

Unranked Tree Algebra

Mikołaj Bojańczyk
Warsaw University

Igor Walukiewicz
CNRS Bordeaux

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 [12] and McNaughton and Papert [9]. The problem is also decidable for words when only the successor relation is available [15, 1]. However, no algorithm is known for the case of tree languages, see [8, 11, 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 [5, 10, 13, 16, 17, 19]. Most of the results have been obtained using algebraic techniques of semigroup theory. Recently, there has even been some progress for tree languages [18, 6, 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.

Tree algebras are defined for unranked trees, where a node may have more than two successors, which are ordered. We feel that this more general setting is justified by cleaner definitions, where semigroup theory can be used more easily. The definition of a tree algebra itself is quite obvious and probably not new; the contribution lies in highlighting how algebraic concepts relate to tree languages.

We begin our discussion of tree algebras with the free tree algebra. For finite words, there is one natural free semigroup: the set of nonempty words along with composition. For unranked, ordered, finite trees there are two natural semigroups:

- *Vertical free semigroup.* Contexts – trees with a single hole in some leaf – along with context composition (see Figure ??).
- *Horizontal free semigroup.* Nonempty lists of trees along with list concatenation.

The first semigroup also induces an action on the second: a context can take a list of trees, substitute it for the hole and return the resulting tree as a one-element list. This is indeed an action, since for two contexts v, w and for a list of trees h , we have:

$$(v \circ w)(h) = v(w(h)),$$

i.e. it does not make a difference if we first compose v with w and then act on h , or if first we act w on h and then act v on the result. We call such a pair of semigroups a tree algebra¹.

In the case of words, a subset of the free semigroup induces a congruence, the Myhill-Nerode equivalence relation, which has finite index if the subset is regular. The same concepts apply to tree algebras, except that we get two congruences: one for the vertical semigroup and one for the horizontal semigroup. A regular language of finite

¹For technical reasons, our actual definition will require two additional conditions, called the insertion conditions.

trees can be thus seen as one where both congruences are of finite index.

An important property of a tree 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 tree 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.

The main result is that a language is definable in first-order logic if and only if it is recognized by a wreath product of tree algebras (H, V) that each satisfy:

- The horizontal semigroup H is aperiodic commutative;
- The vertical semigroup V is aperiodic;
- The tree algebra (H, V) is distributive, i.e. satisfies:

$$v(g \cdot h) = v(g) \cdot v(h) \quad \text{for every } v \in V \text{ and } g, h \in H.$$

Moreover, by tweaking the first two properties, we can get characterizations of three other logics: PDL, CTL* and chain logic.

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

2 Tree algebras and their wreath products

An *unranked Σ -tree* is a partial mapping $t : \mathbb{N}^* \rightarrow \Sigma$ whose domain is finite and prefix closed. Additionally, we do not allow horizontal gaps in the domain: if $t(v \cdot a)$ is defined for $v \in \mathbb{N}^*$, $a \in \mathbb{N}$, then so is $t(v \cdot b)$ for all $b < a$. Since we only consider unranked Σ -trees, we just write Σ -tree from now on. An element of the domain of t is called a *node* of t . Nodes are ordered by the prefix relation. The longest proper prefix of a node is its *parent*, a *successor* is defined symmetrically. 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 v* , denoted $t|_v$, assigns $t(v \cdot w)$ to a node w . A *forest* is a finite sequence of trees. The *successor forest* of a node is the forest of subtrees rooted in that node's successors.

A Σ -context is a $\Sigma \cup \{*\}$ -tree, where $*$ is a special symbol not in Σ . Moreover, $*$ occurs in exactly one leaf, which is

called the *hole*. The *empty context* is the one with the hole in the root. When C is a context and t is a tree, $C[t]$ is the tree obtained from C by replacing the hole with t . Similarly we define the composition of two contexts $C \circ D$.

A *semigroup* is a set together with a binary associative operation, denoted here by \cdot or \circ . A *monoid* is a semigroup with a neutral element ϵ , which satisfies $s \cdot \epsilon = \epsilon \cdot s = s$ for all s in the semigroup. An action of a semigroup S on a set Q is a mapping $a : S \times Q \rightarrow Q$ that satisfies

$$a(s \cdot t, q) = a(s, a(t, q)) \quad \text{for } s, t \in S, q \in Q.$$

We use a functional notation for actions, writing $s(q)$ instead of $a(s, q)$, when the action a is understood from the context. In this notation, the above axiom becomes $(s \cdot t)(q) = s(t(q))$.

