Forest Algebras

Mikolaj Bojanczyk, Igor Walukiewicz

Forest Algebras

Mikolaj Bojańczyk and Igor Walukiewicz

October, 2006

Abstract: If in a transformation semigroup we assume that the set being acted upon has a semigroup structure, then the transformation semigroup can be used to recognize languages of unranked trees. This observation allows us to examine the relationship connecting languages of unranked trees with standard algebraic concepts such as aperiodicity, idempotency, commutativity and wreath product. In particular, we give algebraic characterizations of first-order logic, chain logic, CTL* and PDL. These do not, however, yield decidability results.

1 Introduction

There is a well-known decision problem in formal language theory:

> Decide if a given a regular language of finite binary trees can be defined by a formula of first-order logic with three relations: ancestor, left and right successor.

If the language is a word language (there is only one successor relation in this case) the problem is known to be decidable thanks to fundamental results of Schützenberger [16] and McNaughton and Papert [13]. The problem is also decidable for words when only the successor relation is available [19, 1]. However, no algorithm is known for the case of tree languages, see [12, 15, 3, 2] for some results in this direction.

There is a large body of work on problems of the type: decide if a given regular word language can be defined using such and such a logic [6, 14, 17, 20, 21, 23]. Most of the results have been obtained using algebraic techniques of semigroup theory. Recently, there has even been some progress for tree languages [22, 11, 4, 2]. There is, however, a feeling that we still do not have the right algebraic tools to deal with tree languages. In this paper we propose an algebraic framework, called unranked tree algebras, and study the notion of recognizability in this framework. We wanted it to be as close to the word case as possible to benefit from the rich theory of semigroups. The main result of the paper serves as an example of this close connection. We show how the notion of wreath product of transformation semigroups allows to capture different tree logics.

Forest algebras are defined for unranked trees, where a node may have more than two successors, which are ordered. This more general (more general than, say, binary trees) setting is justified by cleaner definitions, where semigroup theory can be used more easily.

We begin our discussion of forest algebras with the free forest algebra. Just as the set of nonempty words is the free semigroup, the free forest algebra is going to be the set of (nonempty) forests. For finite words, there is one natural semigroup structure: concatenation of words. For unranked, ordered, finite forests there are two natural semigroups:

- *Horizontal free semigroup.* Forests with concatenation.
- *Vertical free semigroup.* Contexts – forests with a single hole in some leaf – along with context composition.

The two semigroups are linked by an action of contexts on forests: if θ is a context and $\vec{t}$ is a forest then $\theta(\vec{t})$ is a forest obtained by substituting $\vec{t}$ in the hole of θ (see Figure 1).

In the case of words, a language of finite words induces a congruence, the Myhill-Nerode equivalence relation, which has finite index if the subset is regular. The same concepts apply to forest algebras, except that we get two congruences: one for the vertical semigroup and one for the horizontal semigroup. A regular language of finite forests can be thus seen as one where both congruences are of finite index.

An important property of a forest algebra is that it is a special case of a transformation semigroup. Recall that a *transformation semigroup* is a semigroup along with an action over a set. In the forest algebra, the acting semigroup is the set of contexts, while that set acted upon is the set of forests (which itself is equipped with a semigroup structure).

There is a well-developed theory of transformation semigroups that is useful in classifying regular word languages. We hope that this theory might extend to the case of trees. The point of this paper is to present some preliminary results in this direction. We show how logical properties of a tree language, such as being definable in first-order logic, correspond to algebraic properties of the language's tree algebra.

Acknowledgments We would like to thank Olivier Carton, Jean-Eric Pin, Thomas Schwentick, Luc Segoufin, Howard Straubing and Pascal Weil for their helpful comments.

1.1 Preliminaries

For trees, we use an alphabet with two types of letters: letters A for leaves and letters B for inner nodes.

Definition 1 Trees and forests over (A, B) are defined as follows:

- Every $a \in A$ is a tree (and therefore also a forest);

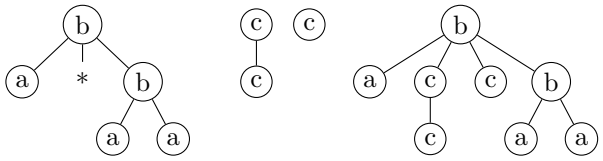
- If $\vec{s}, \vec{t}$ are forests, then $\vec{s} + \vec{t}$ is a forest; moreover $+$ is associative.
- If $\vec{s}$ is a forest, then $b\vec{s}$ is a tree (and also a forest) for every $b \in B$.

The above definition is peculiar in several respects. Note first that all forests are nonempty. Second, we use $+$ to denote the (non-commutative) concatenation of forests. We do this to be consistent with standard semigroup notation.

We denote trees by s, t and u. We denote forests by $\vec{s}$, $\vec{t}$ and sometimes $\vec{u}$.

It will be convenient to interpret a tree as a partial function $t : \mathbb{N}^* \to A \cup B$ with a finite domain. Elements of this finite domain are called *nodes* of t. This function assigns to each node its *label*. If x, y are two nodes of t, we write $x \leq y$ ($x < y$) if x is a (proper) prefix of y (i.e x is closer to the root). If x is a maximal node satisfying $x < y$, then we call x the *parent* of y and we call y a *successor* of x. (Each node has one parent, but may have many successors.) Two nodes are *siblings* if they have the same parent. A *leaf* is a node without successors. The *subtree of t rooted in the node x*, denoted $t|_x$, assigns the label $t(x \cdot y)$ to a node y. The *successor forest* of a node is the forest of subtrees rooted in that node's successors. We extend the notion of nodes to forests in a natural manner.

An (A, B)-*context* is an $(A \cup \{*\}, B)$-forest, where $*$ is a special symbol not in A. Moreover, $*$ occurs in exactly one leaf, which is called the *hole*. Moreover, we require the hole to have a parent (the hole cannot be a root in the forest). We use letters θ, η to denote contexts. A *tree* context is one with only a singe tree. When θ is a context and $\vec{t}$ is a forest, $\theta\vec{t}$ is the forest obtained from θ by replacing the hole with $\vec{t}$ (see Figure 1). Similarly we define the composition of two contexts θ, η – this is the context $\theta \cdot \eta$ that satisfies $(\theta \cdot \eta)\vec{t} = \theta(\eta\vec{t})$.



A context θ A forest $\vec{t}$ The resulting forest $\theta\vec{t}$

Figure 1: Application of context to a forest

1.2 The road not taken

For words, one can use either monoids or semigroups to recognize word languages. In the first case, the appropriate languages are of the form $L \subseteq A^*$, while the second case disallows the empty word, and only languages $L \subseteq A^+$ are considered.

For forests, the number of choices is much greater. Not only do we have two sorts (forests and contexts) instead of just one (words), but these sorts are also more complex. We would list here some of the choices that can be made:

- Is the empty forest a forest? Here, we say no.
- Is the empty context a context? Here, we say no.
- Even if the empty context is not allowed, can the hole be in the root (but with some nonempty trees as siblings)? Here, we say no.
- Can the hole have siblings? Here, we say yes.

Each combination of answers to the above questions gives rise to an appropriate definition of forest algebra, as long as the correct axioms are formulated. We do not lay any claim to the superiority of our choices. The others are just as viable, but we want to fix one set of definitions for this paper.

1.3 Forest algebras

In this section we formally define a forest algebra. We give some examples and explore basic properties.

A *forest prealgebra* (H, V, act) is a pair of semigroups along with an action $act : H \times V \to H$ of V on H. We require *faithfulness* here, i.e. that each element of V induces a different function on H, in other words: for every two distinct $v, w \in V$ there is some $h \in H$ such that $act(h, v) \neq act(h, w)$. In other words, a forest prealgebra is a transformation semigroup where the set acted upon is a semigroup. We call V the *vertical semigroup* and H the *horizontal semigroup*. We will denote the semigroup operation of V multiplicatively ($v \cdot w$ or even vw, for $v, w \in V$) and the semigroup operation of H additively ($g + h$ for $g, h \in H$). Instead of writing $act(v, h)$, we write vh. Recall that the definition of an action requires that:

$$(vw)(h) = v(w(h))$$

and hence it is unambiguous to write vwh. Most of the time we will omit the act coordinate from (H, V, act) and write (H, V), just as we identify a semigroup with its carrier set.

A *morphism* between two forest prealgebras (H, V) and (G, W) is a pair α of semigroup morphisms

$$\alpha_H : H \to G \qquad \alpha_V : V \to W$$

that is compatible with the action:

$$\alpha_V(v)\ \alpha_H(h) = \alpha_H(vh)\ .$$

To simplify notation, we will write $\alpha(h)$ instead of $\alpha_H(h)$ and $\alpha(v)$ instead of $\alpha_V(v)$. Hence, the above equation becomes $\alpha(v)\ \alpha(h) = \alpha(vh)$.

