Regular tree languages definable in FO and FO_{mod}

Michael Benedikt Luc Segoufin

Abstract

We consider regular languages of labeled trees. We give an effective characterization of the regular languages over such trees that are definable in first-order logic in the language of labeled graphs. These languages are the analog on trees of the "locally threshold testable" languages on strings. We show that this characterization yields a decision procedure for determining whether a regular collection of trees is first-order definable: the procedure is polynomial time in the minimal automaton presenting the regular language. We also give a characterization of definability in first-order logic supplemented with modular quantifiers, and a corresponding polynomial time decision procedure.

Keywords: Tree automata, Logic

1 Introduction

This paper is concerned with the relationship between regularity (acceptance by an automaton) and definability in first-order logic and first-order logic with counting quantifiers. Over strings this relationship is well-understood. A fundamental result in formal language theory [Bc60] states that a language of strings is regular – that is, equal to the language of strings accepted by a finite string automaton – exactly when it is definable in monadic-second-order logic (MSO) over the vocabulary consisting of the successor relation on strings and the labels. By restricting to first-order logic (FO) rather than MSO, we can obtain two proper subcollections of the family of regular languages. The languages that are definable in first-order logic over the transitive closure of the successor relation and the label predicates, which we denote FO($<$), are exactly the star-free or, equivalently, the aperiodic languages [MP71, Sch65]. The languages that are definable in first-order logic over the successor relation and the label predicates, which we denote by FO, correspond to locally threshold testable languages (see [Tho97]). Using a fundamental result of Thérien and Weiss [TW85], Beauquier and Pin [BP89] gave an algebraic characterization of the FO languages. They are exactly the languages for which the corresponding monoid satisfies one of a family of sets of identities. Put another way, they show that the monoids corresponding to FO-definable languages form a pseudo-variety within the collection of all finite monoids. Both the characterization of FO($<$)-definability via aperiodicity and the characterization of FO-definability of Beauquier and Pin lead to effective algorithms for checking whether a regular language is FO($<$) (resp. FO) definable. Straubing [Str94] provides an analogous characterization for the logic FO_{mod} that extends first-order logic with quantifiers that count modulo a given integer. A complete overview of the string case can be found in [Tho97] or in [Str94].

We now consider the situation over ranked trees – labeled trees with a fixed bound on branching. Regularity is now defined as acceptance by a (non-deterministic top-down or deterministic bottom-up) tree automaton, and regularity is shown to be equivalent to definability in monadic-second-order logic in the vocabulary of labeled graphs [Don70, TW68]. In this context we use FO($<$) to denote first-order logic over the labels and the transitive closure of the graph relation, FO to denote first-order logic over

the graph relation and the labels, and FO_{mod} to denote first-order logic with counting quantifiers (modulo an integer) over the graph relation and labels. The notions of aperiodicity and star-freeness have natural extensions to the tree context, but here FO($<$) is strictly weaker than aperiodicity and star-freeness [PT93, Heu91, Pot95]. Finding a decidable characterization of FO($<$) within the regular tree languages is a longstanding open problem; partial results (see below) are given in [EW03, BW04]. As in the string case, FO definability is known to be strictly weaker than FO($<$) definability, but surprisingly an effective characterization of FO-definability was also lacking. [Wil96] gives an algebraic characterization and decision procedure for the frontier testable languages, a subclass of the FO definable languages. [BW04] provides a decision procedure for two fragments of FO($<$) defined using existential path quantification; none of these fragments exactly matches the expressiveness of FO. [EW03] gives a characterization of the FO($<$) definable languages in terms of an algebraic structure (the "syntactic pre-clone") associated with the language; this characterization is not known to be effective. To our knowledge, the decidability of FO_{mod} definability was also unresolved over trees.

In this work we settle the issue of effective characterization of definability in FO and FO_{mod} over trees, ranked or unranked. Over ranked trees FO still corresponds to the Local Threshold Testable (LTT) languages, but this characterization does not yield a decision procedure. Our main result is an effective characterization of FO within the regular tree languages that uses a set of equivalences that preserve membership within the language. Unlike the string case, these equivalences include preconditions requiring portions of the tree to be similar "locally". They are thus a midpoint between a definition using locally threshold testability (which characterizes FO over ranked trees, but which is not effective) and a purely algebraic approach. Using a similar set of equivalences, we obtain a characterization of tree languages that are definable in FO_{mod} as well. We extend our characterizations to give characterizations of FO-definable and FO_{mod}-definable languages over unranked trees as well.

As an application of the characterization theorems, we show that over strings, our results yield a new proof of the algebraic characterization of LTT and of the decidability of membership in LTT over strings presented in [BP89, Str94]. The current proofs of the characterization of FO in the string case use either fundamental (and difficult) results in the theory of monoids [BP89] or difficult results within the theory of finite categories [Str94]. This paper provides an alternating proof of the string case using using techniques from finite model theory. This new proof has the advantage that it extends to trees. Nevertheless several of the technical lemmas remain identical in inspiration if not in notation to the earlier proofs.

We then show that our characterization theorem yields that one can decidable whether a regular language of trees is definable in FO and FO_{mod}, both over ranked and unranked trees. We show in fact that membership in these classes can be decided in polynomial time in the size of a minimal automaton accepting the regular language. In the case of unranked trees, we show that our decidability result applies to a slightly larger class of languages, the "extended regular" ones.

Organization: Section 2 gives the basic notation for this article. Section 3 states and proves the characterization theorem for FO in the case of ranked trees. Section 4 extends to prove the characterization in the unranked case. Section 5 provides a characterization of first-order logic supplemented with counting quantifiers over both ranked and unranked trees. Section 6 shows how the results for strings follow from the tree case and gives the decision procedures that follow from the characterization theorem. Section 7 gives conclusions and open issues.

This paper is a journal version of our STACS'05 paper [BS05]. It contains the full proofs of the decidability results claimed in [BS05]. The exact characterization claimed in [BS05], however, was incorrect.

2 Notation

Trees We fix a finite alphabet Σ. In this paper we will deal with two settings. In the *ranked* setting, we fix some integer r and consider Σ-labeled trees of rank r; that is, each node has at most r children. In this case, the children of any given node are *ordered*: that is, we can distinguish the first child, second child, and so forth. In the *unranked* setting there is no bound on the number of children and we will always take the children to be *unordered*. Finding a decidable characterization in the unranked ordered case remains an open question.

In both cases, we use standard notation for trees. By the *descendant* (resp. ancestor) relation we mean the reflexive transitive closure of the child (resp. inverse of child) relation. Given a tree t and a node x of t the subtree of t rooted at x, consisting of all the nodes of t which are descendants of x, is denoted by $t|_x$. Let t be a tree and x be a node of t, the *k-spill* of x, denoted by $N_k^t(x)$, is the restriction of $t|_x$ to the set of nodes of t at distance at most k from x. Given a tree t and a set S of nodes of t, the minimum subtree of t containing S is the smallest tree whose root r is the least common ancestor of all nodes of S and which contains all nodes of S and their ancestors up to r.

Given two nodes x and x' occurring respectively in the trees t, t' we say that x is depth-k similar to x' if the k-spill of x in t is isomorphic to the k-spill of x' in t'. Similarly two trees t and t' are depth-k similar if their roots are depth-k similar. When we are in the ranked case, isomorphism must preserve the order of children, but in the unranked case it need not.

A *context* is an (ordered or unordered) tree with a designated (unlabeled) leaf called its *port* which acts as a hole. Given contexts C and C', their concatenation $C \cdot C'$ is the context formed by identifying the root of C' with the port of C. Concatenation of context C and tree T is defined similarly. Given a tree t and two nodes x, y of t such that y is a descendant (not necessarily strict) of x, the context $C_t[x, y)$ is defined from $t_1 = t|_x$ by replacing $t_1|_y$ by a port.

Tree automata Regular tree languages will be represented by finite state automata. Over r-ranked trees, a (deterministic bottom-up) tree automaton A is defined in the usual way; it has a finite set of states Q, a finite set F of accepting states, and a transition function δ associating a unique state to any pair in $(\Gamma^i \times \Sigma)$ for $i \leq r$.

A tree automaton A over unordered unranked trees has a finite set of states Q, a finite set F of accepting states, and a transition function δ associating a unique state to any pair $(\Gamma^Q \times \Sigma)$ where $\Gamma = \{= i, \geq i \mid i \in \mathbb{N}\}$. We furthermore assume that if there are functions $f, g \in \Gamma^Q$ with f giving a more restrictive constraint than g on every $q \in Q$, then $\delta(f, a) = \delta(g, a)$; that is, A is deterministic. The transitions associate a unique state to any pair in $Q^i \times \Sigma$. The maximum number m occurring in the transition function is called the *tolerance* of A.

As usual a run τ of A on a tree t is a function from the set of nodes of t to Q. The notion of a valid run for ranked trees is standard. In the case of an unranked tree automaton, a run τ is valid if for any node x of label $a \in \Sigma$, there exists a function $f \in \Gamma^Q$ such that $\delta(f, a) = \tau(x)$ and, for every $q \in Q$, the number of children y of x such that $\tau(y) = q$, is consistent with $f(q)$. A tree t is accepted by A if the valid run τ of A on t is such that the image under τ of the root of t is in F. Languages accepted by such automata are called *regular languages*. It is folklore that this corresponds to the usual definition over ranked and unranked trees (see also [Tho97]).

An automaton A with set of states Q and a context C induce a function from Q to Q, sending a state q to the state q' reached by A at the root of C assuming state q at its port. For contexts C_1 and C_2, we write $C_1 \simeq C_2$ if they induce the same function.

Logics Monadic Second Order Logic (MSO) and First Order Logic (FO) are defined over trees in the standard way; in the case of r-ranked trees, they will be defined over the signature containing one unary predicate P_a per letter $a \in \Sigma$ and the tree successor relations $E_1 \dots E_r$, where $E_i(x, y)$ holds if y is the i^{th} child of x. In the case of unranked trees, they are defined over the signature containing one unary predicate P_a per letter $a \in \Sigma$, and the tree successor relation. A tree language is said to be *regular* if it is definable in MSO or, equivalently, recognized by a tree automaton.

For any formula $\varphi \in$ FO, its quantifier rank qr(φ) is defined as the nesting depth of the quantifiers of φ as usual. The elementary equivalence up to depth n is denoted by $\equiv^n$: for any two trees $t, t' \in \mathcal{T}$ we say that $t \equiv^n t'$ if t and t' satisfy exactly the same FO sentences of quantifier rank less than n.

Let P be a finite set of integers. The logic $\text{FO}_{mod(P)}$ extends FO by allowing formula to be built up by the rule $\psi(\vec{y}) = \exists^{r,q} x\ \phi(x, \vec{y})$, where $r < q$ are integers and $q \in P$. This holds in a structure $(G, \vec{y})$ iff the number of x such that $(G, \vec{y}, x)$ holds is equal to r modulo q.

3 Ranked trees

3.1 Statement of the main result

Swaps Let t be a tree, and x, x' be two nodes of t such that x and x' are not related by the descendant relationship. The *horizontal swap* of t at nodes x and x' is the tree t' constructed from t by replacing $t|_x$ with $t|_{x'}$ and vice-versa.

Let t be a tree of root r, and x, y, x', y' be four nodes of t such that y is a descendant of x, x' is a descendant of y and y' is a strict descendant of x'. The *vertical swap* of t between $[x, y)$ and $[x', y')$ is the tree t' constructed from t as depicted in Figure 1. More formally let $C = C_t[r, x)$, $\Delta_1 = C_t[x, y)$, $\Delta_2 = C_t[x', y')$, $\Delta = C_t[y, x')$, $T = t|_{y'}$. Then notice that $t = C \cdot \Delta_1 \cdot \Delta \cdot \Delta_2 \cdot T$. The tree t' is defined as $t' = C \cdot \Delta_2 \cdot \Delta \cdot \Delta_1 \cdot T$.

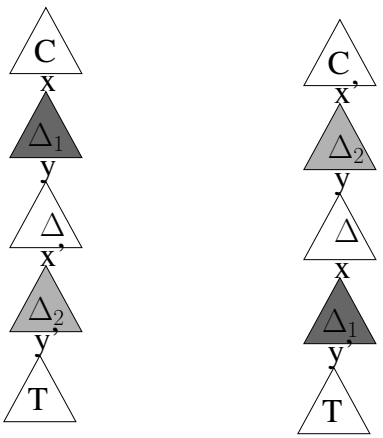


Figure 1: Illustration of the vertical swap

Guarded swaps Let $k \in \mathbb{N}$, $t \in \mathcal{T}$ and x, y, x', y' be nodes of t. A horizontal swap is said to be *k-guarded* if x and x' are depth-k similar. A vertical swap is said to be *k-guarded* if x and x' are depth-k similar and y and y' are depth-k similar.