2.1 Tree algebras

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

A *tree prealgebra* (H, V, act) is a pair of semigroups along with an action $act : V \times H \rightarrow H$ of V on H . We call V the *vertical semigroup* and H the *horizontal semigroup*. We will denote the semigroup operation of V by \circ and the semigroup operation of H by \cdot . We denote the action of V on H using functional notation, writing $v(h)$ instead of $act(v, h)$.

A *tree algebra* is a tree prealgebra (H, V, act) with two more actions $in_r, in_l : H \times V \rightarrow V$ satisfying *inserting conditions*, i.e. for all $h, g \in H$ and $v \in V$.

$$in_l(h, v)(g) = v(h \cdot g) \quad in_r(h, v)(g) = v(g \cdot h)$$

Note that a tree prealgebra is a special case of a transformation semigroup. The novelty is that the two sets are equipped with semigroup structure. Adding inserting conditions is technically important as they guarantee that we can go from forests to contexts and vice versa without problems.

Example: The *standard tree algebra over Σ* is the tree algebra where H is the set of nonempty Σ -forests (equipped with concatenation) and V is the set of nonempty contexts (equipped with context composition). A context acts on a forest substituting it into the hole, the returned value is a forest with just one tree. The left insertion operation $in_l(h, v)$ plugs a forest consisting of h and a hole into the hole of v (see Figure ??); similarly for in_r but the hole is to the right of h . It can be verified that the insertion conditions are satisfied. Observe that thanks to insertions we can construct arbitrary context without having the hole explicitly as a constant.

Note. It might seem more natural to call the above object the *free tree algebra*. As we will later see, the free tree algebra has to be slightly different – in the free tree algebra,

there is a distinction between leaf labels and inner node labels. It is much more to work with trees where there is only one type of labels, hence the somewhat artificial concept of standard tree algebra.

Example: 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 tree algebra from (H, V) it suffices to add actions. The action of V on H is just function application. The left and right insertions are uniquely determined by the inserting conditions.

A *tree algebra morphism* from (H, V, act, in_l, in_r) to $(G, W, act', in_l', in_r')$ is a pair (α, β) of functions $\alpha : H \rightarrow G$ and $\beta : V \rightarrow W$ that preserve all operations in the tree algebra: not only the displayed operations but also the multiplications in semigroups. A set L of Σ -forests is *recognizable* if there is a tree algebra morphism (α, β) from the standard tree algebra over Σ to some finite tree algebra (H, V, act, in_l, in_r) such that L is the inverse image $\alpha^{-1}(F)$ of some $F \subseteq H$. The morphism (α, β) is said to recognize L . A language of Σ -trees is recognizable if it is the restriction of a recognizable set of forests to forests with just one tree.

Example: Consider the set L of trees with an even number of nodes. We present here a finite tree algebra (H, V, act, in_l, in_r) recognizing L . Both H and V are $\{0, 1\}$ with addition modulo 2. All the actions are also addition. The recognizing morphism maps a context (resp. forest) onto 0 if it has an even number of nodes.

Example: A language L of Σ -trees is called *label-testable* if the membership $t \in L$ depends only on two sets of labels: those of internal nodes of t and those of leaves of t . (So a label in a leaf is considered different from a label in an inner node). The appropriate tree algebra is defined as follows. Both H and V are the same semigroup: the set of pairs of nonempty subsets of Σ 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.

2.1.1 Universal algebra viewpoint

Another way to look at a tree algebra is from the point of view of universal algebra. In this setting, a tree algebra is a two-sorted algebra (with the sorts being H and V) along with five operations: (i) semigroup operations in H and V , (ii) an action of V on H and (iii) two actions of H on V . Tree algebras are of course defined equationally by: (i) semigroup equations for H and V , (ii) equation saying that act is an action, (iii) insertion conditions for in_l and in_r . We even do not need to require that in_l and in_r are actions as this follows from the rest of the equations.

The universal algebra viewpoint gives us definitions of such concepts as subalgebra, cartesian product, quotient and morphism (which coincides with the previously given definition).

A free tree algebra can be slightly different than the standard tree algebra defined above. Let Σ be a set of generators for H and let Γ be a set of generators for V . One can verify that the free tree algebra contains forests where Σ are the leaf labels, while Γ are the labels of inner nodes. In this paper we are only interested in the case where $\Gamma = \Sigma$; the definition of the standard tree algebra takes only one parameter. However, it may be interesting in the future to consider also the case of $\Gamma \neq \Sigma$.