A *forest algebra* is a forest prealgebra (H, V) that satisfies the *insertion axioms*. These postulate that for each $v \in V$ and $h \in H$ one can find elements

$$act^l(v, h),\ act^r(v, h),\ act_l(v, h),\ act_r(v, h) \in V$$

whose actions on H are defined by:

$$\begin{aligned} act^l(v,h)\ g &= h + vg \\ act^r(v,h)\ g &= vg + h \\ act_l(v,h)\ g &= v(h+g) \\ act_r(v,h)\ g &= v(g+h) \end{aligned}$$

for each $g \in H$. Note that these elements are unique by the assumption on faithfulness. Note also that the insertion axioms are preserved under morphic images of forest prealgebras.

Example 1 Let H be any semigroup. Let V be the set H^H of all transformations of H into H, with composition as the operation. To obtain a forest algebra from (H, V) it suffices to add the action. The action of V on H is just function application. The insertion axioms are clearly satisfied.

Given two alphabets A, B, we define the *free forest algebra* over A, B, which is denoted by $(A, B)^\Delta$, to be:

- The horizontal semigroup is the set of forests over (A, B).
- The vertical semigroup is the set of contexts over (A, B)
- The action is the substitution of forests in contexts.

Recall that our definition of forests excludes the empty forest, and the hole cannot be a root in contexts. The following lemma shows that free forest algebra is indeed free in the sense of universal algebra.

Lemma 1 The free forest algebra $(A, B)^\Delta$ is a forest algebra. Moreover, for every forest algebra (H, V), any functions $f_A : A \to H, f_B : B \to V$ can be uniquely extended to a morphism $\alpha : (A, B)^\Delta \to (H, V)$

Proof
That $(A, B)^\Delta$ is a forest algebra can be easily verified.

We now proceed to the second part. Let $f_A : A \to H$ and $f_B : B \to V$ be the appropriate mappings. We first define the morphism over forests. The definition is by induction on the size of the forest:

$$\begin{aligned} \alpha(a) &= f_A(a) \\ \alpha(t_1 + \cdots + t_n) &= \alpha(t_1) + \cdots + \alpha(t_n) \\ \alpha(b(t_1 + \cdots + t_n)) &= f_B(b)\alpha(t_1 + \cdots + t_n)\ . \end{aligned}$$

We now proceed to define the morphism α for contexts. We start with contexts of the form $b(\vec{t_1} + * + \vec{t_2})$, i.e. a tree with root $b \in B$ which has a hole as one of its sons. We put:

$$\alpha(b(\vec{t_1} + * + \vec{t_2})) = act_r(act_l(f_B(b), \alpha(\vec{t_1})), \alpha(\vec{t_2}))\ .$$

For contexts where either $\vec{t}_1$ or $\vec{t}_2$ is empty we just omit the component that is empty in the definition above. Function α is extended to bigger contexts exactly in the same way as for forests. Directly from the definition it follows that α is a unique possible extension of f_A and f_B to a homomorphism. A relatively straightforward calculation shows that α is indeed a homomorphism. $\square$

Note 1 The above lemma shows why we consider trees with special labels for leaves and special labels for inner nodes: otherwise we wouldn't get the free forest algebra. Another solution would be to expand the signature with a special operation $forestize : V \to H$ which would make a forest out of a context (intuitively, by removing the hole). Under suitable axioms for the operation $forestize$, we would only need to supply generators for contexts. We chose however, to keep the forest algebra definition simple, and live with two types (A, B) of labels.

We now proceed to define languages recognized by forest algebras.

Definition 2 A set L of (A, B)-forests is said to be *forest-recognized* by a surjective morphism $\alpha : (A, B)^\Delta \to (H, V)$ if L is the inverse image $\alpha_1^{-1}(G)$ of some $G \subseteq H$. The morphism (α, β) is said to forest-recognize L, the set G is called the *accepting set* and L is said to be forest-recognized by (H, V).

Generally we are interested in the case when (H, V) is finite; in this case we say that L is *forest-recognizable*. We use the term forest-recognizable to avoid confusion with a different notion, called tree-recognizable, which will be introduced later on.

Example 2 Consider the set L of forests with an even number of nodes. We present here a finite forest algebra (H, V) forest-recognizing L. Both H and V are $\{0, 1\}$ with addition modulo 2. The action is also addition, the insertion axioms are clearly satisfied. The forest-recognizing morphism maps a context (resp. forest) onto 0 if it has an even number of nodes. The accepting set is $\{0\}$.

Example 3 A language L of (A, B)-forests is called *label-testable* if the membership $\vec{t} \in L$ depends only on the two sets of labels: the set of A-labels of leaves and the set of B-labels of internal nodes of $\vec{t}$. The appropriate forest algebra is defined as follows. Both H and V are the same semigroup: the set $P(A) \times P(B)$ with coordinate-wise union as the operation. The first coordinate keeps track of the labels in the leaves, while the second coordinate keeps track of the labels in the inner nodes. This determines the actions, which must also be coordinate-wise union.

1.4 Universal algebra viewpoint

Another way to look at a forest algebra is from the point of view of universal algebra. In this setting, a forest algebra is a two-sorted algebra (with the sorts being H and V) along with seven operations: (i) semigroup operations in H and V, (ii) the action vh of V on H and (iii) the four insertion axioms act^l, act^r, act_l, act_r, each one of the form $H \times V \to V$. Forest algebras are defined equationally by: (i) associativity for the semigroup operations in H and V, (ii) an equation saying that act is an action, (iii) equations for the insertion axioms. In this setting, a homomorphism is a morphism.

The universal algebra viewpoint gives us definitions of such concepts as subalgebra, cartesian product, free algebra, quotient, morphism. The only problem in this setting is the requirement on faithfulness which is not preserved by homomorphic images and quotients. This implies that every time we take a quotient we will be forced to check if the result is a faithful algebra.

1.5 Syntactic algebra for forest languages

Our aim now is to establish the concept of a syntactic forest algebra of a forest language. This is going to be a forest algebra that forest-recognizes the language, and one that is optimal among those that do. Here we will define "fat" syntactic algebra. Later we will define a "slim" version which will be adapted to recognize languages of trees.

Definition 3 We associate with a forest language L two equivalence relations on the free forest algebra $(A, B)^\Delta$:

- Two (A, B)-forests $\vec{s}$, $\vec{t}$ are *L-equivalent* if for every context θ, either both or none of the forest $\theta(\vec{s})$, $\theta(\vec{t})$ belong to L; and additionally we require that $\vec{t} \in L$ iff $\vec{s} \in L$.
- Two (A, B)-contexts θ, η are *L-equivalent* if for every nonempty forest $\vec{t}$, the forests $\theta(\vec{t})$ and $\eta(\vec{t})$ are L-equivalent.

Lemma 2 Both L-equivalence relations are congruences with respect to the operations of the forest algebra $(A, B)^\Delta$.

Proof
We first show that L-equivalence for forests is a congruence with respect to concatenation of forests. We start with concatenation to the right. We show that if $\vec{s}$ and $\vec{t}$ are L-equivalent, then so are the forests $\vec{s} + \vec{u}$ and $\vec{t} + \vec{u}$, for any forest $\vec{u}$. Unraveling the definition of L-equivalence, we must show that for any tree context θ we have: $\theta(\vec{s} + \vec{u}) \in L$ iff $\theta(\vec{t} + \vec{u}) \in L$. Let η be the context obtained from θ by putting $\vec{u}$ to the right of the hole; so $\theta(\vec{s} + \vec{u}) = \eta(\vec{s})$ and similarly for $\vec{t}$. We get:

$$\theta(\vec{s} + \vec{u}) \in L \iff \eta(\vec{s}) \in L \iff \eta(\vec{t}) \in L \iff \theta(\vec{t} + \vec{u}) \in L \ .$$

where the middle equivalence follows from L-equivalence of $\vec{s}$ and $\vec{t}$. The proof for the concatenation to the left is analogous.

We now proceed to show that L-equivalence for contexts is a congruence with respect to concatenation. We need to show that if two contexts θ and θ' are L-equivalent, then so are the contexts $\theta\eta$ and $\theta'\eta$ for any context η (and similarly for the concatenation to the left). We need to show that for every forest $\vec{t}$ and every context ζ,

$$\zeta(\theta\eta(\vec{t})) \in L \iff \zeta(\theta'\eta(\vec{t})) \in L \ .$$

The above equivalence follows immediately from the L-equivalence of θ and θ': it suffices to consider $\eta(\vec{t})$ as a tree that is plugged into the contexts θ and θ'.

Finally, we need to show that L-equivalence is a congruence with respect to the action $act(\theta, \vec{t}) = \theta(\vec{t})$. The proof is similar to the ones above. □

Definition 4 The *fat syntactic forest algebra* for L is the quotient of $(A, B)^\Delta$ with respect to L-equivalence, where the horizontal semigroup H^L consists of equivalence classes of forests over (A, B), while the vertical semigroup V^L consists of equivalence classes of contexts over (A, B). The *fat syntactic morphism* α^L assigns to every element of $(A, B)^\Delta$ its equivalence class in (H^L, V^L).