Let L be a tree language and k be an integer. We say that L is *closed under k-guarded swaps* if for all tree $t \in L$ and all tree t' constructed from t by either a horizontal or a vertical k-guarded swap then t' is in L. It should be clear that being closed under k-guarded swaps implies being closed under k'-guarded swaps for $k' > k$.

In the following result, *aperiodicity* refers to the classical notion of aperiodicity in the monoid of contexts. A regular tree language L is said to be aperiodic if there exists $l \in \mathbb{N}$, the *aperiodicity number* of L, such that for all contexts C, Δ and all tree T, $C\Delta^l T \in L$ iff $C\Delta^{l+1}T \in L$.

Theorem 1. *Let L be a regular tree language.*

Then L is definable in FO iff L is aperiodic and there exists a k such that L is closed under k-guarded swaps.

One direction of Theorem 1 is easy: If L is definable in FO by a formula ϕ, then L is aperiodic [Tho97] and closed under k-guarded swaps where k is the locality rank of ϕ. The latter holds because k-guarded swaps preserve the number of k-neighborhoods of every isomorphism type.

The opposite direction follows from the following theorem, which will be quite involved:

Theorem 2. *For any regular tree language L which is aperiodic and closed under k-guarded swaps, there exists a K such that for any $t, t' \in \mathcal{T}$ we have:* $s \equiv^K t \quad \Rightarrow \quad s \in L$ *iff* $t \in L$

Before proving Theorem 2 we show how Theorem 1 follows from it. From Theorem 2 we know that if L is aperiodic and closed under k-guarded swaps then L is a union of equivalence classes of $\equiv^K$ for some K. Standard arguments from finite model theory (see e.g. [Lib04]) show that $\equiv^K$ has only finitely many equivalence classes and that each of them is definable in FO. Therefore L is definable in FO as a disjunction of such formulas for the corresponding classes.

3.2 Proof of Theorem 2

In this section we fix an aperiodic regular tree language L with l the aperiodicity number, an integer k and assume that L is closed under k-guarded swaps. We also fix a deterministic bottom-up tree automata A for L.

Because the trees are ranked there are only finitely many isomorphism types of trees of depth k that we denote by $\mathcal{T}_k$. Given a tree t and a node x of t we write $T^t_k(x)$ the isomorphism type of the k-spill of x in t, and denote it as the k-type of x (type of x when k is understood from the context). A crucial observation of the rest of the paper is that the $(k+1)$-type of a node implies the k-types of its children.

For each $\tau \in \mathcal{T}_k$ and any tree t we denote by $|t|_\tau$ the number of occurrences of the type τ in t. Given two trees s, t we denote by $s =^k_d t$ the fact that for all $\tau \in \mathcal{T}_k$, $|s|_\tau = |t|_\tau$ or $|s|_\tau, |t|_\tau > d$ (s and t have the same number of occurrences of type τ up to threshold d). If in addition for all $\tau \in \mathcal{T}_k$, $|s|_\tau \leq |t|_\tau$ then we write $s \leq^k_d t$. If for all $\tau \in \mathcal{T}_k$, $|s|_\tau = |t|_\tau$ then we write $s =^k_\infty t$. Note that if we apply a k-guarded swap move to a tree t, there is an obvious bijection from t to the resulting tree t' that preserves k-types; in particular, we have $t =^k_\infty t'$.

This first lemma is immediate from the locality of FO. It shows that we can assume that s and t have the same number of types up to some threshold.

Lemma 1. For each integer d, let $K_d = d + r^{k+1}$. Then $s \equiv^{K_d} t$ implies that $s =^{k+1}_d t$ and that s, t are depth-$(k+1)$ similar.

This second lemma refines the previous one by showing that not only can we assume that s and t have the same number of types up to some threshold, but that this number is always bigger in t than in s.

Lemma 2. For each integer d' there exists an integer d such that if $s =_d^{k+1} t$ then there exists t' such that $s \leq_{d'}^{k+1} t'$, and t, t' depth-$(k+1)$ similar, and $t' \in L$ iff $t \in L$.

Proof. Assume $s =_d^{k+1} t$ for some large enough d whose value will become apparent during the proof. Let β be the number of $(k+1)$-types τ such that $|t|_\tau < |s|_\tau$. We prove the lemma by induction on β. If $\beta = 0$ this is clear. Otherwise let τ be a $(k+1)$-type that occurs more times in s than in t. By hypothesis τ occurs at least d times in t.

Given two nodes x, y in a tree t with y a strict descendant of x, we write $\tau[x, y)$ for the number of nodes in the context $C_t[x, y)$ that have type τ. A *τ-skeleton of length* d_1 is a sequence $x_i : i \leq d_1$ such that x_{i+1} is a strict descendant of x_i and $\tau[x_i, x_{i+1}) \geq 1$ for each $i \leq d_1 - 1$.

We first show that for every d_1 there is d such that for every tree t', if $|t'|_\tau > d$ then there is a τ-skeleton of length d_1 in t'. Choose $d > (r+1)^{d_1}$. Starting with x_0 being the root of t', choose x_i inductively such that $C_t[x_{i-1}, x_i)$ has at least one node of type τ and x_i has at least $(r+1)^{d_1 - i}$ nodes of type τ below it. Suppose that we have $x_1 \dots x_i$. Let z_1 be a descendant of x_i of type τ having minimal depth. If there are no nodes of type τ in $C_t[x_i, z_1)$ (recall that z_1 is not in $C_t[x_i, z_1)$), then we know that there are at least $(r+1)^{d_1 - i}$ nodes satisfying τ below z_1, including z_1. Hence there is some child of z_1 having at least $(r+1)^{d_1 - (i+1)}$ nodes satisfying τ below it. Set x_{i+1} to be such a child. If there is some node of type τ in $C_t[x_i, z_1)$, then there is some node z_2 between x_i and z_1 which has more than one child having a node of type τ below it. One of the children of z_2 must have at least $(r+1)^{d_1 - (i+1)}$ nodes satisfying τ below it; choose x_{i+1} to be this node. We can verify in either case that this preserves the induction hypothesis.

We apply this to the tree t, for d_1 to be chosen later on, getting a τ-skeleton $x_i : i \leq d_1$. Let q be one more than the number of states of A. The context between x_i and x_{i+q}, for $i < d_1$ with $i = 0 \mod q$ is called a *section* of t. We say that a $(k+1)$-type ν is *safe* if $|t|_\nu \geq d'$. A τ-section is *safe* if it contains only safe $(k+1)$-types. Because the number of τ-sections is $(d_1/q) - 1$, we can choose d_1 big enough so that at least one of them is safe. Given this choice of d_1, fix x_i such that the context between x_i and x_{i+q} is safe. By the choice of q, there are distinct j and k with j, k between i and $i+q$ such that the automaton A run on t reaches the same state at x_j as at x_k. Hence we can replace the context between $C_t[x_j, x_k)$ with arbitrarily many copies of itself, without changing membership in L or adding copies of any unsafe type. Since $C_t[x_j, x_k)$ contains an occurrence of τ, by adding sufficiently many copies the resulting tree t' is indeed such that $t' =_{d'}^{k+1} t =_{d'}^{k+1} s$. Notice also that we have reduced β by one and we can conclude by induction. □

A tree t is *k-pseudo-included* in a tree t' if there is an injective mapping h from nodes of t to nodes of t' such that: (i) h preserves types in $\mathcal{T}_k$, and (ii) if x is the i^{th} child of y in t then $h(x)$ is a descendant of i^{th} child of $h(y)$ in t'. In this case the *h-pseudo-tree* is the minimum subtree of t' which contains $h(t)$.

The next step shows that we can also assume that s is pseudo-included in t. It requires only the closure of L under k-guarded swaps.

Lemma 3. If $s \leq_d^{k+1} t$ and s, t depth-$(k+1)$ similar then there exists t' such that s is $(k+1)$-pseudo-included in t', $t' =_\infty^{k+1} t$, and $t' \in L$ iff $t \in L$.

Proof. The proof is done by induction. We construct $t_1 \cdots t_n$ such that: t_1 is t and, for all $0 \leq i < n$, t_{i+1} is obtained from t_i using only k-guarded swaps, if s_i is the biggest prefix of s such that s_i is $(k+1)$-pseudo-included in t_i then there exists a node x_i of $s - s_i$, connected to s_i, such that $x \in s_{i+1}$. Moreover we require that all pseudo-inclusion mappings h constructed during this process send the root of s_i to the root of t_i. This implies the lemma as L is closed under k-guarded swaps.

Let r be the root of s and $\tau = T^s_{k+1}(r)$. By hypothesis the root r' of t is such that $T^t_{k+1}(r') = \tau$. Thus the prefix of s consisting of the root of s is $(k+1)$-pseudo-included in t_1 with the mapping $h(r) = r'$.

Assume now we have constructed t_i such that the maximum prefix s_i of s is $(k+1)$-pseudo-included in t_i by a mapping h such that $h(r) = r'$. If $s_i = s$ we are done. Otherwise let x be a node of s_i such that its p^{th} child y is not in s_i. Let s' be the prefix of s which contains s_i and y. We show how to transform t_i into t_{i+1} such that s' is $(k+1)$-pseudo-included in t_{i+1}.

Let $\tau = T^s_{k+1}(x)$, $\nu = T^s_{k+1}(y)$ and $x' = h(x)$. By hypothesis we know that there is a node y' in t_i outside of $h(s_i)$ such that $T^{t_i}_{k+1}(y') = \nu$. Let z' be the p^{th} child of x'. Note that z' cannot be in the h-pseudo-tree.

We distinguish several possibilities depending of the relative position of x' and y'. By maximality of s_i we know that y' is not below z'.

Assume first that y' is outside the h-pseudo-tree. Then it is either below x' or not related with x' by the descendant relationship. It is crucial here that $h(r) = r'$ as it rules out the case where y' occurs above $h(r)$. Because x and x' agree on their $(k+1)$-types, z' and y' are depth-k similar. We can apply the k-guarded horizontal swap to these two nodes. This yields the desired tree t_{i+1}, as we can now extend h by setting $h(y) = y'$. We can verify that this yields a $(k+1)$-pseudo-inclusion mapping, since the new $(k+1)$-type of y' remains ν.

Assume now that y' is inside the h-pseudo-tree. Let x_1 be the deepest node in s_i such that $x'_1 = h(x_1)$ is an ancestor of y', and x_2 be the highest node in s_i so that y' is an ancestor of $x'_2 = h(x_2)$. Note that the definition of pseudo-inclusion implies that x_2 is uniquely defined. Assume that x_2 is the j^{th} child of x_1 in s and let z'_1 be the j^{th} child of x'_1 in t'_i. Note that z'_1 cannot be in the image of h, since this would contradict the maximality of x_1. There are two cases two consider.

The first case is when x' is a descendant of x'_2 (see Figure 2). Because h preserves $(k+1)$-types, z'_1 and x_2 are depth-k similar and the same holds for y' and z'. We can thus apply the k-guarded vertical swap between $[z'_1, y')$ and $[x_2, z')$ and obtain the desired tree t_{i+1}. Indeed one can extend h by setting $h(y) = y'$. It remains to verify that it is indeed a $(k+1)$-pseudo-inclusion mapping. This is straightforward and left to the reader.

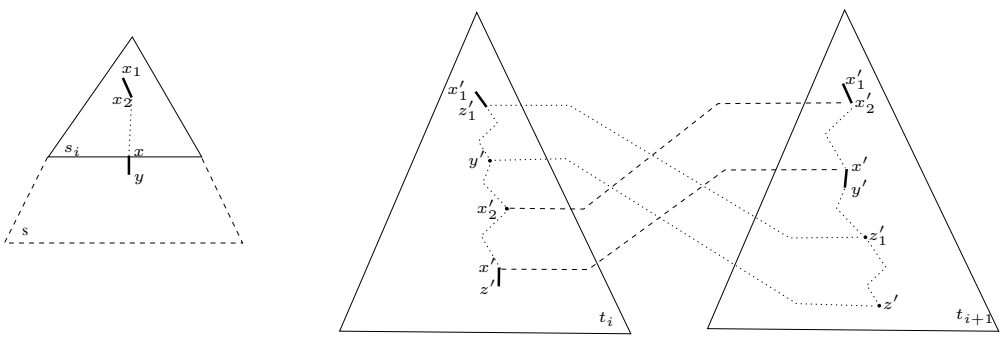


Figure 2: This illustrates the case when x' is a descendant of x'_2. s_i and s are depicted in the left. t_i is depicted in the middle. After applying the k-guarded vertical swap between $[z'_1, y')$ and $[x'_2, z')$ we reach t_{i+1} depicted to the right. The nodes z'_1 and x'_2 have the same k-types and the nodes y' and z' have the same k-types.

