# Tree languages defined in first-order logic with one quantifier alternation[*]

Mikołaj Bojańczyk[1] and Luc Segoufin[2]

[1] Warsaw University
[2] INRIA - LSV

**Abstract.** We study tree languages that can be defined in $\Delta_2$. These are tree languages definable by a first-order formula whose quantifier prefix is $\exists^*\forall^*$, and simultaneously by a first-order formula whose quantifier prefix is $\forall^*\exists^*$, both formulas over the signature with the descendant relation. We provide an effective characterization of tree languages definable in $\Delta_2$. This characterization is in terms of algebraic equations. Over words, the class of word languages definable in $\Delta_2$ forms a robust class, which was given an effective algebraic characterization by Pin and Weil [11].

## 1 Introduction

We say a logic has a decidable characterization if the following decision problem is decidable: "given as input a finite automaton, decide if the recognized language can be defined using a formula of the logic". Representing the input language by a finite automaton is a reasonable choice, since many known logics (over words or trees) are captured by finite automata.

This type of problem has been successfully studied for word languages. Arguably best known is the result of McNaughton, Papert and Schützenberger [12, 9], which says that the following three conditions on a regular word language $L$ are equivalent: a) $L$ can be defined in first-order logic with order and label tests; b) the syntactic semigroup of $L$ does not contain a non-trivial group. Since condition b) can be effectively tested, the above theorem gives a decidable characterization of first-order logic. This result demonstrates the importance of this type of work: a decidable characterization not only gives a better understanding of the logic in question, but it often reveals unexpected connections with algebraic concepts. During several decades of research, decidable characterizations have been found for fragments of first-order logic with restricted quantification and a large group of temporal logics, see [10] and [16] for references.

An important part of this research has been devoted to the quantifier alternation hierarchy, where each level counts the alterations between $\forall$ and $\exists$ quantifiers in a first-order formula in prenex normal form. Formulas that have $n-1$ alternations are called $\Sigma_n$ if they begin with $\exists$, and $\Pi_n$ if they begin with $\forall$. For instance, the word property "some position has label $a$" can be defined by a $\Sigma_1$ formula $\exists x.\, a(x)$, while the language $a^*ba^*$ can be defined by the $\Sigma_2$ formula $\exists x\forall y.\, b(x) \wedge\ (y \neq x \Rightarrow a(y))$.

A lot of attention has been devoted to analyzing the low levels of the quantifier alternation hierarchy. The two lowest levels are easy: a word language is

definable in $\Sigma_1$ (resp. $\Pi_1$) if and only if it is closed under inserting (removing) letters. Both latter properties can be tested in polynomial time based on a recognizing automaton, or semigroup. However, just above $\Sigma_1, \Pi_1$, and even before we get to $\Sigma_2, \Pi_2$, we already find two important classes of languages. A fundamental result, due to Simon [14], says that a language is defined by a boolean combination of $\Sigma_1$ formulas if and only if its syntactic monoid is $\mathcal{J}$-trivial. Above the boolean combination of $\Sigma_1$, we find $\Delta_2$, i.e. languages that can be defined simultaneously in $\Sigma_2$ and $\Pi_2$. As we will describe later on, this class turns out to be surprisingly robust and it is the focus of this paper. Another fundamental result, due to Pin and Weil [11], says that a regular language is in $\Delta_2$ if and only if its syntactic monoid is in DA. The limit of our knowledge is level $\Sigma_2$: it is decidable if a language can be defined on level $\Sigma_2$ [1, 11], but there are no known decidable characterization for boolean combinations of $\Sigma_2$, for $\Delta_3$, for $\Sigma_3$, and upwards.

For trees even less is known. No decidable characterization has been found for what is arguably the most important proper subclass of regular tree languages, first-order logic with the descendant relation, despite several attempts. Similarly open are chain logic and the temporal logics CTL, CTL* and PDL. However, there has been some recent progress. In [5], decidable characterizations were presented for some temporal logics, while Benedikt and Segoufin [2] characterized tree languages definable in first-order logic with the successor relation (but without the descendant relation).

This paper is part of a program to understand the expressive power of first-order logic on trees, and the quantifier alternation hierarchy in particular. The idea is to try to understand the low levels of the quantifier alternation hierarchy before taking on full first-order logic (which is contrary to the order in which word languages were analyzed). We focus on a signature that contains the ancestor order on nodes and label tests. In particular, there is no order between siblings. As shown in [3], there is a reasonable notion of concatenation hierarchy for tree languages that corresponds to the quantifier alternation hierarchy. Levels $\Sigma_1$ and $\Pi_1$ are as simple for trees as they are for words. A recent unpublished result [8] extends Simon's theorem to trees, by giving a decidable characterization of tree languages definable by a Boolean combination of $\Sigma_1$ formulas. There is no known characterization of tree languages definable in $\Sigma_n$ for $n \geq 2$.

The contribution of this paper is a decidable characterization of tree languages definable in $\Delta_2$, i.e. definable both in $\Sigma_2$ and $\Pi_2$. As we signaled above, for word languages the class $\Delta_2$ is a well studied and important, with numerous equivalent characterizations. Among them one can find [11, 15, 13, 7]: a) word languages that can be defined in the temporal logic with operators $\mathsf{F}$ and $\mathsf{F}^{-1}$; b) word languages that can be defined by a first-order formula with two variables, but with unlimited quantifier alternations; c) word languages whose syntactic semigroup belongs to the semigroup variety DA; d) word languages recognized by two-way ordered deterministic automata; e) a certain form of "unambiguous" regular expressions.

It is not clear how to extend some of these concepts to trees. Even when natural tree counterparts exist, they are not equivalent. For instance, the temporal logic in a) can be defined for trees—by using operators "in some descendant" and "in some ancestor". This temporal logic was studied in [4], however it was shown to have an expressive power incomparable with that of $\Delta_2$. A characterization of $\Delta_2$ was left as an open problem, one which is solved here.

We provide an algebraic characterization of tree languages definable in $\Delta_2$. This characterization is effectively verifiable if the language is given by a tree automaton. It is easy to see that the word setting can be treated as a special case of the tree setting. Hence our characterization builds on the one over words. However the added complexity of the tree setting makes both formulating the correct condition and generalizing the proof quite nontrivial.

## 2    Notation

*Trees, forests and contexts*  In this paper we work with finite unranked ordered trees and forests over a finite alphabet $A$. Formally, these are expressions defined inductively as follows: If $s$ is a forest and $a \in A$, then $as$ is a tree. If $t_1, \ldots, t_n$ is a finite sequence of trees, then $t_1 + \cdots + t_n$ is a forest. This applies as well to the empty sequence of trees, which is called the *empty forest,* and denoted 0 (and which provides a place for the induction to start). Forests and trees alike will be denoted by the letters $s, t, u, \ldots$ When necessary, we will remark on which forests are trees, i.e. contain only one tree in the sequence.

A set $L$ of forests over $A$ is called a *forest language.*

The notions of node, descendant and ancestor relations between nodes are defined in the usual way. We write $x < y$ to say that $x$ is an ancestor or $y$ or, equivalently, that $y$ is a descendant of $x$.

If we take a forest and replace one of the leaves by a special symbol $\square$, we obtain a *context.* Contexts will be denoted using letters $p, q, r$. A forest $s$ can be substituted in place of the hole of a context $p$, the resulting forest is denoted by $ps$. There is a natural composition operation on contexts: the context $qp$ is formed by replacing the hole of $q$ with $p$. This operation is associative, and satisfies $(pq)s = p(qs)$ for all forests $s$ and contexts $p$ and $q$.

