, k]$ , and  $a, b, c, d \in \Sigma_{\perp}$  we let

$$\delta_{k_1, k_2}((q, j), (\frac{a}{b})) := ((q, j+1), 1)$$

if  $j \leq k_1 - 2$ ,

$$\begin{aligned} \delta_{k_1, k_2}((q, k_1 - 1), (\frac{a}{b})) &:= ((q, a, k_1), 1) \\ \delta_{k_1, k_2}((q, c, j), (\frac{a}{b})) &:= ((q, c, j+1), 1) \end{aligned}$$

if  $k_1 \leq j \leq k_2 - k - 2$ ,

$$\begin{aligned} \delta_{k_1, k_2}((q, c, k_2 - k - 1), (\frac{a}{b})) &:= ((q, (\frac{c}{b}), k_2 - k), 1) \\ \delta_{k_1, k_2}((q, (\frac{c}{d}), j), (\frac{a}{b})) &:= ((q, (\frac{c}{d}), j+1), 1) \end{aligned}$$

if  $k_2 - k \leq j \leq k - 1$ , and for all  $r_1, r_2 \in [0, r]$  with  $\text{rk}(c), \text{rk}(d) \leq \max\{r_1, r_2\}$  let

$$\delta_{k_1, k_2}((q, (\frac{c}{d}), k), (\frac{\#r_1}{\#r_2})) := \delta'(q, (\frac{c}{d}))$$

where  $\delta'(q, (\frac{c}{d}))$  is the formula  $\delta(q, (\frac{c}{d}))$  in which each variable  $(p, \ell)$  is replaced by  $((p, 0), \ell)$ .

The ATA for the relation  $R_{s_0, i}$  with  $i \in [1, k]$  that checks if in  $p(t_1, \dots, t_k) \otimes p(t_{k+1}, \dots, t_{2k})$  we have that  $(s_0, t_i) \in R$  can be constructed similarly to  $\mathcal{A}_{i, j}$ . Note that all the constructions above can be done in polynomial time.

It now holds that  $s_0 \in \text{Rec}(L_1, \dots, L_k)[R]$  if and only if  $\varphi := \exists^{\text{ram}} x, y: R'(x, y) \vee \exists x: R'(x, x)$  is valid where

$$R' := R_p \cap \bigcap_{i=1}^k R_i \cap \bigcap_{1 \leq i < j \leq 2k} R_{i, j} \cap \bigcap_{i=1}^k R_{s_0, i}.$$

Since an ATA for the intersection of two ATAs can be constructed in linear time, we can construct an ATA  $\mathcal{A}'$  for  $R'$  in time polynomial in the size of the ATAs  $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_k$  and  $s_0$ . By [Theorem 2.4](#) validity of the first disjunct of  $\varphi$  can be checked in time exponential in the size of  $\mathcal{A}'$ . It is easy to see that validity of the second disjunct of  $\varphi$  can also be checked in time exponential in  $\mathcal{A}'$ . This yields an exponential time algorithm in total.  $\square$

## E Lower bounds for monadic decomposability

**Lemma E.1.** *Given a binary regular relation  $R \subseteq \Sigma^* \times \Sigma^*$  by an NFA (resp. DFA), it is PSPACE-hard (resp. NL-hard) to decide whether  $R$  is monadically decomposable. Given a binary tree-regular relation  $R \subseteq \mathcal{T}_{\Sigma} \times \mathcal{T}_{\Sigma}$  by an NTA (resp. D  TA), it is EXP-hard (resp. P-hard) to decide whether  $R$  is monadically decomposable.*

*Proof.* We give a logspace reduction from the universality problem which is known to be PSPACE-complete for NFAs, NL-complete for DFAs, P-complete for D  TAs, and EXP-complete for NTAs. In the following we only consider the word case. Recall that the universality problem asks whether for a given regular language  $L \subseteq \Sigma^*$  it holds that  $L = \Sigma^*$ . Let  $L \subseteq \Sigma^*$  be a regular language given by an NFA (resp. DFA)  $\mathcal{A}$ . We define the regular relation

$$R_L := \{(u \otimes v, w) \mid u \in L \text{ or } v = w \in \Sigma^*\}.$$

It is easy to construct an NFA (resp. DFA) that recognizes  $R_L$  in logarithmic space from  $\mathcal{A}$ . Note that for DFAs the disjunction can be realized with a product construction. It remains to show that  $R_L$  is monadically decomposable if and only if  $L = \Sigma^*$ .