If x' is not a descendant of x'_2 we proceed as follows. As above we know that y' and z' are depth-k similar. If x' is a descendant of y', then it would have to be a descendant of x'_2 as well, since all

pseudo-tree elements beneath y' lie beneath x'_2. Hence we know x' cannot be a descendant of y', and so z' is not a descendant of y' either. We can therefore apply the k-guarded horizontal swap to y' and z', obtaining an intermediate tree t'_i. In t'_i, we have that x'_2 and z'_1 are depth-k similar and we can apply again the k-guarded horizontal swap to obtain the desired tree t_{i+1}. The mapping h is extended in order to send y to y' and it is immediate to see that this preserves $(k+1)$-types. □

An immediate corollary of Lemma 3 is:

Corollary 1. *If s and t are trees such that $s =^{k+1}_{\infty} t$ then $s \in L$ iff $t \in L$.*

Proof. Apply Lemma 3 to s and t and notice that the tree t' obtained is isomorphic to s via the $(k+1)$-pseudo-inclusion mapping h, as the hypothesis implies that t' cannot contain any extra node. □

In order to take care of the last step of the proof, which consists of inserting all extra sections of t' in s and conclude with Corollary 1, we will need the following lemma, which modifies Lemma 3, assuming nothing about the root types but assuming there is one extra copy of each neighborhood available, and concluding that there is a "tight embedding" of one tree into the other.

Given a tree t and a subtree U of some tree s, we say that U is *k-included* in t if there is a homomorphism (preserving the i^{th} child relation for every $i \leq r$) from U to t which also preserves k-types (where the k-type of a node in U refer to its k-type in s). We say that $U <_k t$ if the number of occurrences of each k-type in U is strictly less that the number of occurrences of the same k-type in t.

Lemma 4. Let s be a tree and U be a subtree of s.

If $U <_{k+1} t$ then there exists t' such that U is $(k+1)$-included into t', $t' =^{k+1}_{\infty} t$, and, $t' \in L$ iff $t \in L$.

Proof. The proof is similar to that of Lemma 3. It is done by induction and requires a lengthy case analysis.

The first step is to transform t into t'' such that there is a $(k+1)$-pseudo-inclusion of U into t''. Note that we cannot apply directly Lemma 3 as the hypothesis on the root types was crucial. In the proof of Lemma 3 this was reflected in the fact that if y' is not in the h-pseudo-tree then it cannot be above the image by h of the root of s. Without this the proof would not go through. However with our extra hypothesis on number of types, this case is no longer a problem.

Claim 1. *If $U <_{k+1} t$ then there exists a tree t'' such that U is $(k+1)$-pseudo-included in t'', $t'' =^{k+1}_{\infty} t$, and, $t'' \in L$ iff $t \in L$.*

Proof. The proof is done exactly as in the proof of Lemma 3 with the following differences. In the base case, the image of the root r of s is now an arbitrary node of $t = t_1$; an inclusion mapping does not demand preservation of the root. During the induction, having constructed t_i such that s_i is the maximum prefix of s which is $(k+1)$-pseudo-included in t_i, the case analysis now requires an extra case which we describe next.

Assume y' is an ancestor of $h(r)$. By hypothesis we know that there is a $y'' \neq y'$, outside of $h(s_i)$ whose $(k+1)$-type is also ν. If we redo the case analysis with y'' playing the role of y' we are left again with the case where both y' and y'' are ancestors of $h(r)$. Assume WLOG that y'' is a strict descendant of y'. Notice that z', y' and y'' are depth-k similar. We can apply the k-guarded vertical swap to $[y', y'')$ and $[y'', z')$. This yields the desired tree t_{i+1} as y' is now the p^{th} child of x'. Notice that the presence of y'' was crucial for this step. □

Using Claim 1 we can assume WLOG that U is $(k+1)$-pseudo-included in t.

Let Δ be the set of vertices x of U such that if y is the parent of x, $h(x)$ and $h(y)$ are not in a parent/child relation; note that the nodes in Δ do not necessarily form a subtree. Let $n = |\Delta|$ and $m = \Sigma_{x\in\Delta} d(x)$, where $d(x)$ denotes the depth in the tree of x. If $n = 0$ we already have a $(k+1)$-inclusion mapping and we are done. If not we show that it is possible to modify h and re-arrange t via swappings and obtain a new $(k+1)$-pseudo-inclusion for U mapping with $(n', m') < (n, m)$. By repeating this argument we eventually get a $(k+1)$-inclusion mapping of U into some tree t'.

Assume that $n \neq 0$ and take x and y such that $y \in \Delta$ is the p^{th} child of x, and consider $x' = h(x)$ and $y' = h(y)$. Let $\tau = T^s_{k+1}(x)$, $\nu = T^s_{k+1}(y)$. By assumption we know that there is another node y'' outside of the image $h(U)$, such that the $(k+1)$-type of y'' is ν. Let z' be the p^{th} child of x'. By assumption y', y'', and z' are all distinct nodes. Notice that z', y' and y'' are depth-k similar. We perform a case analysis depending on the relationship between z', y' and y''.

In the first case, we assume that y'' is an ancestor of z'. We apply the k-guarded vertical swap to $[y'', z')$ and $[z'y')$, obtaining a tree t_1. This case is depicted in Figure 3. Notice that U is still $(k+1)$-pseudo-included in t_1 and that y' is now the p^{th} child of x'. Hence n has decreased by one.

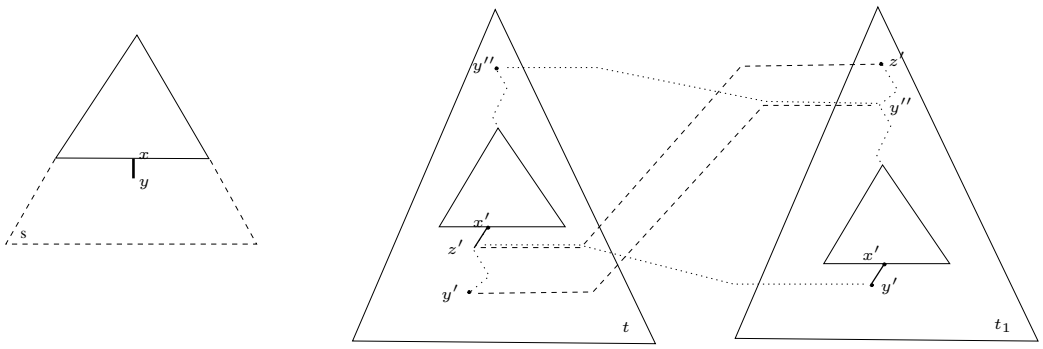


Figure 3: Illustration of the first case. U is depicted on the left, t in the middle and the resulting tree t_1 on the right. Notice that y' becomes a child of x' and that no other nodes of the pseudo-inclusion are affected.

In the second case, we assume that y'' is a descendant of y'. We apply the k-guarded vertical swap to $[z', y')$ and $[y', y'')$, with h being modified so that if before it maps some node w to a node w' that is swapped, it will now map w to the image of w' under the swap. Via this modification U remains $(k+1)$-pseudo-included in the new tree, and in this tree node y' is now the p^{th} child of $h(x)$, thus decreasing n by one.

In the third case, we assume that y'' is not related to z' and y'. We first apply k-guarded horizontal swapping to y'' and z' and again between y'' (i.e. the image of y'' under the previous swap: for brevity we omit this distinction henceforth) and y'. We then modify h by composing with the swap mappings, giving a $(k+1)$-pseudo-inclusion in the obvious way. Again we have connected $h(x)$ and y' and decreased n by one.

In the fourth case, we assume that y'' is between z' and y'. But then we can change h so that $h(y) = y''$ and still get a $(k+1)$-pseudo-inclusion of U into t' via the new h. We then proceed as in the second case.

The last case is when y'' is a descendant of z' but is not related to y'. If $\nu = \tau$ then let z'' be the p^{th} child of y' and notice that z', y' and z'' are depth-k similar. We apply the k-guarded vertical swap to $[z', y')$ and $[y', z'')$ and the reader can verify that we are done. If $\nu \neq \tau$ then by assumption we

know that there is a node x'' outside of $h(U)$ such that the $(k+1)$-type of x'' if τ. Let z'' be the p^{th} child of x''. Notice that z'', z', y' are depth-k similar. Again we have to consider several subcases.

In the first subcase z'' is not related to z'. We apply the k-guarded horizontal swap to z'' and z' followed by k-guarded horizontal swapping applied to z'' and y'. The reader can verify that this yields a tree t_1 with the desired properties: y' is now the p^{th} child of x' and the $(k+1)$-pseudo-inclusion mapping is only affected there, thus decreasing n by one.

In the second subcase z'' is a descendant of y'. Then we apply the k-guarded vertical swap to $[z', y')$ and $[y', z'')$ and obtain a tree where n is decreased by one.

The third subcase is when z'' is an ancestor of z'. Then we apply the k-guarded vertical swap to $[z'', z')$ and $[z', y')$. The reader can verify that this decreased n by one.

The fourth subcase is when z'' is below z' but not related to y'. We apply the k-guarded horizontal swap to z'' and y' attaching y' to x''. We now modify h by mapping x to x'' instead of x'. It is easy to verify that h is still $(k+1)$-pseudo-inclusion mapping for U. It is also easy to check that, with this new mapping, n does not increase: y is now longer in Δ but x is now in Δ, with no other nodes moving into Δ. We remark now that, with this new mapping, m has decreased by one.

The last subcase is when z'' is between z' and y'. Then z', z'' and y'' are related as in the subcase above and we proceed replacing y' with y''. □

We are now ready to complete the proof of Theorem 2. Given a tree t, a *loop* in t is a context U occurring in t such that the $(k+1)$-type of its root is also the $(k+1)$-type of the node attached to its port. Note that if h witnesses that s is $k+1$ pseudo-included in t and y is the p^{th} child of x in s, then the context rooted at the p^{th} child of $h(x)$ with $h(y)$ as a port is a loop in t.

Proof of Theorem 2:

Let Q be the set of states of A, $\alpha = |Q|$ be the number of states of A. Let $\beta_k = |\mathcal{T}_{k+1}|$. Let l be the aperiodicity number of L. Let $d' = r^{\beta_k * \alpha * l} + 1$. Let d be the number required in Lemma 2 for d'. Let K be the number K_d required in Lemma 1 for d. We show that $s \equiv^K t$ implies $s \in L$ iff $t \in L$.

Assume $s \equiv^K t$, we show that $s \in L$ iff $t \in L$. From Lemma 1 we know that $s =_d^{k+1} t$. Therefore by Lemma 2 there is a tree t' such that $t' \in L$ iff $t \in L$ and $t' \leq_{d'}^{k+1} s$. We can now apply Lemma 3 and obtain t'' such that $t'' \in L$ iff $t' \in L$, s is $(k+1)$-pseudo-included in t'' via a mapping h, and $t'' =_\infty^{k+1} t' =_{d'}^{k+1} s$. Therefore it suffices to prove that $s \in L$ iff $t'' \in L$.

By construction t'' is $h(s)$ plus possibly loops in t'' inserted between elements of $h(s)$. The crucial observation is that all $(k+1)$-types occurring outside of $h(s)$ have strictly more occurrences in t'' than in s and therefore appear at least d' times in s (and in t''). Let $U_1 \cdots U_n$ be the set of loops in $t'' \backslash h(s)$. We will "pump" s until the number of occurrences of each $(k+1)$-type exactly matches the number in t''. To achieve this, by induction, we construct $s_0 \cdots s_n$ such that: (i) s_0 is s, (ii) for all $n \geq i > 0$, for all $\tau \in \mathcal{T}_{k+1}$, $|s_i|_\tau = |s|_\tau + |U_1|_\tau + \cdots + |U_i|_\tau$, (iii) $s_i \in L$ iff $s_{i-1} \in L$.

The base case is immediate. Assume the result for $0 < i < n$, and consider $V = U_{i+1}$.