In the free tree algebra over Σ , the generators are a node context and a node node forest:

Any other element of the free tree algebra, be it a forest or a context, can be generated from the above elements using the tree algebra operations. In particular, a morphism from the free tree algebra is uniquely defined by specifying its values on these generators.

2.1.2 Tree algebras and regular languages

The point of tree algebras is to have an algebraic formalism for regular languages of unranked trees. In this section we show that languages recognized by finite tree algebras are exactly the regular languages. We use here a Myhill-Nerode definition of regular languages. We also introduce the concept of a syntactic algebra.

Let L be a set of Σ -trees. We associate with L two equivalence relations on the standard tree algebra over Σ :

- Two nonempty Σ -forests g, h are L -equivalent if for every (perhaps empty) Σ -context v , either both or none of the trees $v(g), v(h)$ belong to L .
- Two nonempty Σ -contexts v, w are L -equivalent if for every nonempty Σ -forest h , either both or none of the trees $v(h), w(h)$ are L -equivalent as forests.

A language is called *regular* if both of these equivalence relations are of finite index (in fact a finite index of the first implies a finite index of the second). This definition coincides with various other existing definitions of regular languages, such as via automata or monadic second-order logic.

One can verify that both equivalence relations are congruences with respect to the operations of the standard tree algebra. This allows us to define the quotient of the standard tree algebra with respect to L , where the horizontal semigroup H_L consists of equivalence classes of nonempty Σ -forests, while the vertical semigroup V_L consists of equivalence classes of nonempty Σ -contexts. The quotient tree algebra

$$(H^L, V^L, act^L, in_l^L, in_r^L)$$

is called the *syntactic tree algebra of L* . The morphism (α^L, β^L) which to every element of the standard tree algebra assigns its equivalence class in (H^L, V^L) is called the *syntactic morphism*. The syntactic morphism recognizes L . This shows the more difficult direction of the following equivalence:

Fact 2.1 A language is regular if and only if it is recognizable.

Moreover, the syntactic tree algebra of L can be found in any tree algebra recognizing L :

Lemma 2.2 The syntactic tree algebra of L is a quotient of any tree algebra recognizing L .

Proof

Let (H, V, act, in_l, in_r) be some algebra recognizing L , under a morphism (α, β) . Let $F \subseteq H$ be the “accepting set”, i.e. a tree t belongs to L if and only if $\alpha(t) \in F$.

We will prove that two nonempty forests (resp. contexts) with the same images under α (resp. β) must be L -equivalent. Let then g, h be two nonempty forests with the same images under α . Suppose that $v(h) \in L$ for some context v . We have $\alpha(v(h)) \in F$ and:

$$\alpha(v(h)) = \beta(v)(\alpha(h)) = \beta(v)(\alpha(g)) = \alpha(v(g))$$

So $\alpha(v(g)) \in F$ and in consequence $v(g) \in L$. Therefore we can define a function α' that assigns to each element of $h \in H$ the unique L -equivalence class of Σ -forests containing $\alpha^{-1}(h)$. We do the same thing for contexts, obtaining a function β' .

A little calculation shows that (α', β') is a tree algebra morphism, therefore the syntactic tree algebra of L is a quotient of (H, V, act, in_l, in_r) . \square

2.1.3 Equations

We will be using equations (in the sense of universal algebra) to specify classes of tree algebras. Consider for instance the class of tree algebras where the horizontal semigroup satisfies:

$$h \cdot h = h \quad \text{and} \quad g \cdot h = h \cdot g \quad \text{for } g, h \in H. \quad (1)$$

If the syntactic tree algebra of a language satisfies the equations, it means that the membership in the language does not depend on the order or multiplicity of successor subtrees.

An important corollary of Lemma 2.2 concerns equations. As equations are preserved under quotients we get the following.

Corollary 2.3 If some tree algebra recognizing L satisfies an equation, then so does the syntactic tree algebra of L .