When $a$ is a letter, we will sometimes also write $a$ for the context that has one root with label $a$ and a hole below. For instance, any tree with label $a$ in the root can be written as $at$, for some forest $t$.

We say a forest $s$ is an *immediate piece* of a forest $s'$ is $s, t$ can be decomposed as $s = pt$ and $s' = pat$ for some contexts $p, q$ and some label $a$. The reflexive transitive closure of the immediate piece relation is called the *piece* relation. We write $s \preceq t$ to say that $s$ is a piece of $t$. In other words, a piece of $t$ is obtained by removing nodes from $t$. We extend the notion of piece to contexts. In this case, the hole must be preserved while removing the nodes. The notions of piece for forests and contexts are related, of course. For instance, if $p, q$ are contexts with $p \preceq q$, then $p0 \preceq q0$. Also, conversely, if $s \preceq t$, then there are contexts $p \preceq q$ with $s = p0$ and $t = q0$. The figure below depicts two contexts, the left one being a piece of the right one, as can be seen by removing the white nodes.
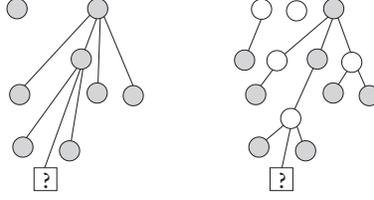
We will be considering three types of languages in the paper: *forest languages* i.e. sets of forests, denoted $L$; *context languages*, i.e. sets of contexts, denoted $K$, and *tree languages*, i.e. sets of trees, denoted $M$.

*Logic* The focus of this paper is the expressive power of first-order logic on trees. A forest can be seen as a logical relational structure. The domain of the structure is the set of nodes. The signature contains a unary predicate $P_a$ for each symbol $a$ of $A$ plus the binary predicate $<$ for the ancestor relation. A formula without free variables over this signature defines a set of forests, these are the forests where it is true. We are particularly interested in formulas of low quantifier complexity. A $\Sigma_2$ formula is a formula of the form

$$\exists x_1 \cdots x_n \ \forall y_1 \cdots y_m \ \ \gamma \ ,$$

where $\gamma$ is quantifier free. Properties defined in $\Sigma_2$ are closed under disjunction and conjunction, but not necessarily negation. The negation of a $\Sigma_2$ formula is called a $\Pi_2$ formula, equivalently this is a formula whose quantifier prefix is $\forall^* \exists^*$. A property forest is called $\Delta_2$ if it can be expressed both by a $\Sigma_2$ and a $\Pi_2$ formula.

*The problem* We want an algorithm deciding if a given regular forest language is definable in $\Delta_2$.

Notice that the forest property of "being a tree" is definable in $\Delta_2$. The $\Sigma_2$ formula says there exists a node that is an ancestor of all other nodes, while the $\Pi_2$ says that for every two nodes, there exists a common ancestor. Hence a solution of the problem for forest languages also gives a solution for tree languages.

As noted earlier, the corresponding problem for words was solved by Pin and Weil: a word language $L$ is definable in $\Delta_2$ if and only if its syntactic monoid $M(L)$ belongs to the variety DA, i.e. it satisfies the identity

$$(mn)^\omega = (mn)^\omega m (mn)^\omega$$

for all $m, n \in M(L)$. The power $\omega$ means that the identity holds for all sufficiently large powers. Since one can effectively test if a finite monoid satisfies the above property (it is sufficient to verify the power $|M(L)|$), it is decidable if a given regular word language is definable in $\Delta_2$. We assume that the language $L$ is given by its syntactic monoid and syntactic morphism, or by some other representation, such as a finite automaton, from which these can be effectively computed. Later on in the paper, we will reprove the result of Pin and Weil, since we need it—actually, a slightly expanded version—in part of our own proof.

4

We will show that a similar characterization can be found for forests; although the identities will be more involved. For decidability, it is not important how the input language is represented. In this paper, we will represent a forest language by a forest algebra that recognizes it. Forest algebras are described in the next section.

*Basic properties of $\Sigma_2$* Most of the proofs in the paper will work with $\Sigma_2$ formulas. We present some simple properties of such formulas in this section.

Apart from defining forest languages, we will also be using $\Sigma_2$ formulas to define languages of contexts. To define a context language we use $\Sigma_2$ formulas with a free variable; such a formula is said to hold in a context if it is true when the free variable is mapped to the hole of the context.

**Fact 1** Let $K$ be a context language, $L$ a forest language, and $M$ a tree language. If these languages are all definable in $\Sigma_2$, then so are:

1. For any letter $a$, the forest language $KaL$.
2. The forest language $M \oplus L$. This is the set of forests $t_1 + t + t_2$ such $t$ is a tree in $M$, and the concatenation of forests $t_1 + t_2$ is in $L$.

**Proof**
We only do the proof for $KaL$. The formula places an existentially quantified variable $x$ on the node $a$, and then relativizes the formulas for languages $K$ and $L$ to nodes that are, respectively, not descendants of $x$ and descendants of $x$. ∎

## 3 Forest algebras

*Forest algebras* were introduced by Bojańczyk and Walukiewicz as an algebraic formalism for studying regular tree languages [6]. Here we give a brief summary of the definition of these algebras and their important properties. A forest algebra consists of a pair $(H, V)$ of finite monoids, subject to some additional requirements, which we describe below. We write the operation in $V$ multiplicatively and the operation in $H$ additively, although $H$ is not assumed to be commutative. We accordingly denote the identity of $V$ by $\square$ and that of $H$ by 0. We require that $V$ act on the left of $H$. That is, there is a map $(h, v) \mapsto vh \in H$ such that $w(vh) = (wv)h$ for all $h \in H$ and $v, w \in V$. We further require that this action be *monoidal*, that is, $h \cdot \square = h$ for all $h \in H$, and that it be *faithful*, that is, if $vh = wh$ for all $h \in H$, then $v = w$. Finally we require that for every $g \in H$, $V$ contains elements $(\square + g)$ and $(g + \square)$ defined by $(\square + g)h = h + g, (g + \square)h = g + h$ for all $h \in H$.

A morphism $\alpha : (H_1, V_1) \to (H_2, V_2)$ of forest algebras is actually a pair $(\gamma, \delta)$ of monoid morphisms such that $\gamma(vh) = \delta(v)\gamma(h)$ for all $h \in H$, $v \in V$. However, we will abuse notation slightly and denote both component maps by $\alpha$.

Let $A$ be a finite alphabet, and let us denote by $H_A$ the set of forests over $A$, and by $V_A$ the set of contexts over $A$. Clearly $(H_A, V_A)$ with forest substitution as action, forms forest algebra which we denote $A^\Delta$.

We say that a forest algebra $(H, V)$ *recognizes* a forest language $L \subseteq H_A$ if there is a morphism $\alpha : A^\Delta \to (H, V)$ and a subset $X$ of $H$ such that $L =$

$\alpha^{-1}(X)$. It is easy to show that a forest language is regular if and only if it is recognized by a finite forest algebra.

Given any finite monoid $M$, there is a number $\omega(M)$ (denoted by $\omega$ when $M$ is understood from the context) such that for all element $x$ of $M$, $x^\omega$ is an idempotent: $x^\omega = x^\omega x^\omega$. Therefore for any forest algebra $(H, V)$ and any element $u$ of $V$ and $g$ of $H$ we will write $u^\omega$ and $\omega(g)$ for the corresponding idempotents.