The above lemma guarantees that the quotient is well defined. This quotient is faithful (by definition of L-equivalence over contexts) and satisfies the insertion axioms (as a quotient of a forest algebra), hence it is a forest algebra. We claim that this forest algebra satisfies the properties required from the syntactic forest algebra of L.

Proposition 5 A language L of (A, B)-forest is recognized by a the fat syntactic morphism α^L. Moreover, any morphism $\beta : (A, B)^\Delta \to (H, V)$ that forest-recognizes L can be extended by a morphism $\gamma : (H, V) \to (H^L, V^L)$ so that $\gamma \circ \beta = \alpha^L$.

Proof
The first part follows immediately by taking as an accepting set the set of L-equivalence classes of all the elements of L. The second statement follows from the observation that if two (A, B)-forests or contexts have the same image under β then they are L-equivalent. □

Note that in the the syntactic forest algebra may be infinite. Clearly, if L is forest-recognized by some finite forest algebra, then its syntactic forest algebra must be finite by the above proposition.

1.6 Forest algebras and tree languages

Forest algebras give a natural definition of recognizable forest languages (Definition 2). This is all very fine, but the more studied object is tree languages. In this section we describe how a forest algebra can be used to recognize a language of unranked trees.

Definition 6 Given a tree language L over (A, B) and a letter $b \in B$, the *b-quotient*, denoted $b^{-1}L$, is the set of forests $\vec{t}$ that satisfy $b\vec{t} \in L$. A language L of (A, B)-trees is *tree-recognized* by a morphism $\alpha : (A, B)^\Delta \to (H, V)$ if $b^{-1}L$ is forest-recognized by α for all $b \in B$.

Note that the above definition does not say anything about trees with only one (root-leaf) node; but these are finitely many and irrelevant most of the time. In particular, regular languages are closed under adding or removing a finite number of trees.

Example 4 A tree language of the form: "the root label is $b \in B$" is tree-recognized by any forest algebra. This because all the quotients $c^{-1}L$ for $c \in B$ are either empty (when $c \neq b$) or contain all forests (when $c = b$).

The above definition of recognizability induces a definition of syntactic forest algebra for a tree language L. Consider the intersection of all $(b^{-1}L)$-equivalences for $b \in B$. This is a congruence on $(A, B)^\Delta$ as it is an intersection of congruences. Call this congruence *tree-L-equivalence*.

Definition 7 The *slim syntactic forest algebra* for a tree language L, denoted (H_L, V_L), is the quotient of $(A, B)^\Delta$ with respect to tree-L-equivalence. The syntactic morphism $\alpha_L : (A, B)^\Delta \to (H_L, V_L)$ assigns to every element of $(A, B)^\Delta$ its equivalence class in (H_L, V_L).

We will often omit qualifier slim and just say syntactic forest algebra. By contrast we will always refer to fat syntactic forest algebras by their full name. An immediate corollary of Proposition 5 is the similar result for tree-recognition.

Proposition 8 A language of (A, B)-trees is recognized by the slim syntactic morphism α_L. Moreover, any morphism $\beta : (A, B)^\Delta \to (H, V)$ that tree-recognizes L can be extended by a morphism $\gamma : (H, V) \to (H^L, V^L)$ so that $\gamma \circ \beta = \alpha^L$.

Note 2 There is an alternative definition of tree-recognizability. In the alternative definition, we say that a tree language L is tree-recognized by a forest algebra (H, V) if there is a forest language K forest-recognized by (H, V) such that L is the intersection of K with the set of trees. Under this alternative definition, there is no correct notion of syntactic algebra. For instance, the tree language "trees whose root label is b" can be tree-recognized by two forest algebras that have no common quotient tree-recognizing this language. Indeed, these may be forest algebras for two different forest languages that agree on trees.

Note 3 Yet another alternative definition of tree-recognizability says that L is tree-recognized iff it is forest-recognized. In this case, the forest-algebra must keep track of what is a single tree, and what is a forest. This leads to some pollution in the syntactic forest algebra. For instance, the syntactic algebra of the language "there is some leaf labeled by a" does not satisfy $h + h = h$.

1.7 Automata over trees

We would like to show that our definition of recognizability is equivalent with the standard notion of regular tree languages. There are numerous presentations of automata on finite unranked trees; here we will use one that matches well our algebraic definitions.

A tree automaton over the pair of alphabets (A, B) is a tuple

$$\mathcal{A} = \langle (Q, \cdot),\ A,\ B,\ \delta : (A \to Q) \times (B \times Q \to Q),\ F \subseteq Q \rangle$$

where $(Q, \cdot)$ is a semigroup; intuitively a set of states with a semigroup structure.

The automaton assigns to every tree t a value $t^{\mathcal{A}} \in Q$; that is defined by induction as follows:

- If t consists of a single leaf labelled a then $t^{\mathcal{A}} = \delta(a)$;
- otherwise $t = b(s_1 + \cdots + s_n)$, and we put $t^{\mathcal{A}} = \delta(b, s_1^{\mathcal{A}} \cdot s_2^{\mathcal{A}} \cdots s_n^{\mathcal{A}})$; observe that the multiplication is done in $(Q, \cdot)$.

A tree t is *accepted by* $\mathcal{A}$ if $t^{\mathcal{A}} \in F$.

Actually, the above automata can also accept forest languages. Indeed one can define $\vec{t}^{\mathcal{A}}$ for a forest $\vec{t} = t_1 + \cdots + t_n$ as $t_1^{\mathcal{A}} \cdots t_n^{\mathcal{A}}$; where the multiplication is done in $(Q, \cdot)$.

Proposition 9 A tree language is tree-recognized by a finite forest algebra if an only if it is the language of trees accepted by some tree automaton.

Proof
Take a tree language L tree-recognized by a morphism $\alpha : (A, B)^{\Delta} \to (H, V)$. We show how it can be recognized by an automaton. For each $b \in B$ take the automaton $\mathcal{A}_b = \langle H, A, B, \delta, F_b \rangle$ where H is the horizontal semigroup, $F_b \subseteq H$ is the accepting set for $b^{-1}L$, and δ is defined as follows:

$$\delta(a) = \alpha(a) \qquad \delta(b, h) = act(\alpha(b), h)\ .$$

By induction on the size of the forest one can show that $\vec{t}^{\mathcal{A}} = \alpha(\vec{t})$. Thus $\mathcal{A}_b$ recognizes the language of forests $b^{-1}L$. Combining all $\mathcal{A}_b$ together it is not difficult to construct an automaton recognizing L.

For the proof in the other direction suppose that we are given an automaton $\mathcal{A} = \langle (Q, \cdot), A, B, \delta, F \rangle$. We consider a tree algebra (H, V) where H is $(Q, \cdot)$ and V is the function space $H \to H$ with the function composition as the operation; the action is the function application. It is easy to see that (H, V) is a forest algebra. Consider the unique homomorphism $\alpha : (A, B)^{\Delta} \to (H, V)$ such that:

$$\alpha(a) = \delta(a) \qquad \alpha(b) = \delta(b)\ ;$$

observe that $\delta(b)$ is a function from H to H. By induction on the height of the forest one can show that $\vec{t}^{\mathcal{A}} = \alpha(\vec{t})$. In order to recognize $b^{-1}L$ we take all the

set F_b of all q such that $\delta(b, q) \in F$. We have that $b^{-1}L = \alpha^{-1}(F_b)$, i.e, $b^{-1}L$ is recognized by α with F_b as an accepting set. □

Actually, the above notion of automaton can be refined to a notion of (H, V) automaton for any forest algebra (H, V). Such an automaton has a form:

$$\mathcal{A} = \langle H,\ A,\ B,\ \delta : (A \to H) \times (B \to V),\ F \subseteq Q \rangle$$

thus the only change is that now states are from H and $\delta(b)$ is an element of from V and not a function from $Q \to Q$. We can do this because using the action *act* of the forest algebra, each $v \in V$ defines a function $act(v) : Q \to Q$.

It is easy to see that every language accepted by a (H, V) automaton is recognized by (H, V) and vice versa: for every language recognized by (H, V) there is an automaton accepting it. This equivalence shows the essential difference between algebras and automata. Algebras do not depend on alphabets, while alphabets are explicitly stated in the description of an automaton. More importantly, the structure of vertical semigroup is not visible in automaton: in automaton we see generators of the vertical semigroup.

One may ask, what would happen if we remove explicit mention of alphabets in automata. In this case we obtain a pair (H, Z) where Z is just a set and not a semigroup, but still be have an action of Z on H. For such objects we do not need to require insertion axioms as these axioms talk about the structure of the vertical semigroup which is not present here. All the theory could be developed in this setting but we refrain from doing this because we think that the structure of the vertical semigroup is important.

2 EF

In this section we show how forest algebras can be used to give a decidable characterization of a known temporal logic for trees.