Example: Consider the class TJ_1 of tree algebras that satisfy equations (1) and also:

$$v(g \cdot h) = v(g) \cdot v(h) \quad (v \circ w)(g) = v(h) \cdot w(h) \\ \text{for } v, w \in V, g, h \in H.$$

Recall that a language is *label testable* if membership in the language depends only on the set of labels in the tree (cf. example on page 3). We claim that a language is label testable if and only if its syntactic tree algebra is in TJ_1 . The “only if” part follows from Corollary 2.3: the tree algebra that calculates the set of labels in a tree satisfies the equations defining TJ_1 , hence so does the syntactic algebra. The “if” part is also simple: using the equations, we can rewrite any two trees into each other, as long as they have the same labels. The idea is that a tree t can be rewritten into a forest containing exactly the trees $a(b)$ or a , where b is a label of an inner node of t , while b is a label of leaf of t .

2.1.4 Monoids

In our definition, a tree algebra is a pair of semigroups. It is also perfectly valid to consider monoids instead of semigroups. The obtained tree algebras behave slightly differently, being better for some applications and worse for others.

Let us consider here the case when both the horizontal and vertical components are monoids. This means that the standard tree algebra contains the empty forest ϵ in the horizontal free monoid and the empty (identity) context in the vertical free monoid. This has some surprising consequences.

First of all, the generators become simpler. Indeed, for an alphabet Σ , the free algebra is generated only by contexts C_a of one letter and one hole since we can obtain a -leaves by substituting the empty forest ϵ into the hole. In particular, there is no distinction between leaf and inner labels in the free algebra; and hence there is no distinction between the standard and free tree algebra.

Second, the contexts become richer. For instance, we can apply the inserting condition to the empty context $*$ and obtain a context that we denote $*_{h:L}$. This is the context

$$h \quad *$$

that maps g to $h \cdot g$. In the standard tree algebra, as we have defined it, the contexts are forests of trees, with one of the trees having a hole.

Third, the inserting conditions are simplified. We only need to define $*_{h:L}$ and $*_{h,R}$, since $in_l(h, v) = v \circ *_{h:L}$.

The above may be seen as advantages of the monoid approach. There are some disadvantages, however. Consider for instance the class of tree algebras that satisfy the equation $v \circ w = v$ for all idempotents $v \in V$ and all $w \in V$

(recall that an element v is an idempotent if $v \circ v = v$). When V is a monoid, this collapses to $w = \epsilon$ and corresponds to trivial languages. When V is a semigroup, this is an interesting property and corresponds to languages that depend only on nodes of bounded depth.

From now on, we return to tree algebras in the semigroup sense.

2.1.5 Path languages

In this section we give an example of a class of languages that can be defined via equations in a tree algebra.

A *path* in a Σ -tree t is a word

$$t(\epsilon) t(v_0) t(v_0 v_1) \dots t(v_0 v_1 \dots v_n) \in \Sigma^*$$

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

We claim that a language is *path-testable* if and only if its syntactic tree algebra satisfies the following three equations:

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

This characterization follows from a more general statement proved in Theorem 4.4; it corresponds to the base of the induction therein. We hope, however, that at least the only if part of the statement is quite obvious: the syntactic tree algebra of any path-testable language must satisfy all three above equations. To our knowledge, this is the first decidable characterization of path-testable languages.

Note also that it is important that tree algebras considered here do not admit the empty forest ϵ in the horizontal semigroup. Indeed, if this were the case, the third equation would imply that

$$v(g) = v(\epsilon) \cdot v(g) \quad \text{for } v \in V, g \in H.$$

The above equation corresponds to languages which depend on paths that do not necessarily end in a leaf. It is not, for instance, satisfied by the path-testable language: “all leaves are at even depth”.

2.2 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. The only thing we need to do is to verify that the wreath product of two tree algebras is also a tree algebra, i.e. it verifies the inserting conditions.

Definition 2.4 (Wreath Product) Let

$\mathcal{B} = (H, V, act^{\mathcal{B}}, in_l^{\mathcal{B}}, in_r^{\mathcal{B}})$ and $\mathcal{A} = (G, W, act^{\mathcal{A}}, in_l^{\mathcal{A}}, in_r^{\mathcal{A}})$ be two tree algebras. The *wreath product* $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$ is the tree algebra $(I, U, act^{\mathcal{C}}, in_l^{\mathcal{C}}, in_r^{\mathcal{C}})$ defined as follows. The horizontal semigroup I is the product semigroup $H \times G$. The vertical semigroup U is $V^G \times W$, with the multiplication defined:

$$\begin{aligned} (f, w) \circ_U (f', w') &= (f'', w \circ_W w') \\ \text{where } f''(g) &= f(w'(g)) \circ_V f'(g) \end{aligned}$$

The action $act^{\mathcal{C}}$ of U on I is defined as follows:

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

The left insertion $in_l^{\mathcal{C}}$ of I on U is:

$$\begin{aligned} in_l^{\mathcal{C}}((h, g), (f, w)) &= (f', in_l^{\mathcal{A}}(g, w)) \\ \text{where } f'(g') &= in_l^{\mathcal{B}}(h, f(gg')) \end{aligned}$$

The definition of the right insertion $in_r^{\mathcal{C}}$ obtained by replacing in_l by in_r in the above.