Given $L \subseteq H_A$ we define an equivalence relation $\sim_L$ on $H_A$ by setting $s \sim_L s'$ if and only if for every context $x \in V_A$, $hx$ and $h'x$ are either both in $L$ or both outside of $L$. We further define an equivalence relation on $V_A$, also denoted $\sim_L$, by $x \sim_L x'$ if for all $h \in H_A$, $xh \sim_L x'h$. This pair of equivalence relations defines a congruence of forest algebras on $A^\Delta$, and the quotient $(H_L, V_L)$ is called the *syntactic forest algebra* of $L$.

We now extend the notion of piece to elements of a forest algebra $(H, V)$. The general idea is that a context $v \in V$ is a piece of a context $w \in V$ if one can construct a term (using elements of $H$ and $V$) which evaluates to $w$, and then take out some parts of this term to get $v$.

**Definition 2** Let $(H, V)$ be a forest algebra. We say $v \in V$ *is a piece* of $w \in V$, denoted by $v \preceq w$, if $\alpha(p) = v$ and $\alpha(q) = w$ hold for some morphism

$$\alpha : A^\Delta \to (H, V)$$

and some contexts $p \preceq q$ over $A$. The relation $\preceq$ is extended to $H$ by setting $g \preceq h$ if $g = v0$ and $h = w0$ for some contexts $v \preceq w$.

## 4   Characterization of $\Delta_2$

In this section we present the main result of the paper: a characterization of $\Delta_2$ in terms of two identities.

**Theorem 3**
*A forest language is definable in $\Delta_2$ if and only if its syntactic forest algebra satisfies the following identities:*

$$h + g = g + h \tag{1}$$

$$v^\omega w v^\omega = v^\omega \qquad \textit{for } w \preceq v \tag{2}$$

**Corollary 4** It is decidable if a forest language can be defined in $\Delta_2$.

**Proof**
We assume that the language is represented as a forest algebra. This representation can be computed based on other representations, such as automata or monadic second-order logic.

Once the forest algebra is given, both conditions (1) and (2) can be tested in polynomial time by searching through all elements of the algebra. The relation $\preceq$ can be computed in polynomial time, using a fixpoint algorithm as in [4].   ∎

Theorem 3 is stated in terms of forest languages, but as mentioned earlier, the same result works for trees.

We begin with the easier implication in Theorem 3, that the syntactic forest algebra of a language definable in $\Delta_2$ must satisfy the identities (1) and (2). The first identity must clearly be satisfied since the signature only contains the descendant relation. The other identity follows from the following claim, whose standard proof is deferred to Appendix A.

**Lemma 5** Let $\varphi$ be a formula of $\Sigma_2$ and let $p \preceq q$ be two contexts. For $n \in \mathbb{N}$ sufficiently large, forests satisfying $\varphi$ are closed under replacing $p^n p^n$ with $p^n q p^n$.

The rest of Section 4 contains the more difficult implication of Theorem 3. We will show that if a language is recognized by a forest algebra satisfying identities (1) and (2), then it is definable in $\Delta_2$.

**Proposition 6** Fix a morphism $\alpha : A^\Delta \to (H, V)$, with $(H, V)$ satisfying (1) and (2). For every $h \in H$, the forest language $\alpha^{-1}(h)$ is definable in $\Sigma_2$.

Before proving this Proposition, we show how it concludes the proof of Theorem 3. The nontrivial part is showing that every forest language $\alpha^{-1}(h)$ is also definable in $\Pi_2$, and not just $\Sigma_2$, as the proposition says (the rest follows by closure of $\Delta_2$ under boolean operations). But this is a consequence of finiteness of $H$:

$$ t \in \alpha^{-1}(h) \qquad \Leftrightarrow \qquad t \notin \bigcap_{g \neq h} \alpha^{-1}(g) \ , $$

since the intersection on the right-hand side is $\Sigma_2$, and therefore non-membership is a $\Pi_2$ condition.

The rest of this section is devoted to showing Proposition 6. The proof is by induction on two parameters: the first is the size of the algebra, and the second is the position of $h$ in a certain pre-order defined below. The second parameter corresponds to a bottom-up pass through the forest, as the types $h$ that are small in the pre-order correspond to forests that are close to the leaves. Moreover, for some types $h$ in the bottom-up pass, we will need a nested induction, involving a bottom top-down pass.

## 5 Bottom-up phase

We now define the pre-order on $H$, which is used in the induction proof of Proposition 6. We say that a type $h$ is *reachable from* a type $g$, and denote this by $g \sqsubseteq h$, if there is a context $v \in V$ such that $h = vg$. If $h$ and $g$ are mutually reachable from each other, then we write $h \sim g$. Note that $\sim$ is an equivalence relation. A type $h$ is said to be *maximal* if $h$ can be reached from all types reachable from $h$.

The proof of Proposition 6 is by induction on the size of the algebra $(H, V)$ and then on the position of $h$ in the reachability pre-order. The two parameters are ordered lexicographically, the most important parameter being the size of

the algebra. As far as $h$ is concerned, the induction corresponds to a bottom-up pass, where types close to the leaves are treated first.

Let then $h \in H$ be a type. By induction on Proposition 6, for each $g \sqsubseteq h$ with $g \not\sim h$, we have a $\Sigma_2$ formula defining the language of forests of type $g$. (The case when there are no such types $g$ corresponds to the induction base, which is treated the same way as the induction step.) In this section we will use these formulas to produce a $\Sigma_2$ formula defining those forests $s$ such that $\alpha(s) = h$.

In the following, we will be using two sets:

$$stab_V(h) = \{v : vh \sim h\} \subseteq V \qquad stab_H(h) = \{g : g + h \sim h\} \subseteq H .$$

The main motivation for introducing this notation is that equation (2) implies that they are both submonoids of $V$ and $H$, respectively.

**Lemma 7** The sets $stab_V(h), stab_H(h)$ only depend on the $\sim$-class of $h$. In particular, both sets are submonoids (of $V$ and $H$, respectively).

**Proof**
We prove the Lemma for $stab_V(h)$, the case of $stab_H(h)$ being similar. We need to show that if $h \sim h'$ then $stab_V(h) = stab_V(h')$. Assume $v \in stab_V(h)$. Then $vh \sim h$. Hence we have $u_1, u_2, u_3$ such that $h = u_1 vh, h = u_2 h'$ and $h' = u_3 h$. This implies that $h' = u_3 u_1 vu_2 h'$ and therefore $h' = (u_3 u_1 vu_2)^\omega h'$. From (2) we have that

$$h' = (u_3 u_1 vu_2)^\omega h' = (u_3 u_1 vu_2)^\omega v(u_3 u_1 vu_2)^\omega h' = (u_3 u_1 vu_2)^\omega vh' .$$

Hence $h'$ is reachable from $vh'$. Since $vh'$ is clearly reachable from $h'$, we get $h \sim h'$ and $v \in stab_V(h')$. ∎

Recall now the piece order $\preceq$ on $H$ from Definition 2, which corresponds to removing nodes from a forest. We say a set $F \subseteq H$ of forest types is *closed under pieces* if $h \preceq g \in F$ implies $h \in F$. A similar definition is also given for contexts. Another consequence of equation (2) is:

**Lemma 8** Both $stab_V(h), stab_H(h)$ are closed under pieces.

**Proof**
We consider only the case of $stab_V(h)$, the case of $stab_H(h)$ being similar. From the definition of piece we need to show that if $u \in stab_V(h)$ and $u' \preceq u$ then $u' \in stab_V(h)$. By definition we have a context $v$ such that $h = vuh$. We are looking for a context $w$ such that $wu'h = h$. From $h = vuh$ we get $h = (vu)^\omega h$. Hence by (2) we have $h = (vu)^\omega u'(vu)^\omega h = (vu)^\omega u'h$ as desired. ∎

We now consider two possible cases: either $h$ belongs to $stab_H(h)$, or it does not. In the first case we will conclude by induction on the size of the algebra while on the second case we will conclude by induction on the partial order $\sqsubseteq$. These are treated separately in Sections 5.1 and 5.2, respectively.