2.1 The logic EF

EF is a temporal logic that expresses properties of trees. The name EF is due to the only temporal operator in the logic — EF — which stands for Exists (some path) Finally (on this path). Formulas of EF are defined as follows:

- If a is a letter, then a is a formula true in trees whose root label is a.
- EF formulas are closed under boolean connectives.
- If φ is an EF formula, then $\mathsf{EF}\varphi$ is an EF formula true in trees with a *proper* subtree satisfying φ.

Restricting to proper subtrees in the definition of EF gives us more power, since the nonproper operator can be defined as $\varphi \vee \mathsf{EF}\varphi$. To be consistent with the forest algebra definitions, we stay with the distinction between labels A for

leaves and labels B for inner nodes. This is not really important for EF, since one can easily verify if a node is an inner node ($\mathsf{EF}true$).

In this section, we present three equations and show that a language can be definable by an EF formula if and only if its syntactic forest algebra satisfies these equations. In particular, it is decidable if a regular tree language can be defined in EF.

Theorem 10
A tree language is definable in EF *if and only if its slim syntactic forest algebra satisfies the following equations, called the* EF *equations:*

$$\begin{aligned} vh &= h + vh \\ g + h &= h + g \\ h + h &= h \ . \end{aligned}$$

The proof of this theorem is split across the following two subsections.

2.2 Correctness

In this subsection we show that each tree language definable in EF must satisfy the EF equations.

Assume then that a tree language L is defined by an EF formula φ. Given a forest $t_1 + \cdots + t_n$, let $I_\varphi(t_1 + \cdots + t_n)$ be the set of (not necessarily proper) subformulas of φ that are satisfied in some tree t_i. The following lemma follows by an easy induction on the size of φ.

Lemma 3 Two forests have the same values under I_φ are L-equivalent.

Corollary 11 The last three EF equations are satisfied.

Proof
Using Lemma 3, we can define a forest algebra (H, V) where H is the set of possible values of I_φ and V is a set of functions $H -> H$ that are "induced" by contexts. This way (H, V) will satisfy the equations. Using the last lemma it is easy to recognize L with this algebra. Since the syntactic forest algebra of L is a morphic image of (H, V), then it must also satisfy the last three equations. □

2.3 Completeness

In this section we show that if a forest algebra satisfies the three EF equations, then any tree language tree-recognized by this algebra can be defined in EF. This is gives the desired result, since the syntactic algebra of L tree-recognizes L.

From now on we fix a forest algebra (H, V) that tree-recognizes a tree language L via a morphism

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

We assume that the forest algebra (H, V) satisfies the three EF equations. We will show that L can be defined using an EF formula.

We first show that the EF equations imply a fourth one:

$$w(vw)^\omega = (vw)^\omega \ .$$

(This is L-triviality of the context semigroup). We need to explain the ω notation, though. In each finite semigroup S, there is a power $n \in \mathbb{N}$ such that all elements $s \in S$ satisfy $s^n = s^n s^n$. We refer to this power as ω, and use it in equations. In particular, every finite semigroup satisfies the equation: $s^\omega = s^\omega s^\omega$. The reader is advised to substitute "a very large power" for the term ω when reading the equations.

Lemma 4 For each $v, w \in V$, we have $w(vw)^\omega = (vw)^\omega$

Proof
First we show that the EF equations imply aperiodicity for the context semigroup:

$$v^\omega = vv^\omega \ .$$

Indeed, by applying the first equation repeatedly to $v^\omega v^\omega$, we obtain:

$$v^\omega = v^\omega v^\omega = v^\omega + vv^\omega + vvv^\omega + \cdots + v^\omega v^\omega$$

Likewise for $vv^\omega v^\omega$:

$$vv^\omega = vv^\omega v^\omega = vv^\omega + vvv^\omega + vvvv^\omega + \cdots + v^\omega v^\omega + vv^\omega v^\omega$$

If we cancel out $vv^\omega v^\omega = vv^\omega$, and use idempotency and commutativity, we obtain the desired equality $v^\omega = vv^\omega$.

We now proceed to show the statement of the lemma.

$$w(vw)^\omega = (vw)^\omega + w(vw)^\omega = vw(vw)^\omega + w(vw)^\omega = vw(vw)^\omega = (vw)^\omega \ .$$

In the first and third equation we use $vh = h + vh$, while in the second and fourth we use aperiodicity. □

Note 4 The reader may be alarmed by this lemma. For words, the analogous equation would be $u(vw)^\omega = uw(vw)^\omega$; that is, the left and right side are equal in any nonempty context u. The reason why we can loose the nonempty context u in trees and still have a valid equation is the peculiarity of our definition of tree-recognizability: the syntactic forest algebra of a tree language can only distinguish forests by placing them in nonempty contexts.

The main idea of the proof is to do an induction with respect to what forests can be found inside other forests. Given $h, g \in H$, we write $h \leq g$ if $h = g$ or there is some context $u \in V$ such that $g = uh$. We write $h \sim g$ if $\leq$ holds both ways.

Lemma 5 If $g \leq h$ then $g + h = h$.

Proof
If $g = h$ then we use idempotency. If $g \neq h$ we use the second EF equation. $\square$

Lemma 6 If $g \sim h$ then $g = h$. In particular, $\leq$ is a partial order.

Proof
Assume that $g \neq h$. If $g \sim h$ then there are contexts v, w such that $h = wg$ and $g = vh$. Iterating this process ω-times we obtain

$$h = wvh = (wv)^\omega h$$

But then, by applying Lemma 4, we get

$$h = (wv)^\omega h = w(vw)^\omega h = g \ .$$

$\square$

The *typeset* of a forest $t_1 + \cdots + t_n$ is defined to be the set of values $\alpha(s)$ for (not necessarily proper) subtrees of $t_1, \ldots, t_n$.

Lemma 7 If the typesets of forests $\vec{t}, \vec{s}$ have the same maximal elements (with respect to $\leq$) then $\alpha(\vec{t}) = \alpha(\vec{s})$.

Proof
Let $\vec{t} = t_1 + \ldots + t_n$. Note that each maximal element in the typeset of t is represented by one of the trees t_i, since bigger trees have bigger values $\alpha(s)$. If for some i, j we have $\alpha(t_i) \leq \alpha(t_j)$, we can use commutativity and Lemma 5 to remove t_j. This way we make sure that only maximal values are represented in $t_1, \ldots, t_m$. Then using equations 3 and 4 we can make sure that each element is represented exactly once. If we apply the same transformation to $\vec{s}$, we get the desired result. $\square$

The following proposition is the main induction in the completeness proof:

Proposition 12 For each $h, g \in H$, there are EF formulas $\varphi_g^{<h}$ and φ_h where for each tree t, we have

- $t \models \varphi_g^{<h}$ iff $\alpha(t) = g$ and all proper subtrees s of t satisfy $\alpha(s) < h$.
- $t \models \varphi_h$ iff $\alpha(t) = h$.

Before we proceed with the proof of this proposition, we show how it concludes the completeness claim. Indeed, once we have a formula φ_h for each $h \in H$, we can easily write an EF formula that describes the language L. For trees with a single node, we write separate formulas (EF can detect that a tree has only one node). If a tree is of the form $b\vec{t}$, then we need to verify if $\vec{t}$ belongs to the quotient $b^{-1}L$. By assumption on tree-recognizability, this quotient is

recognized by the morphism α. By Lemma 7, the value $\alpha(\vec{t})$ depends only on the typeset of $\vec{t}$. Since the modality EF looks at proper subtrees, we can easily describe the typeset of the successor forest $\vec{t}$ with an EF formula.

Proof (of Proposition 12)
The proof is by induction on the depth of h in the order $\leq$, i.e. number of f satisfying $f < h$.

Consider first the base case, when h is minimal for $\leq$. In this case the formulas $\varphi_g^{<h}$ are all vacuous, the only interesting one is φ_h. How can a tree t satisfy $\alpha(t) = h$? First of all, all leaves need to have labels $a \in A$ satisfying $\alpha(a) = h$; this can be easily tested in EF. Second, all internal nodes need to have labels $b \in B$ satisfying $\alpha(b)h = h$; this can also be tested in EF. These conditions are clearly necessary, but thanks to idempotency $h + h = h$, they are also sufficient.

We now proceed with the induction step. We take some $h \in H$ and assume that the proposition is true for all $f < h$. The formula $\varphi_g^{<h}$ is constructed by using Lemma 7. This formula is a disjunction of statements of the form

> The typeset of the successor forest is G and the root label is b

over all possible $b \in B$ and $G \subseteq \{f : f < h\}$ such $\alpha(b\vec{t}) = g$ holds for some forest $\bar{t}$ whose typeset is G. Since only $f < h$ are involved in G, saying that the typeset is G can be done thanks to the induction assumption.