If  $L = \Sigma^*$ , it holds that  $R_L = \{(u \otimes v, w) \mid u, v, w \in \Sigma^*\}$  which is clearly monadically decomposable.

For the converse assume that there exists  $u_0 \in \Sigma^* \setminus L$ . Then the intersection of  $R_L$  with the monadically decomposable relation  $\{(u_0 \otimes v, w) \mid v, w \in \Sigma^*\}$  is the relation  $\{(u_0 \otimes v, w) \mid v = w \in \Sigma^*\}$ , which is not monadically decomposable. Since monadically decomposable relations are closed under intersection, it follows that  $R_L$  is not monadically decomposable.  $\square$

**Lemma E.2.** *Given a binary tree-regular relation  $R \subseteq \mathcal{T}_{\Sigma} \times \mathcal{T}_{\Sigma}$  by a D  TA, it is P-hard to decide whether  $R$  is monadically decomposable.*

*Proof.* We give a logspace reduction from the emptiness problem for D  TAs which is known to be P-complete [\[32\]](#). Let  $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$  be a D  TA. We construct a D  TA

$\mathcal{A}' = (Q, \Gamma_1^2, \Delta', q_0)$  recognizing a binary tree-regular relation  $R'$  over  $\Gamma := \Sigma \cup \{\#\}$  where  $\# \notin \Sigma$  is a symbol of rank 1 as follows. We define the transition relation  $\Delta'$  such that

- $q \xrightarrow{(a,a)}_{\mathcal{A}'} (q_1, \dots, q_r)$  for all  $q \xrightarrow{a}_{\mathcal{A}} (q_1, \dots, q_r)$ ,
- $q_0 \xrightarrow{(\#, \#)}_{\mathcal{A}'} q_0$

Clearly,  $\mathcal{A}'$  can be constructed in lspace from  $\mathcal{A}$ .

It is easy to see that  $R' \subseteq \{(t, t) \mid t \in \mathcal{T}_\Gamma\}$ . Moreover, it holds that  $R'$  is finite if and only if  $L(\mathcal{A}) = \emptyset$ . Indeed, if there exists  $t \in L(\mathcal{A})$ , then  $(t_n, t_n) \in R'$  for all  $n \geq 0$  where  $t_n$  is the resulting tree when padding a chain of  $\#$ -symbols of length  $n$  to the root of  $t$ . Since every finite relation is recognizable and every infinite subrelation of  $\{(t, t) \mid t \in \mathcal{T}_\Gamma\}$  is clearly not recognizable, it holds that  $R'$  is recognizable if and only if  $L(\mathcal{A}) = \emptyset$ .  $\square$

## F Proofs of Section 7

A *nondeterministic unranked tree automaton* (NUTA) over the unranked alphabet  $\Sigma$  is a tuple  $\mathcal{A} = (Q, \Sigma, \Delta, q_0)$  where  $Q$  and  $q_0$  are as in the definition of an NTA and  $\Delta \subseteq Q \times \Sigma \times \text{REG}(Q)$  is a finite set of transitions. Here,  $\text{REG}(Q)$  denotes the set of regular word languages over  $Q$  and we assume that the regular language for each transition is given by an NFA. A *run* of  $\mathcal{A}$  on an unranked tree  $t \in \mathcal{U}_\Sigma$  is an unranked tree  $\rho \in \mathcal{U}_Q$  with  $\text{dom}(\rho) = \text{dom}(t)$  such that  $\rho(\varepsilon) = q_0$  and for each inner node  $u \in \text{dom}(\rho)$  with children  $u1, \dots, ur \in \text{dom}(\rho)$  there is a transition  $(\rho(u), t(u), L) \in \Delta$  such that  $\rho(u1) \cdots \rho(ur) \in L$ . A run  $\rho$  is *accepting* if for each leaf  $u \in \text{dom}(\rho)$  there is a transition  $(\rho(u), t(u), L) \in \Delta$  such that  $\varepsilon \in L$ . We define *unranked tree-regular relations* in the same way as in the ranked case by using NUTAs instead of NTAs.