8

### 5.1 $h \notin stab_H(h)$

For $v \in V$, we write $K_v$ for the set of contexts of type $v$. For $g \in H$, we write $L_g$ for the set of forests of type $h$. For $g \in H$ and $F \subseteq H$, we write $L_g^F$ for the set of forests $t$ of type $h$ that be decomposed as $t = t_1 + \ldots + t_n$, with each $t_i$ a tree with of type in $F$.

Let $G$ be the set of forest types $g$ such that $h$ is reachable from $g$ but not vice-versa. By induction assumption, each language $L_g$ is definable in $\Sigma_2$, for $g \in G$. Our goal is to give a formula for $L_h$.

**Lemma 9** A forest has type $h$ if and only if it belongs to $L_h^G$ or a language $K_u a L_g^G$, with $u\alpha(a)g = h$ and $u \in stab_V(h)$.

**Proof**
Let $t$ be a forest of type $h$, and choose $s$ a subtree of $t$ that has type equivalent to $h$, but no subtree with a type equivalent to $h$. If such $s$ does not exist, then $t$ belongs to $L_h^G$ as a concatenation of trees with type in $G$. By minimality, $s$ must belong to some set $a L_g^G$. Let $p$ be the context such that $t = ps$. Since the type of $s$ is equivalent to $h$, and the type of $t$ is $h$, then the type $u$ of $p$ belongs to $stab_V(\alpha(s))$ which is the same as $stab_V(h)$ by Lemma 7. ∎

In Lemmas 10 and 11, we will show that the languages $K_u$ and $L_g^G$ above can be defined in $\Sigma_2$. To be more precise, we only give an over-approximation $\varphi_g^G$ of the language $L_g^G$, however all forests in the over-approximation have type $g$, which is all we need. Proposition 6 then follows by closure of $\Sigma_2$ under finite union and Fact 1.

We begin by giving the over-approximation of $L_g^G$.

**Lemma 10** For any type $g \in H$, there is a formula $\varphi_g^G$ of $\Sigma_2$ such that:

- Any forest $L_g^G$ satisfies $\varphi_g^G$; and
- Any forest satisfying $\varphi_g^G$ has type $g$.

**Proof**
The proof of the lemma is in two steps. In the first step, we introduce a condition (*) on a forest $t$, and show that: a) any forest in $L_g^G$ satisfies (*); and b) any forest satisfying (*) has type $g$. Then we will show that condition (*) can be expressed in $\Sigma_2$.

> (*) For some $m \leq n$, the forest $t$ can be decomposed, modulo commutativity, as the concatenation $t = t_1 + \cdots + t_n$ of trees $t_1, \ldots, t_n$, with types $g_1, \ldots, g_n$, such that
> 1. $g_1 + \cdots + g_m = g$.
> 2. Each type from $G$ is represented at most $\omega$ times in $g_1, \ldots, g_m$.
> 3. If a tree $s$ is a piece of $t_{m+1} + \cdots + t_n$, then $\alpha(s) \preceq g_i$ holds for some type $g_i$ that occurs $\omega$ times in the sequence $g_1, \ldots, g_m$.

9

We first show that condition (*) is necessary. Let $t_1, \ldots, t_n$ be all the trees in a forest from $t$, and let $g_1, \ldots, g_n \in G$ be the types of these trees. Without loss of generality, we may assume that trees are ordered so that for some $m$, each type of $g_i$ with $i > m$ already appears $\omega$ times in $g_1, \ldots, g_m$. It is not hard to see that identity (2) implies aperiodicity of the monoid $H$, i.e.

$$\omega \cdot f = \omega \cdot f + f \qquad \text{for all } f \in H . \tag{3}$$

In particular, it follows that $g = g_1 + \cdots + g_m$ since all of $g_{m+1}, \ldots, g_n$ are swallowed by the above. It remains to show item 3 of condition (*). Let then $s$ be the piece of a tree $t_i$ with $i > m$. We get the desired result since the type of $t_i$ already appears in $g_1, \ldots, g_m$.

We now show that condition (*) implies $\alpha(t) = g$. Let then $m \leq n$ and $t = t_1 + \cdots + t_n$ be as in (*). We will show that for any $j > m$, we have $g + g_j = g$, which shows that the type of $t$ is $g$. By item 3, $g_j \preceq g_i$ holds for some some type $g_i$ that occurs $\omega$ times in the sequence $g_1, \ldots, g_m$. By (3), we have $g = g + g_i = g + \omega \cdot g_i$. It therefore remains to show that $\omega \cdot g_i + g_j = \omega \cdot g_i$:

$$\omega \cdot g_i + g_j = \omega \cdot g_i + g_j + \omega \cdot g_i =$$
$$(\square + g_i)^\omega (\square + g_j)(\square + g_i)^\omega 0 = (\square + g_i)^\omega 0 = \omega \cdot g_i$$

In the above we have used identity (2). Note that the requirement in (2) was satisfied, since $g_j \preceq g_i$ implies $\square + g_j \preceq \square + g_i$.

It now remains to show that forests satisfying condition (*) can be defined in $\Sigma_2$. Note that $m$ cannot exceed $|G| \cdot \omega$, and therefore there is a finite number of cases to consider for $g_1, \ldots, g_m$. Fix some sequence $g_1, \ldots, g_m$. The only nontrivial part is to provide a $\Sigma_2$ formula that describes the set of forests $t_{m+1} + \ldots + t_n$ that satisfy item 3 of condition (*). From this construction, the formula for (*) follows by closure of $\Sigma_2$ under finite union and $\oplus$ (recall Fact 1), as well as the assumption that each type in $G$ can be defined in $\Sigma_2$.

In order to define forests as in item 3 we use a $\Pi_1$ formula that forbids the appearance of certain pieces of bounded size inside $t_{m+1} + \cdots + t_n$. Let $F$ be the types in $g_1, \ldots, g_m$ that appear at least $\omega$ times. We claim that a sequence of trees $t_{m+1} + \cdots + t_n$ satisfies item 3 if and only if it satisfies item 3 with respect to pieces $s$ that have at most $|H|^{|H|}$ nodes. This later property can be expressed by a $\Pi_1$ formula. The reason for this is that, thanks to a pumping argument, any tree has a piece that has the same type, but at most $|H|^{|H|}$ nodes. $\blacksquare$

**Lemma 11** For any $u \in stab_V(h)$, the context language $K_u$ is definable in $\Sigma_2$.

To prove this lemma, we will use a more general result, Proposition 12, stated below, but proved in Appendix B. We say a tree $t$ is a subtree of a context $p$ if $t$ is the subtree of some node in $p$ that is not an ancestor of the hole.