We now proceed to define the formula φ_h. The general idea is that h must appear somewhere for the first time (which can be detected by $\varphi_h^{<h}$), and then it must be preserved up until the root. We define φ_h as the conjunction of the following three properties:

1. At least one (not necessarily proper) subtree satisfies $\varphi_h^{<h}$.
2. Every node with a proper subtree satisfying $\varphi_h^{<h}$ has a label $b \in B$ that satisfies $\alpha(b)h = h$.
3. For each $f > h$, no (not necessarily proper) subtree satisfies $\varphi_f^{<h}$.

We now prove a tree t satisfies the above three properties if and only if $\alpha(t) = h$. We prove both implications separately:

- Left to right. If t satisfies the first property, than $\alpha(t) \geq h$. Assume then that $\alpha(t) > h$. We will show that a contradiction follows. Let

  $$s = b(s_1 + \ldots + s_n)$$

  be a minimal subtree of t with $\alpha(s) > h$. By minimality of s, we have $\alpha(s_i) \leq h$ for all $i = 1, \ldots, n$. If all s_i satisfy $\alpha(s_i) < h$, then $\varphi_{\alpha(s)}^{<h}$ must hold in s, a contradiction with the third conjunct of φ_h. Otherwise some s_i satisfies $\alpha(s_i) = h$. But then $\alpha(s_1 + \cdots + s_i) = \alpha(s_i)$ holds by repeated application of Lemma 5. In particular we obtain a contradiction with the second conjunct of φ_h.

- Right to left. The first and third conjuncts are obvious. The second follows from a reasoning similar to the one above.

□

3 Wreath products of forest algebras

The goal of this section is to apply wreath products to forest algebras. We show that wreath product corresponds to composition of formulas over trees. Theorem 16 is similar to results of Zoltan Ésik and Szabolcs Iván [8, 7, 9, 10], which consider ranked trees and cascade products.

3.1 Wreath product

In this section we define the wreath product of two tree algebras. We use the standard definition of wreath product of transformation semigroups and apply it to the special case of tree algebras.

Definition 13 (Wreath Product) Let (H, V) and (G, W) be two forest algebras. The *wreath product* $(G, W) \circ (H, V)$ is the pair (I, U) defined as follows. The horizontal semigroup I is the product semigroup $H \times G$. The vertical semigroup U is $V^G \times W$, with multiplication defined:

$$(\psi, w) \cdot_U (\psi', w') = (\psi'', w \cdot_W w') \quad \text{where } \psi''(g) = \psi(w(g)) \cdot_V \psi'(g)$$

The action of U on I is defined as follows:

$$\begin{aligned}(\psi, w)(h, g) =& (\psi(g)h, wg) \\ & \text{for } (\psi, w) \in V^G \times W \ , (h, g) \in H \times G \ .\end{aligned}$$

Note that it is not yet clear if (I, U) is a forest algebra; it may not satisfy some of the requirements (such as faithfulness and the insertion axioms). We will show this in Lemma 8.

The definition above is the standard definition of wreath product of two transformation semigroups. We will try to give the reader some intuition about this construction. The idea is that there are two layers of the forest algebra: one – (G, W) – works on the tree first, and then lets the second layer — (H, V) — read the output of the first. In the vertical semigroup of the wreath product, this is encoded on the two coordinates. The forest algebra (I, U) works "business as usual" on the second coordinate (this choice of coordinates is traditional). On the first coordinate, there is a function ψ that waits for the result g of the first layer, and after receiving this result returns the appropriate "second level" vertical transformation $\psi(g) \in V$. We will expand on this intuition in the next section.

Lemma 8 The wreath product of two forest algebras is a forest algebra.

Proof
It is known that the wreath product of two transition semigroups is a transition semigroup. Therefore, the wreath product of two forest algebras is a transition semigroup. It is also obvious that the first coordinate of the wreath product, $H \times G$ is a semigroup. It remains to verify the insertion axioms.

Let then (ψ, w) be an element of the vertical semigroup $U = V^G \times W$ and let (h, g) be an element of the horizontal semigroup $H \times G$. We only verify one of the insertion axioms: act^l; the other three are done symmetrically. We need to show that there is an element

$$act^l((\psi, w), (h, g)) \in U = V^G \times W$$

whose action on $I = H \times G$ is defined by:

$$act^l((\psi, w), (h, g))(h', g') = (\psi, w)(h, g) + (h', g') \ . \tag{1}$$

We define

$$act^l((\psi, w), (h, g)) = (\varphi, act^l(w, g)) \ ,$$

where $\varphi : G \to V$ is the mapping

$$g' \mapsto act^l(\psi(g'), h) \ .$$

In the above definitions, $act^l(w, g)$ is obtained from the insertion axioms in (G, W) and $act^l(\psi(g'), h)$ is obtained from the insertion axioms in (H, V).

We now verify that (1) is indeed satisfied. If we unravel the definition of the left-hand side, we obtain:

$$(\varphi(g')h', act^l(w, g)g') \ .$$

Using the insertion axiom in (G, W) on the second coordinate, this becomes

$$(\varphi(g')h', g' + wg) \ .$$

Unraveling the definition of φ, this becomes

$$(act^l(\psi(g'), h)h', g + wg') \ .$$

Using the insertion axiom in (H, V) on the first coordinate, this becomes

$$(h + \psi(g')h', g + wg') = (h, g) + (\psi(g')h', wg') = (h, g) + (\psi, w)(h', g') \ ,$$

which completes the proof of (1). □

3.2 Evaluating the wreath product

In this section we show how to decompose a morphism:

$$\alpha : (A, B)^{\Delta} \to (H, V) \circ (G, W)$$

in terms of a relabeling and morphisms to the components. This will allow us later to show the connections between wreath product and formula composition on one side and cascade product of automata on the other side.

Recall that the carrier of $(H, V) \circ (G, W)$ is $(H \times G, V^G \times W)$. Thus the morphism α can be decomposed into two functions:

$$f : (A, B)^{\Delta} \to (H, V^G)$$
$$\gamma : (A, B)^{\Delta} \to (G, W)$$

with the property that

$$\text{for every forest } \vec{t}\text{:} \quad \alpha(\vec{t}) = (f(\vec{t}), \gamma(\vec{t})) \in H \times G \ ; \tag{2}$$

and similarly for contexts. To read the equation as above, recall our convention of using the same symbol for two components of a morphism: so $f(\vec{t}) \in H$ if $\vec{t}$ is a forest, but $f(v) \in V^G$ if v is a context. From the definition of the wreath product it follows that γ is a morphism, but f need not be one. We can at least show that f preserves horizontal composition, i.e. $f(t_1 + \cdots + t_n) = f(t_1) + \cdots + f(t_n)$. Indeed:

$$(h, g) = \alpha(t_1 + \cdots + t_n) = \alpha(t_1) + \cdots + \alpha(t_n) = \\ (h_1, g_1) + \cdots + (h_n, g_n) = (h_1 + \cdots + h_n,\ g_1 + \cdots + g_n)$$

which allows us to conclude as $h = f(t_1 + \cdots + t_n)$ and $h_i = f(t_i)$ for $i = 1, \ldots, n$.

We are now going to show how to describe f in terms of morphisms and relabellings. Consider a morphism $\beta : (A, B \times G)^{\Delta} \to (H, V)$ defined by

$$\beta(a) = f(a) \qquad \beta(b, g) = f(b)(g) \tag{3}$$

In order to apply β we need a tree over the alphabet $(A, B \times G)$. This can be produced using the morphism γ. Given a tree t over (A, B), let $lab_{\gamma}(t)$ be a tree over $(A, B \times G)$ whose every internal node v is labeled by $(b, \gamma(t\downarrow_v))$; where b is the label in the node v and $t\downarrow_v$ is the forest of trees rooted in sons of v.

The following proposition describes α in terms of γ, β and the relabelling.

Lemma 9 For every forest $\vec{t}$ over the alphabet (A, B):

$$\alpha(\vec{t}) = (\beta(lab_{\gamma}(\vec{t})), \gamma(\vec{t})).$$

Proof
From the equation (2) we have equality for the second component. Hence it remains to show that $f(\vec{t}) = \beta(lab_{\gamma}(\vec{t}))$. The proof proceeds by induction on

the size of $\vec{t}$. If $\vec{t}$ consists of one leaf, the equality follows immediately from the definitions. If $\vec{t}$ is a forest for a form $t_1 + \cdots + t_n$ then the equation follows from the preservation of the horizontal composition by β, lab_γ and f.