The definition above is the standard definition of wreath product of two transformation semigroups (except for the insertions part). We will try to give the reader some intuition about this construction. The idea is that there are two layers of the tree algebra: one – \mathcal{A} – works on the tree first, and then lets the second layer – \mathcal{B} – read the output of the first. In the vertical semigroup of the wreath product, this is encoded on the two coordinates. The tree algebra \mathcal{C} works “business as usual” on the second coordinate (this choice of coordinates is traditional). On the first coordinate, there is a function f that waits for the result g of the first layer, and after receiving this result returns the appropriate “second level” vertical transformation $f(g) \in V$.

Lemma 2.5 The wreath product of two tree algebras is a tree algebra.

The proof of this lemma is found in the appendix.

Example: The cartesian product of two tree algebras $\mathcal{B} \times \mathcal{A}$ is a subalgebra of the wreath product $\mathcal{B} \circ \mathcal{A}$. Indeed, keeping the notation from the above definition, an element (v, w) of the vertical part of $\mathcal{B} \times \mathcal{A}$ can be represented by (f_v, w) of the vertical part of $\mathcal{B} \circ \mathcal{A}$, where f_v is the constant function with the value v . It can be checked that this is an algebra morphism which is injective. This observation shows that if a language is recognized by a cartesian product of two algebras then it is also recognized by their wreath product.

Let \mathbb{V}, \mathbb{W} be two classes of tree algebras. We write $\mathbb{W} \circ \mathbb{V}$ for the class of algebras $\{\mathcal{B} \circ \mathcal{A} : \mathcal{B} \in \mathbb{V}, \mathcal{A} \in \mathbb{W}\}$. We write $\langle \mathbb{V} \rangle$ for the class

$$\bigcup_{n \in \mathbb{N}} \mathbb{V}^n \quad \text{where } \mathbb{V}^n = \overbrace{\mathbb{V} \circ \dots \circ \mathbb{V}}^{n \text{ times}}.$$

When $n = 0$, we set \mathbb{V}^n to be the class containing only the trivial tree algebra where both semigroups have one element each.

3 Temporal logics over unranked trees

In this section we define *unranked extended temporal logic* UETL. We will use this as a framework to uniformly describe first-order logic, chain logic, CTL* and PDL.

There are two types of formulas in UETL: tree formulas and path formulas. A tree formula specifies a property of trees; its semantics is a set of trees. A path formula specifies a property of paths in trees; its semantics is a set of pairs (t, π) , where π is a path in the tree t (i.e. contiguous set of linearly ordered nodes, not necessarily maximal). For instance the path formula

$$[E^2(\Sigma^* a \Sigma^*)]^* b$$

is true in those pairs (t, π) where the leaf at the end of π has label b , while all other nodes on the path have at least two independent descendants labeled a . The syntax and semantics of UETL are defined as follows:

- Every letter a of the alphabet is a tree formula that is true in trees whose root label is a . Any boolean combination of tree formulas is a tree formula.
- For any $k \in \mathbb{N}$ and path formula ϕ , $E^k \phi$ is a tree formula that is satisfied in a tree that contains at least k maximal paths satisfying ϕ .
- Every tree formula ϕ is also a path formula, it is satisfied if the tree containing the path satisfies ϕ .
- If ϕ is a path formula, then ϕ^* is a path formula that is satisfied in paths that can be decomposed into zero or more fragments satisfying ϕ . Similarly we define disjunction $\phi + \varphi$, complementation $\neg \phi$ and concatenation $\phi \cdot \varphi$ for ϕ, φ path formulas.

We have chosen maximal paths in the semantics of E^k . Arbitrary paths would give the same logic, but require one more application of E^k . For instance, when E^k quantifies over maximal paths, the property “some leaf has label a ” is written as $E(a + b)^* a$. If E^k quantifies over arbitrary paths, we need to write $E(a + b)^*(a[\neg E(a + b)])$.