**Proposition 12** Let $F \subseteq H$ be a set of forest types definable in $\Sigma_2$ that is closed under pieces. For any $u \in V$, there is a $\Sigma_2$ formula that defines the set of contexts with type $u$ that have no subtree of type outside $F$.

10

**Proof** (of Lemma 11)

Let $F = stab_H(h)$. The result will follow from Proposition 12 once we show that a context in $K_u$ cannot have a subtree outside $F$, and that $F$ satisfies the conditions in the proposition.

By Lemma 8, the set $F = stab_H(h)$ is closed under pieces. We now show that $F \subseteq G$, and therefore each type in $F$ is definable in $\Sigma_2$. To the contrary, if $F$ would contain a type outside $G$, i.e. a type reachable from $h$, then by closure under pieces it would also contain $h$, contradicting our assumption on $h \notin stab_H(h)$. Finally, each subtree $t$ of a context in $stab_V(h)$ is a subtree—and therefore also a piece—of a tree in $stab_H(h) = F$. ∎

## 5.2 $h \in stab_H(h)$

**Lemma 13** If $h \in stab_H(h)$ then $(stab_H(h), stab_V(h))$ is a forest algebra.
**Proof**
We need to show that the two sets are closed under all operations.

$$stab_V(h)stab_V(h) \subseteq stab_V(h)$$
$$stab_H(h) + stab_H(h) \subseteq stab_H(h)$$
$$\square + stab_H(h) \subseteq stab_V(h)$$
$$stab_V(h)stab_H(h) \subseteq stab_H(h)$$

The first two of the above inclusions follow from Lemma 7. The third follows straight from the definition of $stab$. For the last inclusion, consider $v \in stab_V(h)$ and $g \in stab_H(h)$. We need to show that $vg \in stab_H(h)$. This means showing that $vg + h \sim h$. Since we have $g + h \sim h$ and $vh \sim h$ we have $u, u' \in V$ such that $h = u(g + h)$ and $h = u'vh$. Hence $h = u'vu(g + h)$ and $vg \preceq h$. We conclude using Lemma 8 and the fact that $h \in stab_H(h)$. ∎

We have two subcases depending whether $(stab_H(h), stab_V(h))$ is a proper subalgebra of $(H, V)$ or not.

Assume first that $h$ is not maximal. Hence there exists a type $g$ reachable from $h$ but not vice-versa. Let $u$ be a context such that $g = uh$. It is clear that $u$ is not in $stab_V(h)$. Therefore $(stab_H(h), stab_V(h))$ must be a proper subalgebra of $(H, V)$, as we have that $stab_V(h) \subsetneq V$. Furthermore, this algebra contains all pieces of $h$; so it still recognizes the language $\alpha^{-1}(h)$; at least as long as the alphabet is reduced to include only letters that can appear in $h$. We can then use the induction assumption on the smaller algebra to get the $\Sigma_2$ formula required in Proposition 6.

If $h$ is maximal then the algebra is not proper and we need to do more work. The $\Sigma_2$ formula required in Proposition 6 is obtained by taking $v = \square$ in the proposition below. The proof of this proposition introduces a pre-order on $V$ and is done by induction using that pre-order simulating a top-down process. The details can be found in Appendix C.

**Proposition 14** Fix a morphism $\alpha : A^\Delta \to (H, V)$, a context type $v \in V$ and a maximal forest type $h$. The following forest language is definable in $\Sigma_2$:

$$\{t : v\alpha(t) = h\}$$

# 6 Discussion

Apart from label tests, the signature we have used contains only the descendant relation. What about other predicates? For instance, if we add the lexicographic order on nodes, we loose commutativity $g + h = h + g$, although the remaining identity (2) remains valid. Is the converse implication true, i.e. can every language whose algebra satisfies (2) be defined by a $\Delta_2$ formula with the lexicographic and descendant order? Other predicates that could be added to the signature include: the closest common ancestor, next sibling or child. What is the expressive power of $\Delta_2$ in the other signatures?

Probably the most natural continuation would be an effective characterization of $\Sigma_2$. Note that this would strenghten our result: a language $L$ is definable in $\Delta_2$ if and only if both $L$ and its complement are definable in $\Sigma_2$. We conjecture that, as in the case for words [1], the characterization of $\Sigma_2$ requires replacing the equivalence in (2) by a one-sided implication, which says that a language definable in $\Sigma_2$ is closed under replacing $v^\omega$ by $v^\omega w v^\omega$, for $w \preceq v$.

# References

1. M. Arfi. Opérations polynomiales et hiérarchies de concaténation. *Theor. Comput. Sci.*, 91(1):71–84, 1991.
2. M. Benedikt and L. Segoufin. Regular tree languages definable in FO and in FO+mod. Manuscript, a preliminary version appeared in the proceeding of the Symposium on Theoretical Aspects of Computer Science (STACS'05), 2008.
3. M. Bojańczyk. Forest expressions. In *Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 146–160, 2007.
4. M. Bojańczyk. Two-way unary temporal logic over trees. In *Logic in Computer Science*, pages 121–130, 2007.
5. M. Bojańczyk and I. Walukiewicz. Characterizing EF and EX tree logics. *Theoretical Computer Science*, 358(2-3):255–273, 2006.
6. M. Bojańczyk and I. Walukiewicz. Forest algebras. In *Automata and Logic: History and Perspectives*, pages 107 – 132. Amsterdam University Press, 2007.
7. Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.
8. H. Straubing M. Bojańczyk, L. Segoufin. Piecewise testable tree languages. submitted.
9. R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.
10. J.-É. Pin. Logic, semigroups and automata on words. *Annals of Mathematics and Artificial Intelligence*, 16:343–384, 1996.
11. J.-É. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:1–30, 1997.
12. M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.
13. T. Schwentick, D. Thérien, and H. Vollmer. Partially-ordered two-way automata: A new characterization of da. In *Developments in Language Theory*, pages 239–250, 2001.
14. I. Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222, 1975.

15. D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *ACM Symposium on the Theory of Computing*, pages 256–263, 1998.

16. T. Wilke. Classifying discrete temporal properties. In *Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46, 1999.

# A    Correctness of the identities

In this appendix, we prove Lemma 5:

**Lemma 5** Let $\varphi$ be a formula of $\Sigma_2$ and let $p \preceq q$ be two contexts. For $n \in \mathbb{N}$ sufficiently large, forests satisfying $\varphi$ are closed under replacing $p^n p^n$ with $p^n q p^n$.

**Proof**
Any first-order definable tree language is aperiodic, i.e. a subcontext $p^m$ can be replaced by $p^k$ without affecting membership in the language, for any $k, m$ sufficiently large.

Let the quantifier prefix of $\varphi$ be $\exists x_1 \cdots x_i \forall y_1 \ldots y_j$. Let $t$ be a forest satisfying $\varphi$. We need to show that $\varphi$ still holds after a subcontext $p^n p^n$ of $t$ is replaced with $p^n q p^n$. Consider now an assignment of the variables $x_1, \ldots, x_i$ in $t$ that makes $\varphi$ true. The idea is that if $n$ is sufficiently large, then there are at least $2k$, for some large $k$, consecutive copies of $p$ where no existential variables are used. If $k$ is sufficiently large, the $2k$ copies can be replaced by $2k+1$ copies using aperiodicity, and then the one in the middle can be replaced by $q \preceq p$, without affecting validity of the part of the formula that only uses universal quantifiers. Once again using aperiodicity, we can modify the context so that the new $q$ is surrounded by exactly $n$ copies of $p$ on each side. ∎


# B    Treating contexts like words

In this appendix, we prove Proposition 12.