Let f_V be the transition function on states associated to the context V. The first step is to *minimize* the size of V: find V' such that the function associated to V' is f_V, V' uses the same $(k+1)$-types as V, and the size of V' is bounded by $r^{\beta_k * \alpha}$. This is a pumping argument. To find such a V', label each node of V with the pair (q, τ) where τ is the $(k+1)$-type of the node and q is the state reached by the automaton A recognizing L at this node. Now, whenever there is a branch in V which contains the same label twice, we prune the section from (and including) the top node to (and excluding) the bottom one, without affecting f_V. This yields a context V' whose depth is bounded by $\beta_k * \alpha$. As the rank of V' is bounded by r, the total size of V' is bounded by $r^{\beta_k * \alpha}$.

Consider the tree $U = V'^l$ which has size bounded by $r^{\beta * \alpha * l} = d' - 1$. By the remark above, each $(k+1)$-type of U occurs at least d' times in s and therefore in s_i, and by the choice of d' we can apply

Lemma 4 to U and s_i and obtain $s'_i = \Delta_1 \cdot U \cdot \Delta_2$ for some contexts Δ_1 and Δ_2, such that $s'_i \in L$ iff $s_i \in L$, and $s'_i =_\infty^{k+1} s_i$. We can now use the aperiodicity of L and without affecting membership in L obtain a tree $s''_i = \Delta_1 \cdot V'^l \cdot V' \cdot \Delta_2$. Notice now that if we set $s_{i+1} = \Delta_1 \cdot V'^l \cdot V \cdot \Delta_2$ we do not affect membership in L and have all the desired properties.

Let $s' = s_n$. By construction we have $s' =_\infty^{k+1} t''$ and $s' \in L$ iff $s \in L$. Theorem 2 now follows from Corollary 1.

4 Unranked trees

In this section we consider unranked trees. Each node may now have an arbitrary number of children. As mentioned in Section 2 we assume no order among the children of a node; in particular we cannot speak of the first child of a node.

The new difficulty of the unranked case, compared with the ranked case, is that the number of isomorphism types of a k-spill of a node is infinite. We therefore need to relax the notion of similarity. For any integer n we define an equivalence relation $\sim_n^k$ on trees of depth k by induction on k as follows. Let t and t' be two trees of depth k. Let r and r' their respective roots. In the case $k = 0$, $t \sim_n^0 t'$ if r and r' agree on their label. Otherwise $t \sim_n^k t'$ if, for each class $\mathbf{c}$ of $\sim_n^{k-1}$, the number of children of r in $\mathbf{c}$ must agree with the number of children of r' in $\mathbf{c}$ or both numbers must be bigger than n. It is immediate to see that, for each n, k, $\sim_n^k$ is of finite index. For each node x of a tree t, the $\sim_n^k$-equivalence class of its k-spill is called the (n, k)-*type* of x. When n and k are understood from the context we simply say the *type* of x. Let $UT_{n,k}$ be the (finite) set of (n, k)-types. If the (n, k)-type of node x in tree t is μ, we write $UT^t_{n,k}(x) = \mu$. For each $\tau \in UT_{n,k}$ we extend the notation $|t|_\tau$, $=_d^{n,k}$, $\leq_d^{n,k}$, and $\leq^{n,k}$ in the obvious way.

Two nodes x, y are said to be (n, k)-similar if they have the same (n, k)-type. Note that (n, k)-similar implies (n', k')-similar for all $n' < n$ and $k' < k$. Two trees are said to be (n, k)-similar if their roots are (n, k)-similar.

The (n, k)-guarded swaps are defined as in the ranked case, replacing depth-k similar with (n, k)-similar. Note that if L is closed under (n, k)-guarded swaps then it is closed under (n', k')-guarded swaps for all $n' > n$ and $k' > k$.

We first focus on proving the following result, from which our main theorem will follow easily:

Theorem 3. *Let L be a regular language over unranked trees.*

Then L is definable in FO iff L is aperiodic and there exists $n, k \in \mathbb{N}$ such that L is closed under (n, k)-guarded swaps.

As with the proof of Theorem 1, one direction is easy and the other follows from the following theorem.

Theorem 4. *For any regular tree language L which is aperiodic and closed under (n, k)-guarded swaps, there exists a K such that for any $t, t' \in \mathcal{T}$ we have: $s \equiv^K t \quad \Rightarrow \quad s \in L$ iff $t \in L$*

The proof of Theorem 4 will follow along the same lines as the proof of Theorem 2, but differs in the technical details.

We fix an aperiodic regular tree language L, integers k and n and assume that L is closed under (n, k)-guarded swaps. We also fix a deterministic bottom-up unranked tree automaton A for L. Let m be the tolerance of A. WLOG we can assume that $n \geq m$.

The following sequence of lemmas is the extension to the unranked setting of the same sequence of lemmas that we used in the ranked case. The first one is again immediate from the locality of FO.

Lemma 5. For each integers d, n', there exists $K_{d,n'}$ such that $s \equiv^{K_{d,n'}} t$ implies $s =_d^{n',k+1} t$ and s, t $(n', k+1)$-similar.

Lemma 6. For each integers d' and $n' > n$ there exists an integer d such that if $s =_d^{n',k+1} t$ then there exists t' such that $s \leq_{d'}^{n',k+1} t'$, and t, t' $(n', k+1)$-similar, and $t' \in L$ iff $t \in L$.

Proof. Assume $s =_d^{n',k+1} t$ for some large enough d whose value will become apparent during the proof. Let β be the number of $(n', k+1)$-types τ such that $|t|_\tau < |s|_\tau$. We prove the lemma by induction on β. If $\beta = 0$ this is clear. Otherwise let τ be a $(n', k+1)$-type that occurs more times in s than in t. By hypothesis τ occurs at least d times in t. We say that a $(n', k+1)$-type ν is *safe* if $|t|_\nu \geq d'$. A context in a tree t is safe if all the nodes in it have safe types.

First, let d_1 be big enough so that whenever we have d_1 distinct subtrees of t then we can find n' of them and a state q of A such that all the selected subtrees are safe and A reaches state q at the root of each. Let q be one more than the number of states in A, and let d_2 be big enough so that whenever one has $(d_2/q) - 1$ contexts there is at least one that is safe, Such a d_1 and d_2 can be easily computed from the size of A, d', n', and k. We now claim that any d bigger than $(d_1 + 1)^{d_2}$ will suffice.

A τ-skeleton of length d is defined as in the ranked case. We have two cases to consider. In case 1, every node in t has at most d_1 children which have a descendant of type τ. In this case, since we have more than $(d_1 + 1)^{d_2}$ τ nodes, we can use the same proof as in the ranked case to construct a τ-skeleton of length d_2 in t. As in the ranked case, the context between x_i and x_{i+q} in the τ-skeleton of t, including the top node and excluding the bottom one, is called a *section* of t. Using the same argument in the ranked case and the definition of d_2, we can pump a portion of some τ-section in the skeleton as much as we need to get the number of nodes satisfying type τ in t to be larger than the number in s.

In case 2, there is some node x in t that has more than d_1 children which have a descendant of type τ. By the choice of d_1 we can find a state q along with $y_1, \cdots, y_{n'}$ children of x so that the subtree rooted at each of the y_i is safe, contains a node of type τ, and evaluates to q. Because $n' > n > m$, the tolerance of A, it is possible to add an arbitrary number of extra copies of any of the subtrees rooted at y_i without affecting membership in L. Each copy adds at least one node of type τ, and we do this until we have enough nodes of type τ. □

We adapt the notion of pseudo-inclusion to the unranked case. A tree t is $(n, k+1)$-*pseudo-included* in a tree t' if there is an injective mapping h from nodes of t to nodes of t' such that: (i) h preserves $(n, k+1)$-types, (ii) if y is a child of x in t then $h(y)$ is a descendant of a child z' of $h(x)$ in t' such that z' and y have the same (n, k)-type (notice the switch from $k+1$ to k here), and (iii) if y_1 and y_2 are children of x in t then the least common ancestor of $h(y_1)$ and $h(y_2)$ in t' is $h(x)$ (the children of $h(x)$ associated to y_1 and y_2 are distinct). The *h-pseudo-tree* is the minimum subtree of t' which contains $h(t)$.

The following lemma takes care of the pseudo-inclusion step.

Lemma 7. For all d', n there exists n' such that if $s \leq^{n',k+1} t$ and s, t $(n', k+1)$-similar then there exists t' such that s is $(n, k+1)$-pseudo-included in t', $t' =_{d'}^{n,k+1} t$, and $t' \in L$ iff $t \in L$.

Proof. We say that a type $\tau \in UT_{n,k+1}$ is *safe* it it occurs more than d' times in t. A subtree is safe if it contains only safe types. Let n' be computed from d', n, k and m so that whenever one considers n' subtrees, then there exists a state q of A and at least m of the subtrees which are safe and for which the automaton reaches state q at the root.

As in the ranked case, the proof is done by induction. We construct $t_1 \cdots t_\alpha$ such that: t_1 is t, for all i, $t_{i+1} \in L$ iff $t_i \in L$, if s_i is the biggest prefix of s such that s_i is $(n', k+1)$-pseudo-included in t_i there exists a node x of $s - s_i$ so that $x \in s_{i+1}$, $t_{i+1} =_{d'}^{n,k+1} t_i$ and $t_i \leq_{d'}^{n'} t_{i+1}$. Moreover we require that all pseudo-inclusion mappings h constructed during this process send the root of s_i to the root of t_i.

Let r be the root of s and $\tau = UT^s_{n',k+1}(r)$. By hypothesis the root r' of t is such that $UT^t_{n',k+1}(r') = \tau$. Thus the prefix of s consisting of the root of s is $(n', k+1)$-pseudo-included in t_1.

Assume now that we have constructed t_i such that the maximum prefix s_i of s is $(n, k+1)$-pseudo-included in t_i by the mapping h. If $s_i = s$ we are done. Otherwise let x be a node of s_i such that its child y is not in s_i. Let s' be the prefix of s which contains s_i and y. We show how to transform t_i into t_{i+1} such that s' is pseudo-included in t_{i+1}.

Let $\tau = UT^s_{n',k+1}(x)$, $\nu = UT^s_{n',k+1}(y)$ and $x' = h(x)$. By hypothesis we know that there is a node y' in t_i outside of $h(s_i)$ such that $UT^{t_i}_{n',k+1}(y') = \nu$.

Let C be the set of children of x that have an image by h (in particular, y is not in C). Let C' be the set of children of x' having a descendant in $h(C)$. We distinguish two cases depending on whether there exists a child z' of x' which is (n', k)-similar to y' and which is not in C'.

If such a z' exists then we perform exactly the same case analysis as in the ranked case in order to replace z' by y'.

The main difference with the ranked case is that such a z' might not exist. In this case we show that we can expand the number of children of x', without affecting membership in L and violating the induction hypothesis, introducing a node z' (n', k)-similar to y'.

Let $\mu = UT^s_{n',k}(y)$ (note that we move from $k+1$ to k, therefore ν implies μ but not vice versa). Since x and x' have the same $(n', k+1)$-type, the number of children of x with (n', k)-type μ must agree with the number of children of x' of (n', k)-type μ or both numbers must be bigger than n'. Let C_μ be the subset of C consisting of children of x of type μ. Let C'_μ be the subset of C' of children of x' of type μ. By definition of $(n', k+1)$-pseudo-inclusion we have that $|C_\mu| = |C'_\mu|$. Because y is also of type μ and we are in the case where no child of x' outside of C' is of type μ this implies that the total number of children of x of type μ is strictly bigger than the total number of children of x' of type μ and therefore both numbers are bigger than n'. By the choice of n', this implies that we can find m children of x' of (n', k)-type μ such that their subtrees are safe and such that the automaton A reaches the same state q on all their subtrees. By the choice of m we can arbitrarily duplicate each of these subtrees without affecting membership in L. Because of the safety assumption the resulting tree t'_i is such that $t_i =_{d'}^{n,k+1} t'_i$. Because we have only introduced new nodes in t'_i it is clear that $t_i \leq^{n',k+1} t'_i$. This has introduced in t'_i a new node z' of type μ and we can proceed as in the first case. □

Given a tree t and a subtree U of some tree s, the notion of U being (n, k)-*included* into t is defined as expected by a mapping which not only preserves the (n, k)-types but also the child relation. We allow this relation to apply to forests as well as to trees. For forests s and t, we say that $s <_{n,k} t$ if every (n, k)-type occurring in s occurs at strictly more often in t.

Given an (n, k)-type τ, a node x satisfying τ is said to be (n, k)-*thin* if its k-spill can be embedded into k-spill of all nodes of type τ. Note that for each $\tau \in U\mathcal{T}_{n,k}$ there is a tree whose root satisfies τ and is (n, k)-thin: we choose the k-spill of the root so that when we calculating the (n, k)-type, we never go beyond n when counting number of occurrences at each depth.