Example: The formula $E^2(a^* b (a + b)^*)$ is true in $\{a, b\}$ -trees that have at least two incomparable b 's. This property is not definable in PDL (nor in CTL*) over unranked trees. It is definable in first-order logic, since it can be rewritten as $\exists x, y. P_b(x) \wedge P_b(y) \wedge x \not\leq y \wedge y \not\leq x$. The formula $E(aa)^*$ is true in $\{a\}$ -trees that have a maximal path of even length. This property is not definable in first-order logic (nor in CTL*).

The following theorem can be shown using the same techniques as in [7, 3].

Theorem 3.1

- A language is definable in chain logic if and only if it is definable by a UETL tree formula.
- A language is definable in first-order logic if and only if it is definable by a UETL tree formula where the operator ϕ^* is not allowed.
- A language is definable in PDL if and only if it is definable by a UETL tree formula where the operator $E^k \phi$ is allowed only for $k = 1$.
- A language is definable in CTL* if and only if it is definable by a UETL formula where ϕ^* is not allowed and $E^k \phi$ is allowed only for $k = 1$.

The reader unfamiliar with the logics stated above may well treat this theorem as a definition. Note that for the variants of first-order and chain logics considered here, the signature does not contain the horizontal successor or order. Therefore, properties like “there is an a -labeled successor of v to the left of a b -labeled successor of v ” cannot be expressed.

4 The algebraic characterization

In this section we give an algebraic characterization of several tree logics in terms of wreath products. In subsection 4.1 we define four classes of tree algebras. In subsection 4.2 we show our main result, Theorem 4.4, which shows how wreath products of the four base classes characterize first-order logic, chain logic, CTL* and PDL, respectively.

4.1 The base classes

In this section we define the four base classes used in Theorem 4.4. These are defined using standard semigroup concepts of commutativity, aperiodicity and idempotency along with a “new” notion of distributivity. We then go on to prove that these classes behave properly, in particular it is decidable if a given language can be recognized by a tree algebra from any given base class.

First, we recall three important classes of semigroups:

- A semigroup is *idempotent* if it satisfies the equation $s \cdot s = s$ for all $s \in S$;
- A semigroup is *commutative* if it satisfies the equation $s \cdot t = t \cdot s$ for all $s, t \in S$;
- A semigroup is *aperiodic* if for some $n \in \mathbb{N}$ it satisfies the equation $s^n = s^n \cdot s$ for all $s \in S$.

The following definition was already mentioned in subsection 2.1.5:

Definition 4.1 A tree algebra (H, V) is *distributive* if it satisfies the equation

$$v(g \cdot h) = v(g) \cdot v(h) \quad \text{for every } v \in V \text{ and } g, h \in H.$$

Definition 4.2 We define the following four *base* classes of tree algebras:

- \mathbb{X}' are the distributive tree algebras where the horizontal semigroup is commutative idempotent;
- \mathbb{X} are the distributive tree algebras where the horizontal semigroup is commutative aperiodic;
- \mathbb{Y}' are the distributive tree algebras where the horizontal semigroup is commutative idempotent and the vertical semigroup is aperiodic;
- \mathbb{Y} are the distributive tree algebras where the horizontal semigroup is commutative aperiodic and the vertical semigroup is aperiodic;

Note that the base class \mathbb{X}' is exactly the class described in Section 2.1.5.

Lemma 4.3 It is decidable if a language is recognized by one of the base classes.

Proof

The base classes are defined by equations. Therefore, by Corollary 2.3, it is enough to verify if the syntactic tree algebra belongs to the base class. \square

4.2 The main theorem

In this section we state and prove the main theorem of the paper. This theorem gives a uniform characterization of four important tree logics: first-order logic, chain logic and the temporal logics PDL and CTL*.

Theorem 4.4

For every tree language L ,

- L is definable in PDL if and only if it is recognized by $\langle \mathbb{X}' \rangle$.
- L is definable in chain logic if and only if it is recognized by $\langle \mathbb{X} \rangle$.
- L is definable in CTL* if and only if it is recognized by $\langle \mathbb{Y}' \rangle$.
- L is definable in first-order logic if and only if it is recognized by $\langle \mathbb{Y} \rangle$.

The left to right implications are proved in Section 4.3 and the right to left implications are proved in Section 4.4.