**Proposition 12** Let $F \subseteq H$ be a set of forest types definable in $\Sigma_2$ that is closed under pieces. For any $u \in V$, there is a $\Sigma_2$ formula that defines the set of contexts with type $u$ that have no subtree of type outside $F$.

The basic idea is that a context is treated as a word (over smaller contexts), and word arguments can be used to reason about this decomposition.

The proof strategy is as follows. First, in Section B.1, we present the characterization of $\Delta_2$ for words, which was shown by Pin and Weil in [11]. This characterization is slightly strengthened to include what we call stratified monoids, which are used to model the contexts that appear in Proposition 12. Then, in Section B.2, we apply the word result (in its strengthened form) to prove Proposition 12.


## B.1    $\Delta_2$ for words

In this section we present the characterization of $\Delta_2$ for words, extended to stratified monoids. A *stratified monoid* is a monoid $M$ along with an pre-order $\preceq$ that satisfies the following property:

$$m^\omega n m^\omega = m^\omega \qquad \text{for } n \preceq m$$

14

**Proposition 15** Let $A$ be an alphabet (possibly infinite), and let $\beta : A^* \to M$ be a morphism into a stratified monoid $(M, \preceq)$ that satisfies the identity

$$(mn)^\omega m (mn)^\omega = (mn)^\omega \tag{4}$$

For any $m \in M$, the language $\beta^{-1}(m)$ is defined by a finite union of expressions

$$A_0^* B_1 A_1^* \cdots B_i A_i^*$$

where each $B_j$ is of the form $A \cap \beta^{-1}(n)$ for some $n \in M$, and each $A_j$ is of the form $A \cap \beta^{-1}(N)$ for some $N \subseteq M$ downward closed under $\preceq$.

The difference between the above result and the main technical result in Pin and Weil is twofold. First, we use infinite alphabets here. This does not change anything in the proof, since two letters are treated the same way as long as they have the same image under $\beta$; however, we will apply this proposition later on to an infinite alphabet. The second difference is that we use stratified monoids to get a stronger conclusion, where the letters in the blocks $B_i^*$ are downward closed.

Our proof is an adaptation of a proof of Thérien and Wilke in [15], which analyzed the languages recognized by semigroups in DA. Both of the extensions to infinite alphabets mentioned above are simple; however we need the stronger result in our context.

Before proving this result, we show that it gives the characterization of $\Delta_2$ presented by Pin and Weil:

**Corollary 16 (Pin and Weil [11])** A word language (over a finite alphabet) is definable $\Delta_2$ if and only if its syntactic monoid satisfies the identity (4).

**Proof**
The only if implication is shown using a standard Ehrenfeucht-Fraïsse argument, we only consider the if implication.

Let then $L \subseteq A^*$ be a language recognized by a morphism $\beta : A^* \to M$, with $M$ satisfying (4). We can see this $M$ as a stratified monoid under the identity pre-order. By applying Proposition 15, we see that each inverse image $\beta(m)$ is defined by an expression as in Prop 15 (the downward closure is a vacuous condition, since the order is trivial). Since each such expression is clearly expressible in $\Sigma_2$, we get that $L$ is definable in $\Sigma_2$. Furthermore, by Proposition 15 also the complement of $L$ is definable in $\Sigma_2$, and therefore $L$ is also definable in $\Pi_2$. ∎

We now proceed to the proof of Proposition 15. The proof is the same as that of Thérien and Wilke, i.e. by induction on the size of $\beta(A) \subseteq M$, or equivalently the number of elements in the monoid that correspond to single letters. In the original proof of Thérien and Wilke, the induction was simply on the size of the alphabet, but this will not work here, since the alphabet is infinite.

We use the term $\Sigma_2$-*expression* for expressions as in the statement of Proposition 15. It is not difficult to show that languages defined by $\Sigma_2$-expressions are closed under union, intersection and concatenation.

We will use the following notation. Given two elements $m$ and $n$ of $M$ we say that $m \sim^L n$ if there exist $k, l \in M$ such that $m = kn$ and $n = lm$. We say that $m \sim^R n$ if there exist $k, l \in M$ such that $m = nk$ and $n = ml$.

A classical consequence of aperiodicity, itself a consequence of (4), is:

$$m \sim^R n \ \wedge \ m \sim^L n \ \Rightarrow \ m = n \qquad \text{for } m, n \in M \ . \tag{5}$$

**Lemma 17** For all $m \in M$, the language $K_m = \{w : m\beta(w) \sim^R m\}$ is definable by a $\Sigma_2$-expression.

**Proof**

We will use the following property of monoids satisfying (4), which can be proved along the lines of Lemma 7.

$$m \sim^R n \sim^R mk \ \Rightarrow nk \sim^R n \qquad \text{for } m, n, k \in M \ . \tag{6}$$

Thanks to (6), a word belongs to $K_m$ if and only if all of its letters are in

$$A_m = A \cap \beta^{-1}(N) \qquad \text{where } N = \{n : mn \sim^R m\} \ .$$

To conclude, we need to show that $N$ is closed under $\preceq$. Indeed, let $k \preceq n$ and let $n \in N$. By assumption on the monoid being stratified, we have $n^\omega k n^\omega = n^\omega$. In particular, we have

$$mn^\omega k n^\omega \sim^R mn^\omega \sim^R m \ .$$

From the above it follows that $mn^\omega k \sim^R m$, which gives $mk \sim^R m$ by (6), and hence $k \in N$. ∎

Note that by (6) we have $K_n = K_m$ whenever $m \sim^R n$.

**Lemma 18** For any $m \in M$, the following can be defined by a $\Sigma_2$-expression:

$$L_m = \{a_1 \cdots a_i : \ \beta(a_1 \cdots a_i) = m \text{ and } \beta(a_1 \cdots a_{i-1}) \not\sim^R m\} \ ,$$

Before we give the proof, we show how it concludes the proof of Proposition 15. By taking $K_m$ to be the language from Lemma 17, we get

$$\{w : \beta(w) \sim^R m\} = \bigcup_{n : n \sim^R m} L_n K_m \ .$$

Since $\Sigma_2$-expressions are closed under union and concatenation, the above language is definable by a $\Sigma_2$-expression thanks to Lemmas 17 and 18. Using a symmetric version of Lemma 17 and Lemma 18 for $\sim^L$, we can get an expression for $\{w : \beta(w) \sim^L m\}$. Therefore, we can conclude, since by (5),

$$\beta^{-1}(m) = \{w : \beta(w) \sim^R m\} \cap \{w : \beta(w) \sim^L m\}$$

16

and the result follows by closure of $\Sigma_2$-expressions under intersection.

**Proof** (of Lemma 18)

We say that $m$ is *a prefix* of $n$ if there exists $k \in M$ such that $n = mk$. This defines a pre-order in $M$. The proof is by induction on the position of $m$ relative to this pre-order.

The induction base is when $m$ is minimal: If $m = nk$ then $n \sim^R m$. In this case the language $L_m$ contains at most the empty word, since the condition on $a_1 \cdots a_{i-1}$ is infeasible. Clearly both languages $\emptyset$ and $\{\epsilon\}$ are $\Sigma_2$-expressions.