### F.1 Proof of Theorem 7.1

**Definition F.1.** For an unranked tree  $t \in \mathcal{U}_\Sigma$  we define the *first-child next-sibling encoding*  $\text{fcns} : \mathbb{N}^* \rightarrow \{1, 2\}^*$  such that  $\text{fcns}(\varepsilon) = \varepsilon$  and for all  $u \in \mathbb{N}^*$  we have  $\text{fcns}(u1) = \text{fcns}(u)1$  and  $\text{fcns}(u(i+1)) = \text{fcns}(ui)2$  for all  $i \geq 1$ . We let  $\text{fcns}(\text{dom}(t)) := \bigcup_{u \in \text{dom}(t)} \{\text{fcns}(u), \text{fcns}(u)1, \text{fcns}(u)2\}$ . We define  $t' = \text{fcns}(t)$  to be the binary tree with domain  $\text{fcns}(\text{dom}(t))$  over the ranked alphabet  $\Sigma_\#$  such that

- $t'(\text{fcns}(u)) := t(u)$  if  $u \in \text{dom}(t)$ ,
- $t'(\text{fcns}(u)) := \#$  if  $\text{fcns}(u) \in \text{fcns}(\text{dom}(t))$  and  $u \notin \text{dom}(t)$ .

Here, we consider  $\Sigma_\# = \Sigma \cup \{\#\}$  as a ranked alphabet with  $\text{rk}(a) = 2$  for all  $a \in \Sigma$  and  $\text{rk}(\#) = 0$ .

**Definition F.2.** For unranked trees  $t_1, t_2 \in \mathcal{U}_\Sigma$  we define the adapted convolution  $t' = t_1 \otimes' t_2 \in \mathcal{U}_{\Sigma_\# \times \Sigma_\#}$  such that  $\text{dom}(t') := \text{dom}(t_1) \cup \text{dom}(t_2)$  and

- $t'(u) := (t_1(u), t_2(u))$  if  $u \in \text{dom}(t_1) \cap \text{dom}(t_2)$ ,
- $t'(u) := (t_1(u), \#)$  if  $u \in \text{dom}(t_1) \setminus \text{dom}(t_2)$  and there exists  $v \in \text{dom}(t_2)$  such that  $u$  is the first child or right sibling of  $v$ ,

- $t'(u) := (t_1(u), \perp)$  if  $u \in \text{dom}(t_1) \setminus \text{dom}(t_2)$  and the above conditions do not hold,
- the other cases are symmetric.

**Definition F.3.** For the convolution  $t = t_1 \otimes' t_2 \in \mathcal{U}_{\Sigma_\# \times \Sigma_\#}$  we define the adapted fist-child next-sibling encoding  $t' = \text{fcns}'(t_1 \otimes t_2) \in \mathcal{T}_{\Sigma_\# \times \Sigma_\#}$  such that  $\text{dom}(t') := \text{fcns}(\text{dom}(t))$  and for  $u' = \text{fcns}(u) \in \text{fcns}(\text{dom}(t))$  we have

- $t'(u') := t(u)$  if  $u \in \text{dom}(t)$ ,
- $t'(u') := (\#, \#)$  if  $u \notin \text{dom}(t)$  and for parent  $\text{fcns}(v)$  of  $u'$  we have  $t(v) \in \Sigma \times \Sigma$ ,
- $t'(u') := (\perp, \#)$  if  $u \notin \text{dom}(t)$  and for parent  $\text{fcns}(v)$  of  $u'$  we have  $t(v) \in \{\#, \perp\} \times \Sigma$ ,
- $t'(u') := (\#, \perp)$  if  $u \notin \text{dom}(t)$  and for parent  $\text{fcns}(v)$  of  $u'$  we have  $t(v) \in \Sigma \times \{\#, \perp\}$ .