4.3 Tree algebras recognize logics

In this section we prove the left to right implications in Theorem 4.4, i.e. we show that a language definable in PDL (resp. the other logics) can be recognized by $\langle \mathbb{X}' \rangle$ (resp. the other classes). We do the proof for PDL and only comment on the differences for the other cases. We need to show that for every tree PDL formula φ there is a tree algebra in $\langle \mathbb{X}' \rangle$ recognizing the set of trees that satisfy φ . Recall that \mathbb{X}' is the class of distributive algebras where the horizontal semigroup is commutative and idempotent.

By Theorem 3.1, we can use the UETL syntax for PDL. The proof is by induction on the structure of the formula. For every path formula, we show an appropriate tree algebra. The letter tests are obvious. The boolean operations are done using cartesian product. The hard step is the branching quantifier E (note that in PDL E^k is not allowed for $k \geq 2$). Consider a formula of the form $E\varphi$. Let Γ be the set of all tree formulas that are subformulas of φ . By induction hypothesis, each formula in Γ is recognized by $\langle \mathbb{X}' \rangle$. As shown in the example on page 5, the cartesian product of two tree algebras is a subalgebra of their wreath product. Thus if a language is recognized by a cartesian product of two tree algebras it is also recognized by their wreath product. Since $\langle \mathbb{X}' \rangle$ is closed under wreath products we may as well assume that there is one tree algebra $\mathcal{A} = (G, W, act, in_l, in_r)$ that simultaneously recognizes all the formulas in Γ . This means that there is a morphism (α, β) from the standard tree algebra into \mathcal{A} and a set $F_\Delta \subseteq G$ for each set of formulas $\Delta \subseteq \Gamma$ such that:

$$I(t) = \Delta \quad \text{iff} \quad \alpha(t) \in F_\Delta \quad (2)$$

where $I(t)$ is the set the set of formulas from Γ that are true in a tree t . Observe that $\{F_\Delta : \Delta = I(t) \text{ for some tree } t\}$ is a partition of G .

For a leaf $v = a_0 \cdots a_n$ in a tree t , we define $\pi_t(v)$ as

$$\pi_t(v) = I(t|_\epsilon) I(t|_{a_0}) \cdots I(t|_{a_0 \cdots a_n}) \in P(\Gamma)^+.$$

This sequence says what formulas from Γ are true on the path from the root to v . One can easily check that there is a regular word language $L \subseteq P(\Gamma)^+$ such that $E\varphi$ is satisfied in a tree t if and only if there is a leaf v satisfying $\pi_t(v) \in L$. Let S be a semigroup recognizing the word language L under the morphism $\rho : P(\Gamma)^+ \rightarrow S$, i.e.

$$\rho^{-1}(\rho(L)) = L.$$

We now proceed to define a tree algebra $\mathcal{B} = (H, V, act', in_l', in_r')$ that belongs to \mathbb{X}' and such that $\mathcal{B} \circ \mathcal{A}$ recognizes the language $E\varphi$. The idea is that H will keep track of the possible elements of S that can be obtained by evaluating ρ on some maximal path in the forest.

ja
znowu
zmie-
nilem...

- Elements of the the horizontal semigroup H are the nonempty subsets of S , i.e. $P(S) \setminus \{\emptyset\}$. The semigroup operation is union.
- The vertical semigroup V is $P(S) \times S$. The second coordinate describes the transformation on the path from the root to the hole, while the first coordinate keeps track of the results on all the other maximal paths (which may not exist). The operation is defined by:

$$(x, m) \circ (y, n) = (x \cup my, m \cdot n).$$

In the above, my is an abbreviation for $\{mn : n \in y\}$. The action of V on H is defined as follows for $x, y \in P(S)$ and $m \in S$:

$$(x, m)(y) = x \cup my.$$

This is indeed an action:

$$\begin{aligned} ((x, m)(y, n))z &= (x \cup my, mn)z = \\ &= x \cup my \cup (mn)z = x \cup m(y \cup nz) = \\ &= x \cup m((y, n)z) = (x, m)((y, n)z). \end{aligned}$$

- The left insertion is defined by:

$$in_l(x, (y, m)) = (y \cup mx, m).$$

The right insertion is the same as the left, since the trees are unordered. The insertion condition is satisfied:

$$\begin{aligned} in_l(x, (y, m))z &= (y \cup mx, m)z = \\ &= y \cup mx \cup mz = y \cup m(xz) = (y, m)(xz). \end{aligned}$$

To complete the proof, we will now show that \mathcal{B} is a tree algebra in \mathbb{X}' and that $E\varphi$ is recognized by $\mathcal{B} \circ \mathcal{A}$.