Assume now that $m$ is not minimal. We have that

$$L_m = \bigcup L_n K_{n,k} a \qquad \text{with } K_{n,k} = \{w \in A^* : n\beta(w) = k\}$$

where the unions is taken for $n, k \in M$ with $n \sim^R k$, $k \not\sim^R m$ a prefix of $m$ and for $a \in A$ with $k\beta(a) = m$. By induction the language $L_n$ is definable by a $\Sigma_2$-expression. It is also clear that $K_{n,k} \subseteq K_n$. Recall from the proof of Lemma 17 that $K_n = A \cap \beta^{-1}(N)$ for some $N \subseteq M$. Because $n$ is a strict prefix of $m$ ($m$ is not minimal) we have $\beta(K_n) \subsetneq A$. Therefore we also have $\beta(K_{n,k}) \subseteq A$ and by induction on the size of the alphabet in Proposition 15 we obtain a $\Sigma_2$-expression for $K_{n,k}$. This concludes the proof of this lemma as $\Sigma_2$-expressions are closed under concatenation and union. ∎

## B.2   Proof of Proposition 12

We now proceed to show how the word result stated in Proposition 15 can be lifted to the context result in Proposition 12. We fix $F$ and $u$ as in the statement of the Proposition 12. We want to write a $\Sigma_2$ formula that tells us if a context has type $u$; however we know that every forest that is used on the side of this context belongs to $F$.

The basic idea is that we treat the context as a word over an infinite alphabet, which we call $B$. This alphabet has two types of letters. Both types are contexts:

- Contexts of the form $a\square$, for $a \in A$.
- Contexts of the form $t + \square$ or $\square + t$, for $t$ a tree over $A$ with $\alpha(t) \in F$.

Consider now the morphism $\beta : B \to V$, which is simply $\alpha$ restricted do the contexts in $B$. Every context $p$ in the language $K$ from Proposition 12 can be decomposed as $p = b_1 \cdots b_n \in B^*$. In particular, we have

$$K = \beta^{-1}(u) .$$

Therefore, we will conclude the proof of Proposition 12 if we give a $\Sigma_2$ formula defining the set of contexts $\beta^{-1}(u)$.

We can treat $V$ as a stratified monoid, by using the piece relation $\preceq$ as the pre-order. By applying Proposition 15, we see that the inverse image $\beta^{-1}(u)$ can be presented as a finite union of expressions of the form:

$$(B \cap \beta^{-1}(U_0))^*(B \cap \beta^{-1}(u_1)) \cdots (B \cap \beta^{-1}(u_n))(B \cap \beta^{-1}(U_n)) ,$$

where $u_1, \ldots, u_n$ are elements of $V$, and $U_1, \ldots, U_n$ are a subsets of $V$ that are downward closed under $\preceq$. By Fact 1 and commutativity of $H$, it is enough to give a $\Sigma_2$ formula for two types of expression:

$$(B \cap \beta^{-1}(W))^* \qquad \text{with } W \subseteq V \text{ downward closed under } \preceq$$
$$B \cap \beta^{-1}(w) \qquad \text{with } w \in V$$

We first deal with the second type of expression. Recall that there are two types of letters $B$: either $a\square$, or $t + \square$ with $\alpha(t) \in F$. The first type can be easily checked in $\Sigma_2$. Since a $\Sigma_2$ formula can easily enforce that a context is of the form $t + \square$ or $\square + t$, with $t$ a tree, the second type can be described thanks to the assumption on types of $F$ being definable in $\Sigma_2$.

We are now left with the expressions of the first type, $(B \cap \beta^{-1}(W))^*$. These can actually be defined already in $\Pi_1$, as shown by the following lemma:

**Lemma 19** If $W \subseteq V$ is downward closed under $\preceq$, then the context language $(B \cap \beta^{-1}(W))^*$ is definable in $\Pi_1$.

**Proof**

Let $G \subseteq H$ be the set of types $g$ such that $g + \square \in W$. Since $W$ is downward closed, then so is $G$.

Thanks to a pumping argument, any tree has a piece that has the same type, but at most $|H|^{|H|}$ nodes. Let $T = \{t_1, \ldots, t_n\}$ be the trees with at most $|H|^{|H|}$ nodes for all types in $H \backslash F$. We claim that a context $p$ belongs to $(B \cap \beta^{-1}(W))^*$ if and only if it does not contain any of the pieces $\square + t_1, \ldots, \square + t_n$. The statement of the lemma follows, since the latter condition can clearly be expressed in $\Pi_1$.

The only if part of the claim follows immediately from closure of $G$ under $\preceq$: if a context $p$ contains a piece $\square + t_i$, then it can be decomposed as $p_1(\square + t)p_2$, with $t_i$ a piece of $t$. Since $\beta(t_i)$ is outside $G$, then so is $\beta(t)$ by closure of $G$ under pieces.

For the converse implication in the claim, we use the following obvious fact: if all the pieces of a tree $t$ of size $|H|^{|H|}$ have a type in $G$, then so does $t$. ∎

## C Top-down phase

Recall that in the bottom-up phase, in Section 5.2, we did not manage to find a $\Sigma_2$ formula in the case when $h$ was a maximal forest type $h \in H$, i.e. when any type reachable from $h$ can reach $h$ back again. The $\Sigma_2$ formula required in Proposition 6 is obtained by taking $v = \square$ in the proposition below.

**Proposition 14** Fix a morphism $\alpha : A^\Delta \to (H, V)$, a context type $v \in V$ and a maximal forest type $h$. The following forest language is definable in $\Sigma_2$:

$$\{t : v\alpha(t) = h\}$$

The rest of this section is devoted to proving the above proposition. The proof will be by induction on $v$, with respect to a new pre-order on contexts.

We say that a context $u$ is a *prefix* of a context $v$ if there exists a context $w$ such that $v = uw$, and we denote this by $u \leq v$. We overload the use of $\sim$ and denote by $\sim$ the equivalence relation induced by $\leq$. The proof of Proposition 14 is by induction on the index of $v$ relative to $\leq$ starting from the maximal ones towards the minimal ones, i.e. the induction ends at $v = \square$.

**The induction base** The base case is when $v$ is maximal: For all $u$, if $u$ is a prefix of $v$ then $u \sim v$. We say a context $u$ is *constant* if $vg = vh$ holds for all forest types $g, h \in H$.

**Lemma 20** A context is maximal if and only if it is constant.

**Proof**
If a context $v$ is constant, then $vu = v$ holds for all $u$, and therefore $v$ is maximal.

For the converse implication, let $h_1, \ldots, h_n$ be all the forest types in $H$. If $v$ is maximal, then there must be some some $u \in V$ with

$$v = v(\square + h_1 + \cdots + h_n)u \ .$$

Let now $g \in H$ be any forest type. Let $i$ be chosen so that $h_i = \omega(ug)$. From aperiodicity of $H$, i.e. from the identity (3), we get $h_i + ug = h_i$. By commutativity of $H$ we get the desired result.

Therefore, we have

$$vg = v(ug + h_1 + \cdots + h_n) = v(h_1 + \cdots + h_n) \ ,$$

which shows that $vg$ does not depend on $g$. ∎

In particular, if a context $v$ is maximal, then the set $\{t : v\alpha(t) = h\}$ is either empty or contains all forests, both cases easily definable by a $\Sigma_2$ formula.