The last case is when we have a tree $t = b(\vec{s})$. Denote $g = \gamma(\vec{s})$. We have:

$$\begin{aligned}\beta(lab_\gamma(t)) &= \beta(lab_\gamma(b(\vec{s}))) \\ &= \beta(b,g)(\beta(lab_\gamma(\vec{s}))) \\ &= (f(b)(g))(\beta(lab_\gamma(\vec{s}))) \\ &= (f(b)(g))(f(\vec{s}))\end{aligned}$$

where the second equation follows from the definition of the labeling, the third from the definition of β and the forth form the induction hypothesis. Let us now examine the other hand side

$$\begin{aligned}\alpha(b(\vec{s})) &= \alpha(b)\alpha(\vec{s}) \\ &= \big(f(b),\gamma(b))(f(\vec{s}),\gamma(\vec{s})\big) = \big((f(b)(g))f(\vec{s}),\ \gamma(b)\gamma(\vec{s})\big)\end{aligned}$$

The second equation follows from the definition of f and γ and the third from the definition of action in the wreath product. We are done as the first component of the right-hand side is equal to $\beta(lab_\gamma(t))$. □

4 Wreath product and formula composition

In this section we show that wreath product is the same as composition of formulas. We want to have as general as possible a concept of formulas, so we simply use languages. Boolean operations apply just as well to languages as to formulas, so this is not a problem. However, we need a concept of composition applied to languages.

4.1 An abstract view of formula composition

Let $L_1, \ldots, L_n$ be languages over (A, C) and let t be a tree over (A, C). We

$$t[b_1 := L_1, \ldots, b_n := L_n]$$

to be the fores over $(A, \{b_1, \ldots, b_n\})$ obtained from t by relabeling a node v with the label b_i if the subtree $t|_v$ belongs to L_i. Since the L_i may not necessarily be pairwise disjoint, we use the label corresponding to the first L_i that is true. If none of the languages apply, we use the last label b_n (and therefore the actual language L_n is irrelevant).

This operation can be extended to languages. If L is a language over $(A, \{b_1, \ldots, b_n\})$ and $L_1, \ldots, L_n$ are as above, then the *language composition* is the following set of trees over (A, C):

$$L[b_1 := L_1, \ldots, b_n := L_n] = \{t : t[b_1 := L_1, \ldots, b_n := L_n] \in L\} \ .$$

Example 5 Let L_1 be the following language over $(\{a\}, \{b, c\})$: "some inner node has label b and the root label is c". Let L_2 be the complement of L_1. Let t be some tree over $(\{a\}, \{b, c\})$. In the tree

$$t[b_1 := L_1, b_2 := L_2] \ ,$$

an inner node x is labeled by b_1 if and only if in t, the node x had label c and a descendant $y \geq x$ with label b. (Note that the language L_2 is irrelevant, so we could have taken $L_2 = \emptyset$). In particular, if we set L to be the language: "some inner node has label b_1", then the composition

$$L[b_1 := L_1, b_2 := L_2]$$

defines the same language as the EF formula $\mathsf{EF}(c \wedge \mathsf{EF}b)$.

Definition 14 Given a class of languages $\mathbb{L}$, we denote by $\langle \mathbb{L} \rangle$ the smallest class of languages that contains $\mathbb{L}$ and is closed under boolean operations and language composition. In this case, $\mathbb{L}$ is called a *base* of the language class $\langle \mathbb{L} \rangle$. (There may be several bases.)

Definition 15 Given a class of forest algebras $\mathbb{V}$, we denote by $\langle \mathbb{V} \rangle$ the smallest class of forest algebras that contains $\mathbb{V}$ and is closed under wreath product.

Note 5 In the above definition of $\langle \mathbb{V} \rangle$, one might expect a requirement for closure under cartesian products, to match closure of $\langle \mathbb{L} \rangle$ under boolean operations. Later on we will see that this is redundant, since the wreath product can simulate cartesian product.

Example 6 Let $\mathbb{L}$ be the class of languages of the form

"some non-root node has label a" "the root label is a"

for all alphabets (A, B) and all $a \in A \cup B$. In this case, $\langle \mathbb{L} \rangle$ is exactly the class of languages definable in EF.

Note 6 We would like to note here that our abstract definition of languages as formulas is restrictive in several ways. A language can be seen as a type of formula $\varphi(x)$ that takes a single variable x (a node) and verifies if the subtree of this node x satisfies the formula φ. In particular, there is no clear way of modeling multiple free variables in our approach. Moreover, the value of $\varphi(x)$ depends only on the descendants of x, and not on its ancestors.

4.2 Wreath product is composition

The following theorem shows that wreath product is the same as language composition. It is formulated in terms of tree-recognition. Let us remark that anologous result for frest recognition would require some additional hypothesis on $\mathbb{L}$.

Theorem 16
Let $\mathbb{L}$ be a class of languages, and let $\mathbb{V}$ be a class of forest algebras. If $\mathbb{L}$ is exactly the class of languages tree-recognized by $\mathbb{V}$, then $\langle\mathbb{L}\rangle$ is exactly the class of languages tree-recognized by $\langle\mathbb{V}\rangle$.

We show the two inclusions in the following Lemmas 10 and 11.

Lemma 10 If every language tree recognizable by $\mathbb{V}$ belongs to $\mathbb{L}$, then every language tree recognizable by $\langle\mathbb{V}\rangle$ is belongs to $\langle\mathbb{L}\rangle$.

Proof
Take $\alpha : (A,B)^\Delta \to (I,U)$, with $(I,U) \in \langle\mathbb{V}\rangle$. We need to show that any language tree recognized by α belongs to $\langle\mathbb{L}\rangle$.

If (I,U) belongs to $\mathbb{V}$ then we are done. The other possible case is that $(I,U) = (H,V) \circ (G,W)$ and that the lemma is true for the two latter algebras. By definition of recognition, for each $b \in B$ there is $F_b \subseteq I$ with $b^{-1}L = \alpha^{-1}(F_b)$. We will produce for each $(h,g) \in H \times G$ a language $L_{(h,g)} \in \langle\mathbb{L}\rangle$ containing precisely trees $b(\vec{t})$ with $\alpha(\vec{t}) = (h,g)$; i.e., trees such that the forest $\vec{t}$ of descendants of the root evaluates to (h,g). Having this we will be able to conclude by taking the union of all languages of the form $L_b \wedge L_{(h,g)}$ that are included in L; where L_b denotes the set of trees with the root labelled by b. Observe that L_b is in $\langle\mathbb{L}\rangle$ as it is tree-recognizable by any forest algebra (cf. Example 4).

Let us decompose α into $f(A,B)^\Delta \to (H,V^G)$ and $\gamma : (A,B)^\Delta \to (G,W)$ as it is was done in equation (2). We will use the characterisation given by Lemma 9 which says that $\alpha(t) = (\beta(lab_\gamma(t)), \gamma(t)) \in H \times G$; where β is defined by equation (3) and lab_γ is as described just below this equation.

First, as γ is a morphism, by induction assumption for each $g \in G$ there is a language $L_g \in \langle\mathbb{L}\rangle$ consisting of those trees $b(t)$ for which $\gamma(t) = g$. We can also use the induction assumption for $\beta : (A, B \times G)^\Delta \to (H,V)$. This gives a for every $h \in H$ a language L_h of trees $b(t)$ where $\beta(t) = h$.

We claim that the desired language $L_{(h,g)}$ is $L'_h \cap L_g$ where

$$L'_h = L_h[(b_1,g_1) := L_{b_1} \cap L_{g_1}, \ldots, (b_k,g_l) := L_{b_k} \cap L_{g_l}] \ .$$

Indeed, for a tree t over (A,B), the label of a node x of $lab_\gamma(t)$ is (b,g') iff x is labelled by b and the forest of the trees below x satisfies g'. So t is in L'_h iff $lab_\gamma(t)$ is in L_h. □

Lemma 11 If every language in $\mathbb{L}$ is tree-recognized by $\mathbb{V}$, then every language in $\langle\mathbb{L}\rangle$ is tree-recognized by $\langle\mathbb{V}\rangle$.

Proof
The proof is by induction on the number of times boolean operations and language compositions are applied. The induction base follows immediately from the assumptions on $\mathbb{L}$ and $\mathbb{V}$.

The part for boolean operations follows from the following simple claim:

Claim 1 Any language tree-recognized by $(H, V) \times (G, W)$ is tree-recognized by $(H, V) \circ (G, W)$.

Proof
Let L be a language recognized by a morphism

$$\alpha : (A, B)^{\Delta} \to (H, V) \times (G, W) \ .$$

We need to define a recognizing morphism

$$\beta : (A, B)^{\Delta} \to (H, V) \circ (G, W) \ .$$

Over forests, we can make α and β identical, since the horizontal components in both forest algebras are the same. Let then θ be a context, with $\alpha(\theta) = (v, w)$. We define

$$\beta(\theta) = (g \mapsto v, w) \ ,$$

where the first coordinate is the constant function. One can easily verify that β is indeed a morphism. □

Let then $L_1, \ldots, L_n \in \langle \mathbb{L} \rangle$ be languages over (A, C) and let $L \in \langle \mathbb{L} \rangle$ be a language over $(A, \{b_1, \ldots, b_n\})$. We need to show that the language

$$K := L[b_1 := L_1, \ldots, b_n := L_n]$$

is tree-recognized by a forest algebra in $\langle \mathbb{V} \rangle$.