We also have:

Lemma 8. Let s be a tree and U be a subforest of s.

If $U <_{n,k+1} t$ and each node $x \in U$, is $(n, k+1)$-thin, then there exists t' such that U is (n,k)-included in t', $t' =_{\infty}^{n,k+1} t$ and, $t' \in L$ iff $t \in L$.

Proof. As was the case for the ranked variant of this lemma, Lemma 4, the first step in the proof is a pseudo-inclusion of U in t, which is obtained by a refinement of Lemma 7.

Claim 2. *If $U <_{k+1} t$ and every node $x \in U$ is $(n, k+1)$-thin, then there exists a tree t'' such that U is $(n, k+1)$-pseudo-included in t'', $t'' =_{\infty}^{n,k+1} t$, and, $t'' \in L$ iff $t \in L$.*

Proof. The proof is of this claim is built upon the proof of Lemma 7 as the proof of Claim 1 is built upon the proof of Lemma 3. The extra case, which is necessary because we cannot necessarily map the root of U to the root of r is handled as in the proof of Claim 1. It remains to argue that the tree obtained does not contain new occurrences of $(n, k+1)$-types. Recall the proof of Lemma 7. The new occurrences of types were introduced only in the case where x' had less children than x of a given type. The assumption of thinness of x that we have in this claim guarantees that this is never the case. Therefore no new nodes are introduced. □

The rest of the proof proceeds exactly as in the ranked case. As in the proof of Claim 2, the thinness property guarantees that things behave as they do in the ranked case. □

We now have enough infrastructure to prove Theorem 4.

Proof of Theorem 4. Let m be the tolerance of A. Let $d' = (m + l) * \alpha + 1$ where α is spelled out in the proof below. Let n' be the number required in Lemma 7 from d', n and k, and d be the number required in Lemma 6 from d' and n. Let K be the number $K_{d,n'}$ required in Lemma 5 for d.

We show that $s \equiv^K t$ implies $s \in L$ iff $t \in L$. Assume $s \equiv^K t$. From Lemma 5 we know that $s =_d^{n',k+1} t$. By the choice of d we can apply Lemma 6 and construct t' such that $t' \in L$ iff $t \in L$ and $t' \leq_{d'}^{n',k+1} s$. We can now apply Lemma 7 and obtain t'' such that $t'' \in L$ iff $t' \in L$, s is $(n, k+1)$-pseudo-included in t'', and $t'' =_{d'}^{n,k+1} t' =_{d'}^{n,k+1} s$. Therefore it suffices to prove that $s \in L$ iff $t'' \in L$.

By construction t'' is $h(s)$ plus *loops* inserted between nodes of $h(s)$ and extra *branches* branching off the h-pseudo-tree. The crucial fact is that all $(n, k+1)$-types which occur outside of $h(s)$ have strictly more occurrences in t'' than in s and therefore appear at least d' times in s. We transform t'' without affecting membership in L in order to remove all the extra material.

We first show how to remove the extra branches. Let x' be a node of $h(s)$ and y' be a child of x' such that $V = t''|_{y'}$ is not in $h(s)$. Let τ be the (n,k)-type of y' and let C_τ be the set of children of x' of type τ in the h-pseudo-tree. Because h preserves $(n, k+1)$-types, C_τ is of cardinality at least n. Let V be the subtree rooted at y'. As in the ranked case, using simple pumping arguments, we can replace, without affecting membership in L, V by a tree V' having the following properties: (i) V' has exactly the same $(n, k+1)$-types as V, (ii) V' has all its nodes $(n, k+1)$-thin, and (iii) the number of nodes of V is bounded by a constant α easily computable from n, k and A. We use t'' to refer to this new tree as well. Let U be the forest consisting of m copies of V', let $T_{V'}$ be the tree constructed from t'' by removing V'. Notice all types mentioned in V' occur more frequently in t'' than in s, hence occur at least d' times in s and thus in $h(s)$. Since $T_{V'}$ still contains $h(s)$, we have that all the types in V' occur at least d' times in $T_{V'}$. By the choice of d', which is strictly greater than $|U|$, we can apply Lemma 8 to U and $T_{V'}$ obtaining a tree $T'_{V'}$ with m copies of V' $(n, k+1)$-included into it. Recall now the proof of Lemma 8 and notice that it uses only $(n, k+1)$-guarded swaps and therefore does not insert or remove any node and does not modify any $(n, k+1)$-type. During the transformation from $T_{V'}$ to $T'_{V'}$ we assume that each node of $T_{V'}$ has a unique identifier and we maintain a *log* table

LOG which contains a list of swap operations that have been performed. We can now use $(n, k+1)$-guarded swaps to place those m copies below x' (recall that $n > m$ and C_τ has size n, hence there are m children of x' to swap with). Those swaps are also recorded within our LOG table. We now look at our LOG and apply exactly the same transformations to t''. This does not affect membership in L and also does not affect any node of V'. This yields a new tree t_1 such that below x we have the m copies of V' constructed by the swaps plus our initial copy. We can now remove the initial copy without affecting membership in L and obtain $T'_{V'}$. We thus have two ways of obtaining $T'_{V'}$ via L-preserving moves: one from t'' and one from $T_{V'}$. This shows that $t'' \in L$ iff $T_{V'} \in L$. That is, we can remove V' from t'' without affecting membership in L.

We can now proceed with $T_{V'}$ in place of t'' and eventually remove all the branches which are not in $h(s)$ without affecting membership in L.

We remove loops in t'' in a similar way. Let V be a loop in t'' and V' be the reduct of V obtained as above. Apply Lemma 8 for $U = (V')^l$ and $T_{V'}$ (which is t'' minus the loop V') while bookkeeping the transformation into LOG. Using this LOG transform t'' in order to obtain U right above V'. Using aperiodicity of L the initial copy of V' can be safely removed and we proceed as above.

In the end, we have transformed t into a tree that is exactly $h(s)$, which was the desired conclusion. □

We will now show the main result:

Theorem 5. *Let L be a regular language over unranked trees. Then L is definable in FO iff L is aperiodic and there exists $k \in \mathbb{N}$ such that L is closed under k-guarded swaps.*

The theorem will follow immediately from Theorem 3 and the following proposition:

Proposition 1. *For every regular language L and every k there is an integer n such that if L is closed under k-guarded swaps, then it is closed under (n, k)-guarded swaps.*

Proof. To prove the proposition, we first show the following claim:

Claim 3. *For every regular language L there is an integer n such that for every k if tree t_1 is (n, k)-similar to tree t_2, then there are trees t'_1, t'_2 with t'_1 depth-k similar t'_2 such that A reaches the same state on t'_i as on t_i for $i = 1, 2$.*

Proof of the claim. Let m be the tolerance of A, and $n = m|Q|$. Notice that n is defined so that if there are n nodes, at least m of them will have the automaton A reach the same state. We show by induction on k that n suffices. For $k = 0$ it is clear. To prove this for $k+1$, consider trees t_1, t_2 with t_1 $(n, k+1)$-similar t_2. Let S be the set of (n, k)-types that are satisfied by some child of the root of t_1 or some child of the root of t_2. For each $\tau \in S$, let $t_1(\tau)$ be the number of satisfiers of τ by children of the root of t_1 and $t_2(\tau)$ be the number of satisfiers of τ by children of the root of t_2. If for every τ we have $t_2(\tau) = t_1(\tau)$. then we can let $t'_1 = t_1$, $t'_2 = t_2$ and we are done. For each τ for which this is not the case, we know that $t_2(\tau) > n$ and $t_1(\tau) > n$. Assume WLOG that $t_2(\tau) > t_1(\tau)$. By the definition of n, there are at least m τ-satisfying children of the root of t_1 for which the automaton reaches the same state. Pick one of these nodes x and add $t_2(\tau) - t_1(\tau)$ many copies of the subtree of x as children of the root of t_1. Since m is the tolerance of A, this does not affect the state of the root of t_1. Applying induction, we can change t_2 and t_1 without affecting the state at the root so that all the children of the root of t_2 satisfying τ are depth-k similar to one another, and each of these are depth-k similar to the children of the root of t_1 satisfying τ. Doing this for each $\tau \in S$ we end up with trees t'_1 and t'_2 with the property that: for every $\tau \in S$ all the children of the root of t_1 with type τ are depth-k similar to one another, they are all depth-k similar to every child of the root of t'_2 with type τ and the number of such children is the same in t'_1 and t'_2. It is clear that t'_1 and t'_2 are depth-$(k+1)$ similar.□

Given the above claim, we show how Proposition 1 follows. Suppose L is closed under k-guarded horizontal swaps. We show that L is closed under (n,k)-guarded horizontal swaps for n given by the claim. Given t_1 (n,k)-similar t_2 , let t_1', t_2' be as in the claim above. We know that:

$T[t_1,t_2] \in L \leftrightarrow T[t_1',t_2'] \in L \leftrightarrow T[t_2',t_1'] \in L \leftrightarrow T[t_2,t_1] \in L$

The first and third equivalences are because t_i' and t_i are equivalent in the automaton, and the middle equivalence is from closure under k-guarded swaps.

The proof for vertical swapping is similar; instead of the claim above, we show that: For every regular language L there is an integer n such that for every k if context C_1 is (n,k)-similar to context C_2, then there are contexts C_1', C_2' with C_1' depth-k similar C_2' such that C_i' induces the same state function as C_i for $i = 1, 2$.

Here two contexts are (n,k)-similar if they are (n,k)-similar as trees. The extension of the inductive argument for this is left to the reader. □

5 Modulo counting

In this section we obtain characterization of first-order logic with counting quantifiers. As before we start with the ranked case, to illustrate the main idea, and then we move to unranked trees. In [Str94] (See VII.3.1), the result of [BP89] is extended to a characterization of $\text{FO}_{mod(P)}$ on strings, by simply replacing the aperiodicity of L with a *periodicity* condition. Let q be lcm(P), rephrased on trees this yields: A language L is said to be q-periodic if,

(q-periodicity) $\quad \exists l$ such that $\forall s, u$ contexts, and $\forall t$ tree, $s \cdot u^l \cdot t \ \in L$ iff $s \cdot u^{l+q} \cdot t \ \in L$.

Fix $P \subseteq \mathbb{N}$ finite and let q be the lcm of all numbers occurring in P. The goal of this section is to show that replacing aperiodicity with q-periodicity in the theorems of the previous sections gives a characterization of $\text{FO}_{mod(P)}$.

5.1 Ranked trees and modulo counting

We start with the ranked tree case and fix the rank r for the whole section. Note first that for ranked trees, $\text{FO}_{mod(P)}$ is included in MSO (this is no longer the case for unranked trees). This is because over ranked trees, a linear order can be defined in MSO and therefore counting quantifiers can be simulated using this order. Thus $\text{FO}_{mod(P)}$ defines only regular languages.

Theorem 6. *Let L be a regular q-periodic tree language.*

Then L is definable in $\text{FO}_{mod(P)}$ *iff there exists a k such that L is closed under k-guarded swaps,*

Proof. The proof that q-periodicity is necessary is done as in [Str94]: one shows by structural induction that all $\text{FO}_{mod(P)}$ formula are q-periodic where free variables are treated as sentences in a product alphabet. The base case of atomic formulas follows from the result for FO, while the inductive cases are already shown in the proof of VII.3.1 of [Str94].

For the converse, we denote by $t \equiv_q^K t'$ the fact that t and t' agree on all sentences of $\text{FO}_{mod(P)}$ with at most K (first-order or modular) quantifiers. As in the case without modulo quantifiers, Theorem 6 follows immediately from:

Theorem 7. *For any regular language L which is q-periodic and closed under k-guarded swaps, there exists a K such that for any $t, t' \in \mathcal{T}$ we have:* $\quad t \equiv_q^K t' \quad \Rightarrow \quad t \in L$ *iff* $t' \in L$

□

Proof of Theorem 7. We follow the lines of the proof of Theorem 2. Fix L and q, k such that L is q-periodic and closed under k-guarded swaps. Fix a deterministic automaton A for L.

The notion of k-spill and depth-k similar is as in the ranked case for FO. Given two trees s, t we denote by $s =_d^{q,k} t$ the fact that for all $\tau \in \mathcal{T}_k$, $|s|_\tau = |t|_\tau$ or, $|s|_\tau, |t|_\tau > d$ and $|s|_\tau = |t|_\tau$ modulo q. If in addition for all $\tau \in \mathcal{T}_k$, $|s|_\tau < |t|_\tau$ then we write $s \leq_d^{q,k} t$. If for all $\tau \in \mathcal{T}_k$, $|s|_\tau = |t|_\tau$ then we write $s =_\infty^{q,k} t$.