Here, we consider  $\Sigma_{\#, \perp} = \Sigma_\# \cup \{\perp\}$  as ranked alphabet with  $\text{rk}(\perp) = 0$ .

See Figure 6 for an example of the adapted convolution and encoding.

**Lemma F.4.** Let  $R \subseteq \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma$  be an unranked tree-regular relation given by the NUTA  $\mathcal{A}$ , i.e.,  $R = \{(t_1, t_2) \in \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma \mid (t_1 \otimes t_2) \in L(\mathcal{A})\}$ . Then we can construct an NUTA  $\mathcal{A}'$  in polynomial time such that  $R = \{(t_1, t_2) \in \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma \mid (t_1 \otimes' t_2) \in L(\mathcal{A}')\}$ . That is,  $\mathcal{A}'$  uses the convolution  $\otimes'$  instead of  $\otimes$ . Conversely, we can also construct  $\mathcal{A}$  from  $\mathcal{A}'$  in polynomial time such that the above is satisfied.

*Proof.* To construct  $\mathcal{A}'$ , we adapt  $\mathcal{A}$  such that the conditions of Definition F.2 are satisfied. To this end, a state stores for each of the two components if it is the first child or right sibling of a node where the component is labeled by a symbol of  $\Sigma$ . Conversely,  $\mathcal{A}$  can be constructed by simply replacing  $\#$  in the transition of  $\mathcal{A}'$  by  $\perp$ .  $\square$

The next lemma shows that the connection between the adapted and classical notions of convolution and encoding suggested by Figure 6 holds true in general.

**Lemma F.5.** For unranked trees  $t_1, t_2 \in \mathcal{U}_\Sigma$  it holds that

$$\text{fcns}'(t_1 \otimes' t_2) = \text{fcns}(t_1) \otimes \text{fcns}(t_2).$$

*Proof.* The definitions of  $\otimes'$  and  $\text{fcns}'$  ensure that the padding symbol  $\#$  is used if the node would also be padded by  $\text{fcns}$  and otherwise the padding symbol  $\perp$  is used. The result follows since nodes at the same position in  $t_1$  and  $t_2$  are mapped to the same position in the encodings  $\text{fcns}(t_1)$  and  $\text{fcns}(t_2)$ .  $\square$

The following lemma shows that  $\text{fcns}$  preserves all properties of an unranked tree-regular relation and an NTA for the encoded relation over binary trees can be computed in polynomial time.

**Lemma F.6.** The encoding  $\text{fcns}$  is an isomorphism from a graph  $(\mathcal{U}_\Sigma, R)$  of unranked trees to a graph  $(\text{fcns}(\mathcal{U}_\Sigma), R')$  of binary trees where  $R$  is an unranked tree-regular relation