By induction assumption we have forest algebras

$$(H, V), (H_1, V_1), \ldots, (H_n, V_n) \in \langle \mathbb{V} \rangle$$

and morphisms

$$\alpha : (A, \{b_1, \ldots, b_n\})^{\Delta} \to (H, V) \qquad \alpha_i : (A, C)^{\Delta} \to (H_i, V_i)$$

that tree-recognize L and the languages L_i. We need to find a forest algebra in $\langle \mathbb{V} \rangle$ that tree-recognizes the language composition.

In the first step we compose all the (H_i, V_i) into one forest algebra. This is because the cartesian product

$$(H_1, V_1) \times \cdots \times (H_n, V_n)$$

simultaneously tree-recognizes all the L_i. Therefore we can use the claim to assume without loss of generality that all the languages L_i are recognized by the same morphism

$$\beta : (A, C)^{\Delta} \to (G, W) \in \langle \mathbb{V} \rangle \ .$$

(Although the morphism is the same for all languages $L_1, \ldots, L_n$, the accepting sets are going to be different.)

We now claim that the wreath product $(H, V) \circ (G, W)$ tree-recognizes the language composition. We need to define the recognizing morphism

$$\gamma : (A, C)^{\Delta} \to (H, V) \circ (G, W)$$

and the accepting sets. The morphism will satisfy the following property:

$$\gamma(t) = (\alpha(t[b_1 := L_1, \ldots, b_n := L_n]), \beta(t)) \tag{4}$$

We define the morphism only for the generators $A \cup B$. Recall that the wreath product is of the form

$$(H \times G, V^G \times W)$$

For the leaf generators $a \in A$, the goal (4) leaves us no choice:

$$\gamma(a) = (\alpha(a), \beta(a)) \ .$$

For an inner node generator $c \in C$, we need to provide a pair

$$\psi : G \to V \quad , \quad w \in W \ .$$

Over the second coordinate, we have no choice: w must be $\beta(c)$. This ensures that (4) holds over the second coordinate. We now proceed to define the result of ψ over an element $g \in G$. Since all the languages $L_1, \ldots, L_n$ are tree-recognized by β, then the label c along with a value $g \in G$ uniquely determines which of the languages L_i contain a tree $c\vec{s}$ satisfying $\beta(\vec{s}) = g$. Therefore, the label c and the value $g \in G$ uniquely determine the root label of a tree

$$t[b_1 := L_1, \ldots, b_n := L_n]$$

where the root label of t is c and the successor forest of t evaluates to g under β. If $b_{c,g} \in \{b_1, \ldots, b_n\}$ is this uniquely determined label (depending on c and g), then the function ψ assigns to $g \in G$ the value $\alpha(b_{c,g}) \in V$.

Using the values defined above, one can verify by induction on the size of the tree t that the property (4) is satisfied. □

5 Characterizing logics

In this section, we use Theorem 16 to show an algebraic characterization of the temporal logics CTL* and PDL.

5.1 The characterization

In this section we state the main result of Section 5, Theorem 17. The point of this theorem is to provide a base for four tree logics: PDL, CTL* , chain logic and first-order logic.

Before stating the theorem, we need to define the bases, then we state the theorem. A more extensive explanation of the bases can be found in Sections 5.2 and 5.3. Consider the following three classes of forest algebras:

- Let $\mathbb{V}$ be the class of forest algebras that satisfy the equations:

$$\begin{aligned} h + h &= h && \text{for } h \in H \ , \\ h + g &= g + h && \text{for } g, h \in H \ , \\ v(h+g) &= vh + vg && \text{for } v \in V, g, h \in H \ . \end{aligned}$$

- Let $\mathbb{V}_{ap}$ be the class of forest algebras in $\mathbb{V}$ that additionally satisfy:

$$v^\omega = v^\omega v \qquad \text{for } v \in V \ .$$

- Let $\mathbb{W}$ be the class of forest algebras that satisfy

$$\begin{aligned} h^\omega + h &= h^\omega && \text{for } h \in H \ , \\ h + g &= g + h && \text{for } g, h \in H \ , \\ vh &= vg && \text{for } v \in V, g, h \in H \ . \end{aligned}$$

We do not define the logics involved in the theorem here, for these the reader is referred to Theorem 22 for one possible definition:

Theorem 17

- *PDL is the class of languages tree-recognizable by* $\langle \mathbb{V} \rangle$.
- *CTL* is the class of languages tree-recognizable by* $\langle \mathbb{V}_{ap} \rangle$.
- *Chain logic is the class of languages tree-recognizable by* $\langle \mathbb{V} \cup \mathbb{W} \rangle$.
- *First-order logic is the class of languages tree-recognizable by* $\langle \mathbb{V}_{ap} \cup \mathbb{W} \rangle$.

5.2 Path languages

In this section we describe the classes $\mathbb{V}$ and $\mathbb{V}_{ap}$. We show that they correspond to languages that only look at the set of paths in a tree.

A *path* in an (A, B)-tree t is a word

$$t(\epsilon)\ t(v_0)\ t(v_0 v_1)\ \ldots\ t(v_0 v_1 \ldots v_n) \in B^*(A + \epsilon)$$

obtained by reading all the labels leading to some node $v_0 \ldots v_n$. If v_n is a leaf, then the path is called *maximal*. A language L of unranked trees is called *path-testable* if membership $t \in L$ depends only on the set of maximal paths in the tree t.

The following definition provides a link between word languages and tree languages:

Definition 18 Let $K \subseteq B^* A$ be a word language. We define $\mathsf{E}K$ to be the set of trees t over (A, B) where some maximal path belongs to K.

Proposition 19 For a regular tree language L, the following three conditions are equivalent:

1. The language L is path-testable.
2. The slim syntactic forest algebra of L belongs to $\mathbb{V}$.
3. L is definable by a boolean combination of formulas of the form $\mathsf{E}K$.

Proof
The implication from 3 to 1 is obvious.

We begin with the implication from 1 to 2. We show that the syntactic algebra of a path-testable language must satisfy the equations in the definition of $\mathbb{V}$. The first two equations are obvious, we only consider the last one. We need to show that for any forests $\overline{s}, \overline{t}$ and any context, the following forests are L-equivalent:

$$\theta(\vec{s} + \vec{t}) \quad \theta\vec{s} + \theta\vec{t} \ .$$

This is quite simple, since both sides of the equation have the same set of paths. However, it is important that we are only interested in the set of paths. Indeed, if the hole in the context θ has siblings, then there are more paths in the right-hand side of the equation than in the left-hand side.

Finally, we show the implication from 2 to 3. Assume that L is tree-recognized by a morphism

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

and that (H, V) satisfies the three equations in the definition of $\mathbb{V}$. As there are only finitely many trees with a root in A we will simply consider only trees from L with root label in B. Unraveling the definition of tree-recognizability, for each $b \in B$ there is a set $F_b \subseteq H$ such that

$$b\vec{t} \in L \quad \text{iff} \quad \alpha(\vec{t}) \in F_b \ .$$

For each $h \in H$, let

$$K_h = \{b_1 \cdots b_n a \in B^* A : \alpha(b_1 \cdots b_n a) = h\} \ .$$

We will show that

$$L = \bigcup_{b \in B, h_1 + \cdots + h_n \in F_b} \Big(\bigcap_{h \in \{h_1, \ldots, h_n\}} \mathsf{E}(bK_h) \cap \bigcap_{h \notin \{h_1, \ldots, h_n\}} \neg \mathsf{E}(bK_h) \Big) \ . \tag{5}$$

Note that the above expression is actually finite, since there are only finitely many possible nonequivalent expressions within the parentheses. Moreover, one can easily verify that each of the word languages K_h is regular, as long as (H, V) is finite.

Let us begin by establishing the right-to-left inclusion in (5). Consider a tree $b\vec{t}$ that satisfies the formula on the right-hand side for some fixed $h_1 + \cdots + h_n$:

$$b\vec{t} \in \Big(\bigcap_{h \in \{h_1, \ldots, h_n\}} \mathsf{E}(bK_h) \cap \bigcap_{h \notin \{h_1, \ldots, h_n\}} \neg \mathsf{E}(bK_h) \Big) \ . \tag{6}$$

Consider the transformation, which replaces a subtree of the form $b(\vec{s}+\vec{t})$ with a forest of the form $b\vec{s}+b\vec{t}$. This transformation preserves (6), and it also preserves images under α by the third equation of $\mathbb{V}$. If we apply this transformation repeatedly to the forest $\vec{t}$, we obtain a forest $t_1 + \cdots + t_m$, where in each t_i all inner nodes have degree one. Since $t_1 + \cdots + t_m$ satisfies (6), we must have

$$\{\alpha(t_1), \ldots, \alpha(t_m)\} = \{h_1, \ldots, h_n\} \ .$$

by definition of the languages K_h. By using the first two equations defining $\mathbb{V}$, we obtain

$$\alpha(t_1 + \cdots + t_m) = h_1 + \cdots + h_n \in F_b \ .$$

The result then follows, since the left-had side is the same as $\alpha(\vec{t})$, since the transformation preserved images under α.

