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 Δ_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 Δ_2. This characterization is in terms of algebraic equations. Over words, the class of word languages definable in Δ_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 two 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 Σ_n if they begin with $\exists$, and Π_n if they begin with $\forall$. For instance, the word property "some position has label a" can be defined by a Σ_1 formula $\exists x.\ a(x)$, while the language a^*ba^* can be defined by the Σ_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

* Work partially funded by the AutoMathA programme of the ESF, the PHC programme Polonium, and by the Polish government grant no. N206 008 32/0810

definable in Σ_1 (resp. Π_1) if and only if it is closed under inserting (removing) letters. Both properties can be tested in polynomial time based on a recognizing automaton, or semigroup. However, just above Σ_1, Π_1, and even before we get to Σ_2, Π_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 Σ_1 formulas if and only if its syntactic monoid is $\mathcal{J}$-trivial. Above the boolean combination of Σ_1, we find Δ_2, i.e. languages that can be defined simultaneously in Σ_2 and Π_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 Δ_2 if and only if its syntactic monoid is in DA. The limit of our knowledge is level Σ_2: it is decidable if a language can be defined on level Σ_2 [1, 11], but there are no known decidable characterization for boolean combinations of Σ_2, for Δ_3, for Σ_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 Σ_1 and Π_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 Σ_1 formulas. There is no known characterization of tree languages definable in Σ_n for $n \geq 2$.

The contribution of this paper is a decidable characterization of tree languages definable in Δ_2, i.e. definable both in Σ_2 and Π_2. As we signaled above, for word languages the class Δ_2 is 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 F and 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 Δ_2. A characterization of Δ_2 was left as an open problem, one which is solved here.

We provide an algebraic characterization of tree languages definable in Δ_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.

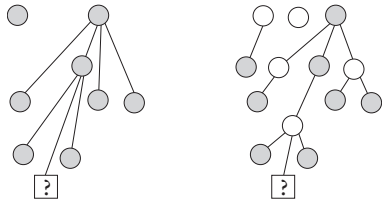
A set L of forests over A is called a *forest language.*

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 $\Box$, 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' if s, t can be decomposed as $s = pt$ and $s' = pat$ for some contexts p, some label a, and some forest t. 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 Σ_2 formula is a formula of the form

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

where γ is quantifier free. Properties defined in Σ_2 are closed under disjunction and conjunction, but not necessarily negation. The negation of a Σ_2 formula is called a Π_2 formula, equivalently this is a formula whose quantifier prefix is $\forall^*\exists^*$. A forest property is called Δ_2 if it can be expressed both by a Σ_2 and a Π_2 formula.

The problem We want an algorithm deciding whether a given regular forest language is definable in Δ_2.

Notice that the forest property of "being a tree" is definable in Δ_2. The Σ_2 formula says there exists a node that is an ancestor of all other nodes, while the Π_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 Δ_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 ω means that the identity holds for sufficiently large powers (in different settings, ω is defined in terms of idempotent powers, but the condition on sufficiently large powers is good enough here). 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 whether a given regular word language is definable in Δ_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.

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 Σ_2 Most of the proofs in the paper will work with Σ_2 formulas. We present some simple properties of such formulas in this section.

Apart from defining forest languages, we will also be using Σ_2 formulas to define languages of contexts. To define a context language we use Σ_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 Σ_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 $\Box$ 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, $h \cdot \Box = h$ 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 $(\Box + g)$ and $(g + \Box)$ defined by $(\Box + g)h = h + g, (g + \Box)h = g + h$ for all $h \in H$.

A morphism $\alpha : (H_1, V_1) \to (H_2, V_2)$ of forest algebras is actually a pair (γ, δ) 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 α.

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 a forest algebra which we denote A^Δ.

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 [6].

Given any finite monoid M, there is a number $\omega(M)$ (denoted by ω when M is understood from the context) such that for all element x of M, x^ω 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^ω 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^Δ, and the quotient (H_L, V_L) is called the *syntactic forest algebra* of L. Each equivalence class of $\sim_L$ is called a *type*.