**The induction step** We now proceed to the induction step, and assume that $v$ is not maximal. As previously, we fix $G \subseteq H$ to be the set of forest types $g$ such that $h$ is reachable from $g$ but not vice versa. Note that because $h$ is maximal, $h \sim h_1 + \cdots + h_n$ where $h_1, \cdots, h_n$ are all the types of $H$. Hence $h$ is reachable from any type $g$. In particular, any forest type $g$ either belongs to $G$, or $g \sim h$. From the previous section we know that for each $g \in G$ the set $L_g$ of forests with type $g$ can be defined in $\Sigma_2$.

We define stabilizers for context types in the same way they were defined for forest types:

$$stab_V(v) = \{w : vw \sim v\} \qquad stab_H(v) = \{h : (h + \square) \in stab_V(v)\}$$

As in the previous section, it follows from (2) that both sets are submonoids (of $V$ and $H$ respectively), and that both are closed under pieces. A useful consequence of the assumption on $v$ not being maximal is that the set of contexts with type $v$ can be defined in $\Sigma_2$.

**Lemma 21** If $u$ is not maximal, then $K_u$ is definable in $\Sigma_2$.

**Proof**
We first show that $stab_H(u) \subseteq G$. Assume to the contrary, that the inclusion does not hold, i.e. $stab_H(u)$ contains some type $g \sim h$. Since $h \sim h_1 + \cdots + h_n$ and $stab_H(u)$ is closed under pieces, $stab_H(u)$ would also contain all forest types. But then $\square + h_1 + \cdots + h_n$ would be in $stab_H(u)$ and by reasoning as in the proof of Lemma 20, the context type $u$ would be constant, and therefore maximal.

To show that $K_u$ is definable in $\Sigma_2$, we will use Proposition 12, with $F = stab_H(u)$. The assumptions in the proposition are satisfied for the same reasons as in Lemma 11. ∎

Recall that we want to give a $\Sigma_2$ formula for the language

$$\{t : v\alpha(t) = h\} .$$

Similar to what we did in Lemma 9, we are going to decompose the forest $t$ into simpler pieces, which can be defined in $\Sigma_2$. In the following lemma, we write $L_g$ (respectively, $M_g$) for the set of all forests (respectively, trees) of type $g$.

**Lemma 22** A forest $t$ satisfies $v\alpha(t) = h$ if and only if, for some label $a$ and $u \in stab_V(v)$, it belongs to one of the three kinds of languages below

| | | |
|---|---|---|
| $K_u a L$ | for $\alpha(a) \notin stab_V(v)$ | with $L = \{t : vu\alpha(a)\alpha(t) = h\}$ |
| $K_u a (M_g \oplus L)$ | for $g \in G \setminus stab_H(v)$ | with $L = \{t : vu\alpha(a)(g + \alpha(t)) = h\}$ |
| $K_u a (M_{h_1} \oplus L_{h_2})$ | for $h_1 \sim h \sim h_2$ | with $vu\alpha(a)(h_1 + h_2) = h$ |

**Proof**
Assume the forest $t$ is not in any language of the first kind. Then it has a node $x$ such that if $x$ and its subtree are replaced by a hole then the corresponding context is in $stab_V(v)$, but that there is no successor of $x$ with this property. Then $t$ is of the form $t = pa(t_1 + \cdots + t_n)$ where the $a$ is the position of $x$, and $t_1, \ldots, t_n$ trees. Assume that at most one tree $t_i$ has a type outside $stab_H(v)$. Then by replacing $t_i$ with a hole we have extended $p$ to a new context whose type remains in $stab_V(v)$, contradicting our hypothesis of maximality of $p$. Therefore, we can assume that there are at least two trees $t_i, t_j$ with types outside $stab_H(v)$. If at least one of these trees has a type in $G$, then the whole tree is captured by an expression of the second kind. Otherwise, we use the third kind. ∎

We will now show that each of the languages in the above lemma can be defined in $\Sigma_2$. Note that if $u$ belongs to $stab_V(v)$, then it cannot be maximal, since otherwise it would be constant, and therefore so would be $vu$, contradicting the assumption on $v$ being non-maximal. Therefore, by Lemma 21, each of the context languages $K_u$ above can be defined in $\Sigma_2$. It remains to show that the other languages above can be defined in $\Sigma_2$.

We begin with the expression of the first kind. By $\alpha(a) \notin stab_V(v)$, we get $vu\alpha(a) \not\sim v$. Therefore, we can use the induction hypothesis in Proposition 14 to obtain a $\Sigma_2$ formula for $L$. We conclude using Fact 1.

The second kind is treated in a similar way.

It is the third kind that requires more effort. The problem is that the forest types $h_1, h_2$ are maximal, and therefore we do not know how to define the tree language $M_{h_1}$ or the forest language $L_{h_2}$ in $\Sigma_2$. Our solution is to write a $\Sigma_2$ formula that only approximates the values of $h_1, h_2$, however the approximation is good enough for the purposes of Lemma 22.

For two forest types $h_1, h_2$, we write $h_1 \approx h_2$ if $vuh_1 = vuh_2$ holds for every context $u$ such that $vu \not\sim v$. This is an equivalence relation.

**Lemma 23** For each maximal forest type $f$, there is a $\Sigma_2$ formula defining the language $L_f^{\approx}$ of forests whose type is maximal and equivalent to $f$ under $\approx$.

**Proof**

First note that each equivalence class of $\approx$ can be defined by a $\Sigma_2$ formula. This is because each equivalence class is an intersection of languages obtained from induction assumption in Proposition 14.

It remains to show that the set of forests with a maximal type can be defined in $\Sigma_2$. By looking for the smallest subtree with a maximal type, every tree $t$ with maximal type can be decomposed as $t = pa(t_1 + \ldots + t_n)$, with the trees $t_1, \ldots, t_n$ all having non-maximal types, but $a(t_1 + \ldots + t_n)$ having a maximal type. We use Lemma 10 and Fact 1 to witness such a decomposition with a $\Sigma_2$ formula. ∎

Since the set of trees is definable in $\Sigma_2$, then also the intersection $M_f^{\approx}$ of $L_f^{\approx}$ with the set of trees is definable in $\Sigma_2$.

The following lemma shows that whenever two maximal types appear next to each other as arguments of a context with prefix $v$, it is only their equivalence class under $\approx$ that is important for the result.

**Lemma 24** If $h_1 \approx h_1'$ and $h_2 \approx h_2'$ are maximal forest types, then

$$vu(h_1 + h_2) = vu(h_1' + h_2') \qquad \text{for any } u \in V .$$

**Proof**

By commutativity, it suffices to consider the case when $h_1 = h_1'$. Since $h_1$ is maximal, it cannot belong to $stab_H(v)$. In particular, $v \not\sim vu(h_1 + \square)$. Therefore, by definition of $h_2 \approx h_2'$,

$$vu(h_1 + h_2) = vu(h_1 + \square)h_2 = vu(h_1 + \square)h_2' = vu(h_1 + h_2') .$$

∎

The above lemma shows that the union of languages of the third kind in Lemma 22 is the same as the union of languages

$$K_u a(M_{h_1}^{\approx} \oplus L_{h_2}^{\approx})$$

for $h_1, h_2$ maximal types and $u \in stab_V(v)$ such that $u\alpha(a)(h_1 + h_2) = h$. This union is definable in $\Sigma_2$ thanks to Lemma 23 and Fact 1.