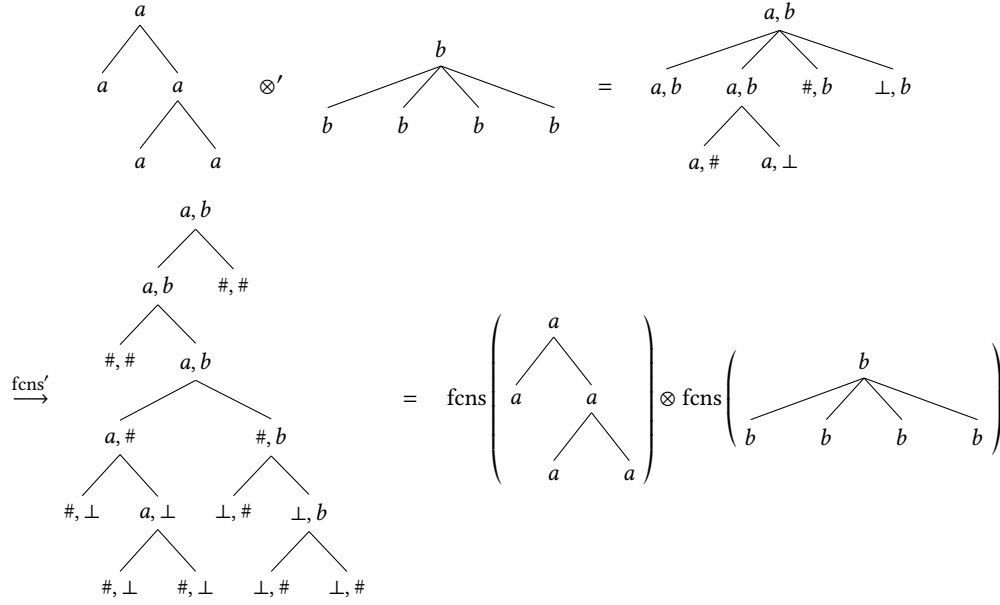


Figure 6. Example for adapted convolution and encoding

and  $R'$  is a tree-regular relation over binary trees. Moreover, an NTA that recognizes  $R'$  can be constructed in polynomial time given an NTA recognizing  $R$  and vice versa.

*Proof.* Let  $R \subseteq \mathcal{U}_\Sigma \times \mathcal{U}_\Sigma$  be an unranked tree-regular relation given by the NUTA  $\mathcal{A}$ . By Lemma F.4 one can construct an NTA  $\mathcal{A}'$  that recognizes  $R$  but uses  $\otimes'$  as convolution. From  $\mathcal{A}'$  we can construct an NTA  $\mathcal{B}'$  that accepts the language

$$\begin{aligned} & \{ \text{fcns}'(t_1 \otimes' t_2) \mid (t_1 \otimes' t_2) \in L(\mathcal{A}') \} \\ &= \{ \text{fcns}(t_1) \otimes \text{fcns}(t_2) \mid (t_1, t_2) \in R \} \end{aligned}$$

where the equality holds by Lemma F.5. The automaton  $\mathcal{B}'$  can be constructed in polynomial time in the usual way for the first-child next-sibling encoding (cf. [10]) but we additionally store in states if the label of the parent of the current node is in  $\Sigma \times \Sigma$ ,  $\{\perp, \#\} \times \Sigma$ , or  $\Sigma \times \{\perp, \#\}$ . Let  $R' := \{ (\text{fcns}(t_1), \text{fcns}(t_2)) \in \mathcal{T}_{\Sigma_\#} \times \mathcal{T}_{\Sigma_\#} \mid (\text{fcns}(t_1) \otimes \text{fcns}(t_2)) \in L(\mathcal{B}') \}$  be the tree-regular relation over binary trees recognized by  $\mathcal{B}'$ . Then it holds that  $(\text{fcns}(t_1), \text{fcns}(t_2)) \in R'$  iff  $(t_1, t_2) \in R$  for any unranked trees  $t_1, t_2 \in \mathcal{U}_\Sigma$ . Since the first-child next-sibling encoding is injective, it follows that  $\text{fcns}$  is an isomorphism from  $(\mathcal{U}_\Sigma, R)$  to  $(\text{fcns}(\mathcal{U}_\Sigma), R')$ .

The construction of an NTA for  $R$  given an NTA for  $R'$  works analogously using  $(\text{fcns}')^{-1}$  and the reverse direction of Lemma F.4.  $\square$

Note that we can generalize the above constructions from the binary to the  $n$ -ary case such that the same statements hold.

*Proof of Theorem 7.1.* By Lemma F.6 we can compute an NTA  $\mathcal{A}'$  that recognizes the relation

$$R' := \{ (\text{fcns}(t_1), \dots, \text{fcns}(t_{k+2})) \mid (t_1, \dots, t_{k+2}) \in R \}$$

over binary trees in polynomial time. Moreover, it holds that  $(\mathcal{U}_\Sigma, R)$  is isomorphic to  $(\text{fcns}(\mathcal{U}_\Sigma), R')$ . By Theorem 2.4 we can construct an NTA  $\mathcal{B}'$  that recognizes the relation

$$\{ c \in \text{fcns}(\mathcal{U}_\Sigma)^k \mid \exists^{\text{ram}} x, y: R'(x, y, c) \}$$