The first two lemmas are an easy adaptation of Lemma 1 and Lemma 2 to the modulo counting case.

Lemma 9. For each integer d, let K_d be $d + r^{k+1}$ then $s \equiv_q^{K_d} t$ implies that $s =_d^{q,k+1} t$ and s, t depth-$(k+1)$ similar.

Lemma 10. For each integer d' there exists an integer d such that if $s =_d^{q,k+1} t$ then there exists t' such that $s \leq_{d'}^{q,k+1} t'$, and t, t' depth-$(k+1)$ similar, and $t' \in L$ iff $t \in L$.

Lemma 3 and Lemma 4 are proven using only swapping moves, without modifying the occurrences of $(k+1)$-types, and therefore we can make use of them in the modulo counting case. The same holds for Corollary 1. It therefore remains to revisit the last part of the proof of Theorem 2 and adapt it to the modulo counting case.

Let Q be the set of states of A, $\alpha = |Q|$ be the number of states of A. Let $\beta_k = |\mathcal{T}_{k+1}|$. Let l be the q-periodicity number of L. Let $d' = r^{\beta_k * \alpha * l} + 1$. Let d be the number required in Lemma 10 for d'. Let K be the number K_d required in Lemma 9 for d. We show that $s \equiv_q^K t$ implies $s \in L$ iff $t \in L$.

Assume $s \equiv^K t$, we show that $s \in L$ iff $t \in L$. From Lemma 9 we know that $s =_d^{q,k+1} t$. Therefore by Lemma 10 there is a tree t' such that $t' \in L$ iff $t \in L$ and $t' \leq_{d'}^{q,k+1} s$. We can now apply Lemma 3 and obtain t'' such that $t'' \in L$ iff $t' \in L$, s is $(k+1)$-pseudo-included in t'' via a mapping h, and $t'' =_\infty^{q,k+1} t' =_{d'}^{q,k+1} s$. Therefore it suffices to prove that $s \in L$ iff $t'' \in L$.

Once more following the proof of Theorem 2, we have by construction that t'' is $h(s)$ plus loops (of size larger than 1) inserted between nodes of $h(s)$. As before, all $(k+1)$-types which occur outside of $h(s)$ have strictly more occurrences in t'' than in s and therefore appear at least d' times in s (and in t''). Notice also that for each $(k+1)$-type τ the total number of occurrences of τ outside of $h(s)$ is zero modulo q. Let $V_1 \cdots V_n$ be the set of loops in $t'' \backslash h(s)$ and let V be the forest $\bigcup_{1 \leq i \leq n} V_i$. From the remark above we have $\forall \tau \in \mathcal{T}_{k+1}, |V|_\tau = 0$ modulo q.

Before continuing we need some additional definitions. A *j-pointed tree* is a tree with j designated leaves, which we call (generalizing the notation for contexts) ports. An "abstract j-pointed tree" is a j-pointed tree C supplemented with an assignment λ of $(k+1)$-types to each node such that the assignments are *consistent*: there is some tree T such that for every node x in C, the $(k+1)$-type of x in $C \cdot T$ matches $\lambda(x)$. An abstract context is an abstract 1-pointed tree, and an abstract loop is an abstract context such that λ assigns the same type to the root as to the port. With this notation the contexts $V_1, \cdots, V_n$ can be seen as abstract loops.

Lemma 11. There exists n' and abstracts loops $U_1 \cdots U_{n'}$, with each U_i having cardinality > 1, such that if $U = \bigcup_{1 \leq i \leq n'} U_i$ we have $\forall \tau \in \mathcal{T}_{k+1}, |U|_\tau = q.|V|_\tau$.

Proof. For each $(k+1)$-type τ and each k-type ν let $\alpha_\tau(\nu)$ be 1 if ν is induced by τ, 0 otherwise. Let $\beta_\tau(\nu)$ be the number of children of k-type ν that a node of $(k+1)$-type τ must have. For any multiset B of $k+1$-types, let $\alpha_B(\nu)$ be $\Sigma_{\tau \in B} |B(\tau)| \alpha_\tau(\nu)$, where $|B(\tau)|$ is the number of occurrences of τ in B. Let $\beta_B(\nu)$ be $\Sigma_{\tau \in B} |B(\tau)| \beta_\tau(\nu)$ and $\gamma_B(\nu)$ be $\alpha_B(\nu) - \beta_B(\nu)$.

Now let B be the multiset of all $(k+1)$-types occurring in V. Because V contains only abstract loops, we have that for each $\nu \in \mathcal{T}_k$, $\gamma_B(\nu) = 0$. Let B' be B where the multiplicity of each type has been divided by q. Note that each k-type is a union of the $(k+1)$-types that refine it, and for each $(k+1)$-type τ, $|B(\tau)|$ is a multiple of q. Using this we see that for each $\nu \in \mathcal{T}_k$, $\gamma_{B'}(\nu) = 0$. We will construct abstract loops $U_1 \cdots U_{n'}$ such that the multiset formed with the $(k+1)$-types of nodes of $\bigcup_{1 \leq i \leq n'} U_i$ is B'.

Assume we have already constructed n_1 abstract loops $U_1 \cdots U_{n_1}$ together with a "partially constructed loop" – an abstract j-pointed tree X which we hope to extend into an abstract loop. Let B_1 be the multiset of $(k+1)$-types occurring in $X \cup \bigcup_{1 \leq i \leq n_1} U_i$. Let $B_2 = B' \backslash (\bigcup_{1 \leq i \leq n_1} U_i)$.

If X is empty then we proceed by setting X to be a single node and setting its $(k+1)$-type to be an arbitrary type in $B' \backslash B_1$. If X is not empty and is not a loop, then there exists a leaf x of X, of $(k+1)$-type τ, which does not have the same $(k+1)$-type as the root of X. Let $\nu_1, \cdots, \nu_r$ be the induced k-types of the children of a node of $(k+1)$-type τ. Note that for each $j \leq n_1$ we have $\gamma_{U_j} = 0$, thus, by linearity of γ, $\gamma_{B_2} = 0$. Therefore we must be able to find in $B' \backslash B_1$ $(k+1)$-types $\tau_1, \cdots, \tau_r$ whose induced k-types are $\nu_1, \cdots, \nu_r$. We attach the corresponding elements of V as children of X and proceed by induction.

Let $U = \bigcup_{1 \leq i \leq n'} U_i$, by construction we have $\forall \tau \in \mathcal{T}_{k+1}, q.|U|_\tau = q.|B'|_\tau = |B|_\tau = |V|_\tau$ as required. □

Fix $U = \{U_1 \cdots U_{n'}\}$ as in Lemma 11. By induction we construct $s_0 \cdots s_{n'}$ such that: (i) s_0 is s, (ii) $\forall \tau \in \mathcal{T}_{k+1}$ $|s_i|_\tau = |s|_\tau + q.|U_1|_\tau + \cdots q.|U_i|_\tau$, (iii) $s_i \in L$ iff $s_{i-1} \in L$.

The base case is immediate. The induction is done as in the proof of Theorem 2, first U_i is *reduced* to W_i, which size is bounded by d', and then W_i^l is inserted to s_{i-1} using Lemma 4. We can now use q-periodicity of L to insert q extra copies of W_i (and therefore U_i) as required. All this is done without affecting membership in L.

Let $s' = s_n$. By construction we have $\forall \tau \in \mathcal{T}_{k+1}, |s'|_\tau = |s|_\tau + q.|U|_\tau = |s|_\tau + |V|_\tau = |t''|_\tau$ and $s' \in L$ iff $s \in L$. Theorem 7 now follows from Corollary 1.

5.2 Unranked trees and modulo counting

In the unranked case, the main difference is that $\text{FO}_{mod(P)}$ is no longer included in MSO but in CMSO. From Courcelle [Cou90] we know that the following family of automata has exactly the same expressive power as CMSO. The automaton is defined as for unranked trees, but each transition, in addition to counting the number of states, up to some threshold m, reached for its children nodes, also counts their occurrences modulo some constant q. We call them *modulo q counting automata*. A language that is defined by a modulo q counting automaton for some q, or equivalently definable in CMSO, is called *extended-regular*.

As in the unranked case, the difficulty is that the number of isomorphism types of trees of depth k is no longer finite, so we need to reason via approximation. We extend the notion of $\sim_n^k$ to $\sim_n^{k,q}$ in the obvious way by requiring that we count the $(n, k-1)$-types of the children of a node modulo q when above threshold n. We refer to those as (q, n, k)-types. The notion of similarity and guards are then extended as expected.

Again, the heart of the proof is the following intermediate result:

Theorem 8. *Let L be a q-periodic extended-regular tree language.*

Then L is definable in FO iff there exists n, k such that L is closed under (q, n, k)-guarded swaps.

Proof. (sketch) The proof is a combination of the ideas in the proof of Theorem 6 and Theorem 3. We only give an overview here. That the conditions are necessary is proved as in Theorem 6. We now fix q, n, k and an extended-regular tree language L which is q-periodic (with l for the number from the periodicity condition) and closed under (q, n, k)-swaps. We fix an automaton A recognizing L and m very big relative to $|A|$.

Lemma 12. For each integers d, there exists K_d such that $s \equiv_q^{K_d} t$ implies $s =_d^{q,n,k+1} t$ and s, t $(n, q, k+1)$-similar.

With the obvious intended meaning for the notation $s =_d^{q,n,k+1} t$, and using the same tools as in Lemma 6 we have:

Lemma 13. For integers d' there exists an integer d such that if $s =_d^{q,n,k+1} t$ then there exists t' such that $s \leq_{d'}^{q,n,k+1} t'$, and t, t' $(n, q, k+1)$-similar, and $t' \in L$ iff $t \in L$.

Pseudo-inclusion is defined as in the unranked case for first-order logic. We have the obvious extension of the pseudo-inclusion lemma:

Lemma 14. For all d', n' then exists n such that if $s \leq_{d'}^{q,n,k+1} t$ and s, t $(n, q, k+1)$-similar then there exists t' such that s is $(q, n', k+1)$-pseudo-included in t', $t' =_{d'}^{q,n',k+1} t$, and $t' \in L$ iff $t \in L$.

Proof. As in the proof of Lemma 7. When a node x of $(q, n, k+1)$-type τ needs to be expanded, we do so by adding children in groups of size q in order not to affect the cardinality modulo q. n is big enough so that we can always do this. □

And this is also the case with inclusion.

Lemma 15. Let U be a context. If $U <_{q,n,k+1} t$ and for all node $x \in U$, its k-spill is $(q, n, k+1)$-thin, then there exists t' such that U is (q, n, k)-included in t', $t' =_\infty^{q,n,k+1} t$ and, $t' \in L$ iff $t \in L$.

Proof of Theorem 9 for unranked trees (sketch). As usual we set all the numbers to be big enough in order to be able to apply all the previous lemmas. Starting with two trees s and t such that $s \equiv_q^K t$, we end up with two trees s and t'' such that s is $(q, n, k+1)$-pseudo-included in t'', $t'' \in L$ iff $t \in l$, and $t'' =_{d'}^{q,n,k+1} s$. We wish to show that $s \in L$ iff $t'' \in L$.

By construction t'' is $h(s)$ plus *loops* inserted between nodes of $h(s)$ and extra *branches* branching off the h-pseudo-tree. As before, each $(q, n, k+1)$-type which occurs outside of $h(s)$ must occur strictly more within t'' than in s and therefore must appear at least d' times in s (and in t''). Therefore for each $(q, n, k+1)$-type τ the total number of occurrences of τ outside of $h(s)$ is zero modulo q. Let Vl be the forest of loops occurring in t'' outside of $h(s)$ and Vb the forest of branches occurring in t'' outside of $h(s)$. As before we first reduce the size of Vl and Vb without affecting either membership in L or the cardinality modulo q of each type, and we can further assume that all types are $(q, n, k+1)$-thin. Let $Vl_1 \cdots Vl_\alpha$ be the set of loops in $t'' \backslash h(s)$ and $Vb_1 \cdots Vb_\beta$ be the set of extra branches in $t'' \backslash h(s)$. From the remark above we have $\forall \tau \in \mathcal{T}_{q,n,k+1}$, $|V|_\tau = 0$ modulo q.

The main difference from the ranked case lies in the following lemma, replacing Lemma 11. Its proof can be obtained by induction on V using the same ideas as in Lemma 11, and is left to the reader.