The left-to-right inclusion in (5) follows from a similar argument (in fact, the reasoning above can be reversed). □

Here we just state an analogous description of $\mathbb{V}_{ap}$:

Proposition 20 For a regular tree language L, the following two conditions are equivalent:

1. The slim-syntactic forest algebra of L belongs to $\mathbb{V}_{ap}$.
2. L is definable by a boolean combination of formulas of the form $\mathsf{E}K$, where each word language K is definable in first-order logic.

Proof
The proof is the same as in Proposition 19. We obtain that the syntactic semigroup of each word language K_h satisfies $s^\omega = ss^\omega$. We apply Schützenberger's Theorem to get first-order definability. □

Note 7 A tree language L is *truly path-testable* if membership $t \in L$ only depends on the *sequence* of maximal paths in the tree (ordered from left to right with respect to the lexicographical ordering on leaves). For instance, the language "the leftmost leaf is a" is truly path-testable. One can find two languages L, K with the same syntactic forest algebra, but where L is truly path-testable but K is not. In particular, being truly path-testable cannot be judged based solely on the syntactic forest algebra.

5.3 Jellyfish algebras

In this section we describe the expressive power of forest algebras in the class $\mathbb{W}$. The essential idea is that we can only look at the labels in the successors of the root; these can be counted up to a finite threshold.

For $k \in \mathbb{N}$ and $b \in B$, a formula $\mathsf{E}^k\mathsf{X}b$ is true in those trees $c(t_1 + \cdots t_n)$ where at least k of the trees $t_1, \cdots, t_n$ have the root label b. (The formula $\mathsf{E}^k\mathsf{X}b$ does not care about the root label c.)

The following rather obvious proposition is given without proof:

Proposition 21 For a regular tree language L over (A, B), the following two conditions are equivalent:

1. The slim-syntactic forest algebra of L belongs to $\mathbb{W}$.
2. L is definable by a boolean combination of formulas of the forms $\mathsf{E}^k\mathsf{X}b$ and "the root label is a"; where $b \in B$ and $a \in A \cup B$.

5.4 A wreath product characterization

In this section we complete the proof of Theorem 17. To avoid four separate proofs, we define a temporal logic ETL. The point of this logic is to give a uniform framework for defining the four logics involved in Theorem 17. We then use Theorem 16 to characterize the class of languages definable in ETL and its various fragments.

Formulas of ETL over an alphabet (A, B) for are defined as follows.

- "The root label is a" is a formula, for any label $a \in A \cup B$.
- Boolean combinations of ETL formulas are ETL formulas.
- If φ is an ETL formula and $k \in \mathbb{N}$, then $\mathsf{E}^k\mathsf{X}\phi$ is a formula. This formula is satisfied in a tree $b(t_1 + \cdots + t_n)$ where at least k of the trees $t_1, \ldots, t_n$ satisfy φ.
- If Φ is a finite set of ETL formulas and $L \subseteq \Phi^*$ is a regular word language, then $\mathsf{E}L$ is a ETL formula. This formula is true in a tree t if there exists a maximal path $x_1 \ldots x_n$ and some word $\varphi_1 \cdots \varphi_n \in L$ such for all $i = 1, \ldots, n$ the subtree $t|_{x_i}$ satisfies the formula φ_i.

Example 7 The language "some a-labelled leaf is at even depth" is definable by the PDL formula $\mathsf{E}(\mathit{true}\ \mathit{true})^*a$.

The following theorem shows that ETL contains all logics from Theorem 17:

Theorem 22
The following can be found as fragments of ETL:

- *Chain logic: these are exactly languages definable in ETL.*
- *First-order logic: these are exactly languages definable in ETL, where only first-order definable word languages L are allowed for $\mathsf{E}L$.*
- *PDL: these are exactly languages definable in ETL, where the operator $\mathsf{E}^k\mathsf{X}\phi$ is not allowed.*
- *CTL* : these are exactly languages definable in ETL, where the operator $\mathsf{E}^k\mathsf{X}\phi$ is not allowed and only first-order definable word languages L are allowed for $\mathsf{E}L$.*

The last two statements are more or less the same as the definitions of PDL and CTL* . The first two characterisations are also known, we refer the reader to [5]. Theorem 16 allows to deduce the following corollary that gives the proof of Theorem 17:

Corollary 23 Let $\mathbb{L}, \mathbb{L}_{ap}$ and $\mathbb{K}$ be the language classes from Propositions 19, 20 and 21.

- Chain logic is exactly $\langle \mathbb{L} \cup \mathbb{K} \rangle$.
- First-order logic is exactly $\langle \mathbb{L}_{ap} \cup \mathbb{K} \rangle$.
- PDL is exactly $\langle \mathbb{L} \rangle$.
- CTL* is exactly $\langle \mathbb{L}_{ap} \rangle$.

6 Further work

We have presented an algebraic characterization of regular languages of finite trees. We have shown pertinence of this approach by characterizing several known logics. Unquestionably, there remains a number of basic questions to be answered.

This work is motivated by decidability problems for tree logics. As mentioned in the introduction, surprisingly little is known about them. We hope that this paper represents an advance, if only by making more explicit the algebraic questions that are behind these problems. The characterizations we have presented make it clear that a theory of wreath product decompositions of tree algebras would be very useful.

Wherever there is an algebraic structure for recognizing languages, there is an Eilenberg theorem. It would be interesting to study varieties of tree algebras. What classes of tree algebras can be defined using equations? What are the appropriate implicit operations – like the ω power – that can be used in the equations? A related topic concerns $\mathcal{C}$-varieties [18]. This is a notion from semigroup theory, which — among others — does away with the tedious distinction between semigroup and monoid varieties. Is there a $\mathcal{C}$-variety of tree algebras?

There are of course classes of tree languages — perhaps even more so in trees than words — that are not closed under boolean operations like, for instance, languages defined by deterministic top down automata. In the case of words, ordered semigroups extend the algebraic approach to such classes. It would be interesting to develop a similar concept of ordered tree algebras.

The logics considered in this paper do not permit to talk about the order on siblings in a tree. It would be worth to find the correct equations for logics with the order relation on siblings.

Finally, it is of course interesting to look at other kinds of trees. One can ask what is the right concept of tree algebras for languages of infinite trees. It is also not clear how to cope with trees of bounded branching.

References

[1] D. Beauquier and J-E. Pin. Factors of words. In *International Colloquium on Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 63 – 79, 1989.

[2] M. Benedikt and L. Segoufin. Regular languages definable in FO. In *Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 327 – 339, 2005.

[3] M. Bojańczyk. *Decidable Properties of Tree Languages*. PhD thesis, Warsaw University, 2004.

[4] M. Bojańczyk and I. Walukiewicz. Characterizing EF and EX tree logics. volume 3170 of *Lecture Notes in Computer Science*, pages 131–145, 2004.

[5] Mikolaj Bojanczyk. *Definable Properties of Tree Languages*. PhD thesis, Warsaw University, 2004.

[6] J. Cohen, D. Perrin, and J-E. Pin. On the expressive power of temporal logic. *Journal of Computer and System Sciences*, 46(3):271–294, 1993.

[7] Z. Ésik. An algebraic characterization of temporal logics on finite trees. parts i, ii, iii. In *1st International Conference on Algebraic Informatics*, pages 53 – 77, 79 – 99, 101 – 110. Aristotle University, Thessaloniki, 2005.

[8] Z. Ésik. Characterizing CTL-like logics on finite trees. *Theoretical Computer Science*, 356:136–152, 2006.

[9] Z. Ésik and Sz. Iván. Products of tree automata with an application to temporal logic. To appear.

[10] Z. Ésik and Sz. Iván. Some varietites of finite tree automata related to restricted temporal logics. To appear.

[11] Z. Esik and P. Weil. On certain logically defined tree languages. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 2003.

[12] U. Heuter. First-order properties of trees, star-free expressions, and aperiodicity. In *Symposium on Theoretical Aspects of Computer Science*, volume 294 of *Lecture Notes in Computer Science*, pages 136–148, 1988.

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

[14] J-E. Pin. Logic, semigroups and automata on words. *Annals of Mathematics and Artificial Intelligence*, 16:343–384, 1996.

[15] A. Potthoff. First-order logic on finite trees. In *Theory and Practice of Software Development*, volume 915 of *Lecture Notes in Computer Science*, pages 125–139, 1995.

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

[17] H. Straubing. *Finite Automata, Formal Languages, and Circuit Complexity*. Birkhäuser, Boston, 1994.

[18] H. Straubing. On logical descriptions of regular languages. In *LATIN*, volume 2286 of *Lecture Notes in Computer Science*, pages 528 – 538, 2002.

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

[20] D. Thérien and T. Wilke. Temporal logic and semidirect products: An effective characterization of the Until hierarchy. In *Foundations of Computer Science*, pages 256–263, 1996.

[21] 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.

[22] T. Wilke. Algebras for classifying regular tree languages and an application to frontier testability. In *International Colloquium on Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pages 347–358.

[23] 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.