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 Δ_2

In this section we present the main result of the paper: a characterization of Δ_2 in terms of two identities.

Theorem 3
A forest language is definable in Δ_2 if and only if its syntactic forest algebra satisfies the following identities:

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

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

Corollary 4 It is decidable whether a forest language can be defined in Δ_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 Δ_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 proof is standard.

Lemma 5 Let φ be a formula of Σ_2 and let $p \preceq q$ be two contexts. For $n \in \mathbb{N}$ sufficiently large, forests satisfying φ are closed under replacing $p^n p^n$ with $p^n q p^n$.

The rest of the paper 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 Δ_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 Σ_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 Π_2, and not just Σ_2, as the proposition says (the rest follows by closure of Δ_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 Σ_2, and therefore non-membership is a Π_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 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, using Proposition 6, for each $g \sqsubseteq h$ with $g \not\sim h$, we have a Σ_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 Σ_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 $h = u_1vh, h = u_2h'$ and $h' = u_3h$. This implies that $h' = u_3u_1vu_2h'$ and therefore $h' = (u_3u_1vu_2)^\omega h'$. From (2) we have that

$$h' = (u_3u_1vu_2)^\omega h' = (u_3u_1vu_2)^\omega v(u_3u_1vu_2)^\omega h' = (u_3u_1vu_2)^\omega vh' \ .$$

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 in 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.

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 can 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 Σ_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 aL_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 Σ_2. To be more precise, we only give an over-approximation φ_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 Σ_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 φ_g^G of Σ_2 such that:

- Any forest L_g^G satisfies φ_g^G; and
- Any forest satisfying φ_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 Σ_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 ω 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 ω times in the sequence $g_1, \ldots, g_m$.

We first show that condition (*) is necessary. Let $t_1, \ldots, t_n$ be all the trees in a forest 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 ω 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 ω 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 $\omega \cdot g_i + g_j = \omega \cdot g_i$:

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

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

It now remains to show that forests satisfying condition (*) can be defined in Σ_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 Σ_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 Σ_2 under finite union and $\oplus$ (recall Fact 1), as well as the assumption that each type in G can be defined in Σ_2.

In order to define forests as in item 3 we use a Π_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 ω 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. The latter property can be expressed by a Π_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. ∎

Lemma 11 For any $u \in stab_V(h)$, the context language K_u is definable in Σ_2.

To prove this lemma, we will use a more general result, Proposition 12, stated below. The proof of this Proposition will appear in the journal version of this paper. 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 Σ_2 that is closed under pieces. For any $u \in V$, there is a Σ_2 formula that defines the set of contexts with type u that have no subtree of type outside F.

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 Σ_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.

$$\begin{aligned} stab_V(h)stab_V(h) &\subseteq stab_V(h) \\ stab_H(h) + stab_H(h) &\subseteq stab_H(h) \\ \Box + stab_H(h) &\subseteq stab_V(h) \\ stab_V(h)stab_H(h) &\subseteq stab_H(h) \end{aligned}$$

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 Σ_2 formula required in Proposition 6.

If h is maximal then the algebra is not proper and we need to do more work. The Σ_2 formula required in Proposition 6 is obtained by taking $v = \Box$ 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 are omitted here and will appear in the journal version of this paper.

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 Σ_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 lose 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 Δ_2 formula with the lexicographic and descendant order? What is the expressive power of Δ_2 in the other signatures, with predicates such as the closest common ancestor, next sibling or child?

Probably the most natural continuation would be an effective characterization of Σ_2. Note that this would strenghten our result: a language L is definable in Δ_2 if and only if both L and its complement are definable in Σ_2. We conjecture that, as in the case for words [1], the characterization of Σ_2 requires replacing the equivalence in (2) by a one-sided implication, which says that a language definable in Σ_2 is closed under replacing v^ω 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 (preliminary version in 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. In *Logic in Computer Science*, 2008
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 *Devel. 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 *STOC*, 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.