Lemma 16. There exists α' abstract loops $Ul_1 \cdots Ul_{\alpha'}$ and β' abstract branches $Ub_1 \cdots Ub_{\beta'}$ such that if Ul is the multiset of $(q, n, k+1)$-types occurring in all of the Ul_i and Ub is the same for the Ub_i, then we have $\forall \tau \in \mathcal{T}_{q,n,k+1}$, $|Ul|_\tau = q.|Vl|_\tau$ and $|Ub|_\tau = q|Vb|_\tau$.

We can now transform s by induction in order to insert q copies of Ul_i for each i, and likewise transform t to remove q copies of each Ub_j. This is done as in the proof of Theorem 6 for loops, and as in the unranked case of Theorem 2 for branches, working with groups of size q. The details are left to the reader. □

From Theorem 8 we can now show:

Theorem 9. *Let L be an extended-regular tree language.*

Then L is definable in FO iff it is q-periodic and there exists k such that L is closed under k-guarded swaps.

Proof. Clearly, it is enough to show that closure under k-guarded swaps implies closure under (q, n, k)-guarded swaps for sufficiently large n. This is proved as in the unranked case without modulo quantifiers (Theorem 5). □

6 Decidability

We first show that the characterizations of Theorem 1 and Theorem 5 are generalizations of the string case. Then we show that they lead to decision procedures for membership in FO and FO$_{mod}$.

6.1 The string case

We view a string as a tree whose nodes have a unique child which is its string successor. With this kind of tree, only the vertical swap can be applied. Theorems 1 and 5 imply that a regular language is definable in FO iff it is aperiodic and closed under k-guarded swaps for some k.

In the string case a similar characterization of FO, saying that a regular language is definable in FO iff it is aperiodic and closed under *idempotent-guarded swaps*. This result was proved by Beauquier and Pin [BP89]. We will show how this can be derived from our characterization.

We define the notion of *idempotent-guarded swaps*. Fix a regular (string) languages L and a deterministic automaton A recognizing L. Recall that for any string w, A induces a transition function f^A_w on w such that $f^A_w(q) = q'$ if q' is the state reached by A at the end of w when assuming q at its beginning. Two strings w and w' are *equivalent* if $f^A_w = f^A_{w'}$.

Given a node x, and $k \in \mathbb{N}$, its k-spill is *idempotent* if A induces an idempotent transition function on it (recall that a function f is idempotent if $f \circ f = f$). Two nodes x and x' are *idempotent-similar* if there exists $k, k' \in \mathbb{N}$ such that the k-spill of x is an idempotent equivalent to the k'-spill of x'. The notion of idempotent-guarded swaps is defined as for guarded swaps by replacing the notion depth-k similar with idempotent-similar.

We show that the two notions of guarded swaps are equivalent over strings.

Theorem 10. *A regular language L over strings is closed under idempotent-guarded swaps iff it is closed under k-guarded swaps for some $k \in \mathbb{N}$.*

Proof. One direction is simple: if L is closed under k-guarded swaps then it is closed under idempotent-guarded swaps. Consider a string w and nodes x, y, x', y' such that $x < y < x' < y'$ (we leave the case of equalities to the reader) and such that x, x' are idempotent-similar and y, y' are idempotent-similar. Thus w is of the form $usvs'u'$, where s is $[x, y)$ and s' is $[x', y')$. By hypothesis $svs'u'$ and $s'u'$ starts with a prefix which is equivalent to some idempotent e. By hypothesis we also know that $vs'u'$ and u' starts with a prefix which is equivalent to some idempotent f. Because e, f are idempotent they can

be duplicated as often as necessary without affecting membership into L. Therefore, for all $\alpha \in \mathbb{N}$, $w \in L$ iff $ue^\alpha sf^\alpha ve^\alpha us'f^\alpha u' \in L$. By choosing $\alpha = k$ we can now apply the k-guarded vertical swap and have $ue^\alpha sf^\alpha ve^\alpha s'f^\alpha u' \in L$ iff $ue^\alpha s'f^\alpha ve^\alpha sf^\alpha u' \in L$. We can now conclude because e and f are idempotent: $ue^\alpha s'f^\alpha ve^\alpha sf^\alpha u' \in L$ iff $us'vsu' \in L$.

The other direction is proven as follows. Assume L is regular, let A be a deterministic automaton for L, and assume L is closed under idempotent-guarded swaps. Let α be the number of states of A and take $k = \alpha^\alpha + 1$. We show that L is closed under k-guarded swaps. First, consider a string w and a node $x \in w$. For $i \leq k$ let v_i be the i-spill of x. By the choice of k there must be $i < j$ such that $f^A_{v_i} = f^A_{v_j}$. Let u be the string such that $v_j = v_i u$. We have $w = w_1 v_i u w_2$. Because $f^A_{v_i} = f^A_{v_j}$ we have forall $\beta \in \mathbb{N}^+$, $w_1 v_i u^\beta w_2 \in L$ iff $w = w_1 v_i u w_2 \in L$. Notice now that for all strings u there exists β such that u^β is idempotent. Hence we can insert an idempotent e into k-spill in a canonical way without affecting membership in L.

Now consider a string w and positions x, y, x', y' such that $x < y < x' < y$ and x, x' are depth-k similar, and y, y' are depth-k similar. For convenience we will assume that y is not in the k-spill of x, x' is not in the k-spill of y, and y is not in the k-spill of x': the other cases are proven similarly. Thus w can be decomposed into $w_1 \cdot sv \cdot s'w_2 \cdot sv' \cdot s'w_3$ where s and s' are the k-spill of x and y and the "dots" mark the position of x, y, x', y'.

We can now apply the technique mentioned in the paragraph above and insert an idempotent e into s and an idempotent f into s' without affecting membership in L. Thus we have

$$w \in L \text{ iff } w_1 \cdot s_1 e s_2 v \cdot s'_1 f s'_2 w_2 \cdot s_1 e s_2 v' \cdot s'_1 f s'_2 w_3 \in L.$$

We now use the closure of L under idempotent-guarded swaps and obtain (the "dots" now indicate the sections that are swapped):

$w_1 s_1 \cdot e s_2 v s'_1 f \cdot s'_2 w_2 s_1 \cdot e s_2 v' s'_1 f \cdot s'_2 w_3 \in L$ iff $w_1 s_1 \cdot e s_2 v' s'_1 f \cdot s'_2 w_2 s_1 \cdot e s_2 v s'_1 f \cdot s'_2 w_3 \in L$.

But the latter is precisely $w_1 \cdot sv' \cdot s'w_2 \cdot sv \cdot s'w_3$ as required for k-guarded swapping. □

Note that Theorem 10 does not generalize to trees. In the tree case it is still true that any k-spill, for a sufficiently large k, will contain an idempotent, but which idempotent this is and where it can be inserted can no longer be computed by looking only at the k-spill.

6.2 Decision Procedure

Let L be a regular tree language, A be a minimal deterministic bottom-up automaton (ranked or unranked) recognizing L, and let Q be the set of states of A. In this section we will consider deciding whether L is in FO. The input of the problem is A, and thus the complexity is relative to the number of states of A.

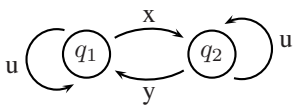
In the string case deciding whether a regular language L can be defined in FO is PTIME in the size of such an A [Pin05]. Note that this is not immediate, as checking aperiodicity alone is PSPACE-complete [CH91]. It turns out that ideas similar to [Pin05] show that membership in FO can actually be checked in PTIME also in the tree setting. We will show this only for the ranked case; the unranked case is proved along the same lines.

We will first show that the aperiodicity condition in Theorem 1 can be replaced by one that is easier to check. Following the approach of [Pin96]: we replace aperiodicity by the condition $(\dagger)l$: there exists l such that for any $s, x, y \in \mathcal{T}^1$, $e \in \mathcal{T}^1$ idempotents, and $s' \in \mathcal{T}$ $s \cdot (e \cdot x \cdot e \cdot y \cdot e)^l \cdot s' \in L$ iff $s \cdot (e \cdot x \cdot e \cdot y \cdot e)^l \cdot e \cdot x \cdot e \cdot s' \in L$.

In [Pin05] it is shown that for a string language that is closed under idempotent-guarded swaps, aperiodicity is equivalent to $(\dagger)l$. More precisely, [Pin05] shows that in any monoid satisfying $e \cdot u \cdot f \cdot s' \cdot e \cdot v \cdot f = e \cdot v \cdot f \cdot s' \cdot e \cdot u \cdot f$ for idempotents e, f and arbitrary monoid elements u, s', v, we

have the following are equivalent: a) there is l such that the identity $u^l = u^{l+1}$ holds and b) there is l such that $(e \cdot x \cdot e \cdot y \cdot e)^l = (e \cdot x \cdot e \cdot y \cdot e)^l \cdot e \cdot x \cdot e$ holds (where again e, f range over idempotents and x, y, u, v, s' range over monoid elements). Since the set of $\simeq$ classes of pointed trees associated with a regular tree language forms a monoid, this result can be applied to pointed trees. We can now observe that the proof of Theorem 10 shows that if a tree language is closed under k-guarded vertical swaps, then it is closed under idempotent-guarded vertical swaps (where a tree context is idempotent iff the corresponding function is an idempotent in the monoid). Combining this with the equivalence of a) and b) above and our characterization theorems, we have that: a regular tree language is in FO iff for some k is closed under under k-guarded vertical and horizontal swaps, and for some l it satisfies $(\dagger)l$.

We will now show that one can decide in PTIME whether or not a regular tree language satisfies $(\dagger)l$. This will rely on the notion of graph pattern matching, which we review here. For the purposes of this section, a *pattern* is a graph whose edges are labeled by variables which range over elements of Λ^* for some finite alphabet Λ. In addition a pattern comes with side conditions stating which nodes of the pattern should be interpreted as distinct nodes. Let G be a graph whose edges are labeled in Γ. Such a graph G *matches* a pattern if there is a mapping f taking each variable in the pattern to strings in Λ^* and each node of the pattern to a node of G such that for each side constraint $p_1 \neq p_2$, $f(p_1) \neq f(p_2)$, and such that whenever there is an edge from p_1 to p_2 in the pattern labeled with v, there is a path from $f(p_1)$ to $f(p_2)$ in G whose labels yield the string $f(v)$. In [CPP93] it is noted that for every fixed pattern, the problem of determining given a graph, whether the graph matches the pattern, is in PTIME. This result was used to show that FO-definability is in PTIME in the string case. From a minimal automaton A recognizing L one constructs an edge-labeled graph $G_A = (V_A, E_A)$ as follows. The vertex set V_A of G_A is the set of states of A. The transitions $E_A \subseteq V_A \times \Sigma \times V_A$ are labeled with letters of the alphabet Σ of L and correspond to the transitions of A.



Let P be the pattern depicted above together with the condition $q_1 \neq q_2$. It has been shown that [CPP93] L verifies $(\dagger)l$ for some l iff G_A does not match P. Minimality of A is used in the left to right direction.

This result extends to trees as follows. From an automaton A define $G_A = (V_A, E_A)$ as follows. The set of vertices V_A is Q. The set of edges E_A is included in $V_A \times \Lambda \times V_A$ where $\Lambda = \Sigma \times \bigcup_{i<r} Q^i \times \{1 \dots r\}$ where r is the rank. We connect a node p to a node p' via an edge (a, λ, j), where λ is a sequence of (at most) r states, if $\delta(\lambda, a) = p'$ where the j^{th} state of λ is p. represents the inverse of a transition of the automaton. The same proofs as in [Pin05, CPP93] show that:

Lemma 17. L verifies $(\dagger)$l for some l iff G_A does not match P.

Therefore we have:

Proposition 2. *One can decide in* PTIME *(in the size of a minimal deterministic bottom-up automaton) whether or not a regular language L satisfies $(\dagger)l$ for some l.*

We now show that it is decidable in PTIME whether there exists k such that L is closed under k-guarded swaps. We first show that if L is closed under k-guarded swaps then it is closed under k'-guarded swaps where k' is computable from A.

Lemma 18. Let L be a regular language. Let α be the number of states of an automaton for L and $k' = \alpha^2 + 1$. Then L is closed under k-guarded swaps for some k iff it is closed under k'-guarded swaps.

Proof. We show that if L is closed under k-guarded horizontal swaps for any k then it is closed under k'-guarded horizontal swaps. The proof for vertical swaps is similar and is left to the reader.

Let t be a tree and x, y two nodes of t that are depth-k' similar. We first show that we can transform t by pumping in new subtrees so that x, y become k-similar in the new tree, without affecting membership in L. Let $t_1 = t|_x$ and $t_2 = t|_y$. Let $f(t, x, y)$ be the lex-minimum of integers (n, m) such that there is a leaf node z in the common prefix of t_1 and t_2 such its level is n and its position is m among nodes of level m: we know that n is no smaller than k'. We will transform t by pumping so that $f(t, x, y)$ increases.