in exponential time given  $\mathcal{A}'$ . If  $R$  and therefore also  $R'$  are transitive, Theorem 2.5 implies that  $\mathcal{B}'$  can be computed in polynomial time. From  $\mathcal{B}'$  one can compute an NUTA  $\mathcal{B}$  that recognizes the relation

$$\begin{aligned} & \{ (\text{fcns}^{-1}(c_1), \dots, \text{fcns}^{-1}(c_k)) \in (\mathcal{U}_\Sigma)^k \mid \exists^{\text{ram}} x, y: R'(x, y, c) \} \\ &= \{ c \in (\mathcal{U}_\Sigma)^k \mid \exists^{\text{ram}} x, y: R(x, y, c) \} \end{aligned}$$

in polynomial time by applying the reverse direction of Lemma F.6.  $\square$

## F.2 Definition of subtree and flat prefix rewriting systems

For a tree  $t \in \mathcal{U}_\Sigma$  and node  $x \in \text{dom}(t)$  we write  $t_{\downarrow x}$  for the subtree of  $t$  rooted in  $x$ . We denote by  $t[x|s]$  the tree that is obtained from  $t$  if we replace  $t_{\downarrow x}$  by the tree  $s \in \mathcal{U}_\Sigma$ . If  $\text{ht}(t) = 1$ , we denote the sequence of leaves of  $t$  read from left to right by  $\text{flatfront}(t)$ . Here,  $\text{ht}(t) := \max\{|x| \mid x \in \text{dom}(t)\}$  is defined as the height of  $t$ .

**Definition F.7.** A subtree and flat prefix rewriting system (SFPRS) over unranked trees in  $\mathcal{U}_\Sigma$  is of the form  $\mathcal{R} =$



$(\Sigma, \Gamma, R, t_{\text{in}})$ , with a finite unranked alphabet  $\Sigma$ , a finite transition alphabet  $\Gamma$ , an initial tree  $t_{\text{in}}$ , and a finite set  $R$  of rules of two types:

1. subtree substitution with rules of the form  $r_j: s_j \xrightarrow{\sigma} s'_j$  for  $j \in J$ ,  $s_j, s'_j \in \mathcal{U}_\Sigma$ ,  $\sigma \in \Gamma$ , and
2. flat prefix substitution at the flat front of the tree with rules of the form  $r_i: u_i \xrightarrow{\sigma} u'_i$  for  $i \in I$ ,  $u_i, u'_i \in \Sigma^+$ ,  $\sigma \in \Gamma$ ,

with  $I \cup J = \{1, \dots, |R|\}$  and  $I \cap J = \emptyset$ .

A tree  $t'$  is derived from  $t$  (denoted  $t \rightarrow_{\mathcal{R}}^\sigma t'$ ) by applying a subtree rewrite rule  $r_j$ , if there is a node  $x \in \text{dom}(t)$  with  $t_{\downarrow x} = s_j$  such that  $t[x|s'_j] = t'$ .

A tree  $t'$  is derived from  $t$  by applying a prefix rewrite rule  $r_i$ , if there is a node  $x \in \text{dom}(t)$  with  $\text{ht}(t_{\downarrow x}) = 1$  and  $\text{flatfront}(t_{\downarrow x}) = u_i v$  and a tree  $s \in \mathcal{U}_\Sigma$  with  $\text{ht}(s) = 1$ ,  $s(\varepsilon) = t(x)$ , and  $\text{flatfront}(s) = u'_i v$  such that  $t[x|s] = t'$  for some  $v \in \Sigma^*$ .

The definition of an SFPRS can be extended to a *regular SFPRS* by allowing subtree rewrite rules of the form  $S_j \xrightarrow{\sigma} S'_j$  with unranked tree-regular languages  $S_j, S'_j \subseteq \mathcal{U}_\Sigma$  and prefix rewrite rules of the form  $L_i \xrightarrow{\sigma} L'_i$  with regular languages  $L_i, L'_i \subseteq \Sigma^*$ . Clearly, SFPRSs are special regular SFPRSs where the rules only have singleton sets.