Let P be the common prefix of t_1 and t_2, w be a leaf node in P witnessing $f(t, x, y)$. Consider a run r of A on t. r assigns a state q to each node of P when running up both t_1 and t_2. We assign to each node of P the pair of states (q_1, q_2) such that q is the state of r at the corresponding node in t_1 while q' is the state of r at the corresponding node in t_2. By the choice of k', on the path from the root to w, there must be a pair of states that repeats. Let z and z' be two such nodes. Consider the context C_1 between the copy of z and z' within t_1, including z and replacing z' by a port. Without affecting membership into L, we can duplicate C_1 in t_1 as many times as we wish. Considering the context C_2, between the copy of z and z' within t_2 we can perform the same duplication within t_2. Performing both these duplications, we have now increased the depth of w to be above k without removing any nodes from P, thus increasing $f(t, x, y)$.

Performing this repeatedly, we arrive at a tree t' obtained from t by adding new sections such that $f(t', x, y) = (k, l)$ for a given k. We can now apply k-guarded swapping to switch the subtrees under x and y in t'. We can now remove the extra sections, resulting in a tree formed from the original tree t by swapping the subtrees under x and y, as required.

□

From the above it is already clear that one can decide whether L is closed under k-guarded swaps for some k, since one needs to check only that L is closed under k-guarded swaps for k above, and for a fixed k checking closure under swaps is easily seen to be decidable. Below we will show a stronger result:

Theorem 11. *There is a* PTIME *algorithm that decides, given a minimal automaton for a regular language L and an integer k, whether or not L is closed under k-guarded swaps*

We will prove Theorem 11 in several steps. We first show that deciding closure under horizontal swapping is in PTIME.

Claim 4. *There is a* PTIME *algorithm that, given a deterministic bottom-up automaton A on ranked trees, positive integer k, and a subset of the alphabet symbols Σ_0, decides whether or not there is a tree T accepted by A such that only symbols in Σ_0 occur in the first k levels of T.*

Proof. (of claim)

We calculate (and store) by induction on k a table $T_A(q, k)$ of booleans such that $T_A(q, k)$ is true iff there is a tree T such that A reaches q at the root of T and T has only letters in Σ_0 on the first k levels. In the induction step to calculate the value of $T_A(q, k)$ we check for each transition $\delta(q_1 \dots q_i, a) = q$ in A with $a \in \Sigma_0$ and each $q_i : i \leq r$ in Q, whether or not we have $T_A(q_i, k-1) = true$ for every q_i. □

From this we see:

Claim 5. *There is a* PTIME *algorithm that, given a deterministic bottom-up automaton A, positive integer k, and states q_1, q_2 of A, determines whether or not there exist k-similar trees T_1, T_2 such that when A is run on $T_i : i = 1, 2$ it leads to state q_i at the root of T_i. We write $q_1 \sim_k q_2$ when this occurs.*

Proof. Let $\Sigma_\perp = (\Sigma \cup \perp)^2$, where $\perp$ is a symbol not in Σ. Given two ordered Σ-trees T_1, T_2, the $(\Sigma_\perp)^2$-tree $zip(T_1, T_2)$ is defined as follows: consider trees T_1, T_2 as functions from a prefix-closed finite subset of $\bigcup_i N^i$ into Σ. The tree $zip(T_1, T_2)$ contains one node for every node in $dom(T_1) \cup dom(T_2)$, with the label of node x being $(T_1^\perp(x), T_2^\perp(x))$, where $T_i^\perp(x)$ is the label of x in T_i if $x \in dom(T_i)$ and $\perp$ otherwise. We consider the synchronous product A^2 of A with itself. In the ranked case, this is the automaton over alphabet $(\Sigma_\perp)^2$ that accepts a tree T iff $T = zip(T_1, T_2)$ for some $T_1, T_2 \in L$. It is easy to show that A^2 can be taken to have state space Q^2, and can be constructed in PTIME from A.

We let Σ_0 be the set of pairs $(a, a) : a \in \Sigma$, and then apply Claim 4 to A^2 and any state (q_1, q_2). □

From Claim 5, we can derive our first result:

Lemma 19. Verifying that the language of trees given by a minimal deterministic automaton A satisfies closure under k-guarded horizontal swapping is in PTIME in $|A|$ and k.

Proof. We prove this in the case of binary trees. The extension to arbitrary ranked trees and to unranked trees is left to the reader.

Recall that a 2-pointed tree is a tree with two two distinguished leaves called ports. An automaton A and a 2-pointed tree Δ induce a function Δ_A from $Q \times Q$ to Q simulating a bottom-up evaluation.

By minimality and determinism of A, we need to check that for all states q_1, q_2, if $q_1 \sim_k q_2$ then for all 2-pointed trees Δ we have $\Delta_A(q_1, q_2) = \Delta_A(q_2, q_1)$.

From Claim 5 the lemma follows assuming that we have shown: given two states q_1 and q_2 such that $q_1 \sim_k q_2$, we can check in PTIME whether for each 2-pointed tree Δ whether $\Delta_A(q_1, q_2) = \Delta_A(q_2, q_1)$.

This is an easy consequence of the fact that reachability for tree automata can be computed in PTIME. Indeed consider the automata B which is the synchronous product of A with itself. The set $S(q_1, q_2)$ of states of B reachable from (q_1, q_2) is computable in PTIME. The same holds for $S(q_2, q_1)$. It is now sufficient to check that for all labels $a \in \Sigma$ and all pairs $(p, q) \in S(q_1, q_2)$ and $(p', q') \in S(q_2, q_1)$, $\delta_A(p, q', a) = \delta_A(q, p', a)$. This is easy to do in PTIME.

□

We now turn to deciding closure under vertical swaps, which is checked using similar ideas.

We need the following two relation among states of A. We say that $R_k(r, p_1, q_1, p_2, q_2)$ holds whenever there exists a context Δ and a tree T such that C and T agree up to depth k, $\Delta_A(p_1) = q_1$, $\Delta_A(p_2) = q_2$ and r is the state reached by A on T. We say that $S_k(p_1, q_1, p_2, q_2, p_1', p_2', q_1', q_2')$ holds whenever there exist two contexts Δ and Δ' such that Δ and Δ' agree up to depth k, $\Delta_A(p_i) = q_i$ and $\Delta'_A(p_i') = q_i'$.

Claim 6. *For each $k \in \mathbb{N}$, both R_k and S_k are computable in* PTIME.

Proof. This is done as in Claim 4 and Claim 5, with the extra reachability constraints being verifiable in PTIME. □

We are now ready to show:

Lemma 20. Verifying that the language of r-ranked trees given by automaton A satisfies closure under k-guarded vertical swapping is in PTIME in $k, |A|$.

Proof. Immediate from Claim 6. k-guarded vertical swapping fails if we can find $(r, p_1, q_1, p_2, q_2) \in R_k$ and $(r, p_1, q_2, s', q_1, s, r, p_2) \in S_k$ with $s \neq s'$. This can be checked in PTIME once S_k and R_k are computed.

□

From Lemma 19, Lemma 20, we have Theorem 11.

From Lemma 18, Theorem 11, and Proposition 2, we conclude the following:

Theorem 12. *There is a* PTIME *algorithm that takes a minimal deterministic automaton for a regular language of ranked trees and decides whether or not it is definable in FO.*

The theorem also applies to unranked trees, via easy extensions of Lemma 18, Theorem 11, and Proposition 2.

Extensions to FO$_{mod}$ To extend the above complexity bound to FO$_{mod}$, we need only modify the above argument to accommodate q-periodicity rather than aperiodicity. We can use that for a monoid satisfying $e \cdot u \cdot f \cdot s' \cdot e \cdot v \cdot f = e \cdot v \cdot f \cdot s' \cdot e \cdot u \cdot f$ we have $u^l = u^{l+q}$ holds for some l iff $(e \cdot x \cdot e \cdot y \cdot e)^l = (e \cdot x \cdot e \cdot y \cdot e)^l \cdot (e \cdot x \cdot e)^q$ holds for some l (where, again, e, f range over idempotents and x, y, u, v, s' range over monoid elements). The proof of this is a simple modification of the proof for aperiodicity. One can decide whether this latter equality holds using a pattern P^q obtained from P by adding $q-1$ additional nodes, which with q_2 form a chain of length q, with edges from one element of the chain to the next labeled with x. Each element of the chain has a self-loop labeled with u associated with it, and the last element in the chain is constrained to be distinct from q_1. In the ranked case, the graphs which are matched against pattern P^q are the same as those matched against P in the ranked FO case. In the unranked case, we use the same patterns, but now we must match graphs whose labels correspond to the input of the transition function for a modulo counting automaton. This argument shows:

Theorem 13. *There is a* PTIME *algorithm that, given a minimal deterministic bottom-up ranked tree automaton, checks whether the corresponding tree language is definable in* FO$_{mod}$*. There is a* PTIME *algorithm that, given a minimal deterministic modulo counting automaton on unranked trees, checks whether the corresponding language is definable in* FO$_{mod}$*.*

7 Conclusions

The main result presented here is the decidability of FO-definability in trees. We believe that our characterization (and the decidability results that follow) extends to ω-trees. In addition to giving a decision procedure, the characterization here has been useful for demonstrating that certain queries are first-order; for example, it is has been used to prove that order-invariant first-order queries over trees are first-order expressible.

The class LT of languages is defined as for LTT but without the threshold. That is, one can check the occurrence or absence of a pattern in a string but can no longer count the number of occurrences. We are considering how to modify our axioms to characterize LT.

Acknowledgment: We wish to thank Jean-Eric Pin for many fruitful discussions on the word case and Mikołaj Bojańzyk for his help on an earlier draft of this paper.

References

[BP89] D. Beauquier and J-E. Pin. Factors of words. In *Automata, Languages and Programming*, pages 63–79, 1989.

[BS05] M. Benedikt and L. Segoufin *Regular Languages definable in FO*. In *STACS 2005*.

[BW04] M. Bojanczyk and I. Walukiewicz Characterizing EF and EX tree logics. To appear, in *Concur*, 2004.

[Bc60] J. Büchi. Weak second-order logic and finite automata. *S. Math. Logik Grunlagen Math.*, 6:66–92, 1960.

[CH91] S. Cho and D T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88(1):99–116, 1991.

[Cou90] B. Courcelle. *The Monadic Second Order Logic of Graphs I: Recognizable Sets of Finite Graphs Information and Computation*, 85:12–75, 1990.

[CPP93] J. Cohen, D. Perrin, and J-E. Pin. On the expressive power of temporal logic for finite words. *Journal of Computer and Science Systems*, 46(1993):271–294, 1993.

[Don70] J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[EW03] Z. Ésik and P. Weil. On logically defined recognizable tree languages. In *FSTTCS*, pages 195-207, 2003.

[Heu91] U. Heuter. First-order properties of trees, star-free expressions, and aperiodicity. *Informatique Thorique et Applications*, 25:125–146, 1991.

[Lib04] L. Libkin. *Elements of finite model theory*. Springer, 2004.

[MP71] R. McNaughton and S. Papert. *Counter-free Automata*. MIT Press, 1971.

[Pin96] J-E. Pin. The expressive power of existential first order sentences of Bchi's sequential calculus. In *Proc. of Intl. Coll. on Automata, Languages and Programming*, pages 300–311, 1996.

[Pin05] J-E. Pin. The expressive power of existential first order sentences of Büchi's sequential calculus. *Discrete Mathematics*, 291, pages 155–174, 2005.

[Pot95] A. Potthoff. First-order logic on finite trees. In *Theory and Practice of Software Development (TAPSOFT)*, pages 125–139, 1995.

[PT93] A. Potthoff and W. Thomas. Regular tree languages without unary symbols are star-free. In *Fundamentals of Computation Theory (FCT)*, pages 396–405, 1993.

[Sch65] M. P. Schtzenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhuser, 1994.

[Tho97] W. Thomas. *Handbook of Formal Languages*, volume 3, chapter 7. Springer, 1997.

[TW68] J.W. Thatcher and J.B. Wright. Generalized finite automata with an application to a decision problem of second order logic. *Math. Syst. Theory*, 2:57–82, 1968.

[TW85] D. Thérien and A. Weiss. Graph congruences and wreath products *J. Pure and Applied Algebra*, 36:205–215, 1985.

[Wil96] T. Wilke. An algebraic characterization of frontier testable tree languages. *Theoretical Computer Science*, 154(1):85–106, 1996.