Lemma 4.5 \mathcal{B} is a tree algebra in \mathbb{X}' .

Proof

Since H clearly is idempotent and commutative, it remains to show that \mathcal{B} is distributive. Let (x, m) belong to V and let y, z be two elements of H :

$$\begin{aligned} (x, m)(yz) &= x \cup m(yz) = x \cup my \cup mz = \\ &= x \cup my \cup x \cup mz = (x, m)y(x, m)z. \end{aligned}$$

□

Lemma 4.6 $\mathcal{B} \circ \mathcal{A}$ recognizes $E\varphi$.

Proof

Recall that the language $L \subseteq P(\Gamma)^+$ is such that a tree t satisfies $E\varphi$ if and only if it contains a leaf v with $\pi_t(v)$ in L . Recall moreover, that $\rho : P(\Gamma)^+ \rightarrow S$ is a morphism recognizing L . We now proceed to define a tree algebra morphism (α'', β'') from the standard tree algebra over Σ into $\mathcal{B} \circ \mathcal{A}$ that recognizes $E\varphi$. The morphism will satisfy the following invariant for every Σ -tree t :

$$\text{if } \alpha''(t) = (h, g) \text{ then } h = \{\rho(\pi_t(v)) : v \text{ is a leaf in } t\} \quad (3)$$

In particular, the tree t satisfies $E\varphi$ if and only if the first coordinate of $\alpha''(t)$ contains an element from $\rho(L)$. Therefore in order to prove the lemma it suffices to define the morphism so that (3) is satisfied.

Such a morphism can be specified by just saying what elements are assigned to the generators. Recall that we have already a morphism (α, β) from the standard tree algebra to \mathcal{A} satisfying (2) and we can use it in our definition of (α'', β'') . There are two types of generators. First there are the leaf generators consisting just of a leaf labelled by a letter a . To such a leaf generator α'' simply assigns the unique pair (h, g) such that h satisfies the invariant (3) and $g = \alpha(a)$. Second, there are the context generators C_a which are contexts with one node labelled a . For each such context we need to defined an element

$$(f_a, w_a) \in V^G \times W$$

of the vertical semigroup in $\mathcal{B} \circ \mathcal{A}$. On the second coordinate, β'' behaves like β , so we set $w_a = \beta(C_a)$. To define the first coordinate f_a , we need to say what element $f_a(g) \in V$ is assigned to an element g of G . Observe that by (2) the value g determines which formulas from Γ hold in a tree t such that $\alpha(t) = g$. Let $I(g)$ denote this set of formulas. We put.

$$f_a(g) = (\emptyset, \rho(I(g))) \in P(S) \times S = V.$$

This completes the definition of the tree algebra morphism (α'', β'') .

We now proceed to show that the invariant (3) is satisfied. The proof is by induction on the depth of the tree. Consider a tree t with letter a labeling the root and with t_1, \dots, t_k being the subtrees in its successors. For $i = 1, \dots, k$, let (h_i, g_i) be $\alpha''(t_i)$. Using the induction assumption for the invariant, we have

$$h_i = \{\rho(\pi_{t_i}(v)) : v \text{ is a leaf in } t_i\}. \quad (4)$$

By definition of α'' , β'' and the action in the wreath product, we have

$$\alpha''(t) = (f_a(g_1 \cdots g_n)(h_1 \cdots h_n), w_a(g_1 \cdots g_n)) \in H \times G$$

Let $h \in H = P(S)$ be the first coordinate of $\alpha''(t)$; in order to show the invariant we need to prove that

$$h = \{\rho(\pi_t(v)) : v \text{ is a leaf in } t\}.$$

By definition of the function f_a , we have

$$h = (\emptyset, \rho(I(t)))(h_1 \cdots h_n).$$

By definition of the action of $V = P(S) \times S$ on $H = P(S)$, we have:

$$h = \emptyset \cup \rho(I(t))(h_1 \cdots h_n) = \bigcup_{i=1, \dots, k} \rho(I(t))h_i.$$

Finally, by using (4), this becomes

$$\begin{aligned} h &= \bigcup_{i=1, \dots, k} \rho(I(t))\{\rho(\pi_{t_i}(v)) : v \text{ is a leaf in } t_i\} = \\ &= \bigcup_{i=1, \dots, k} \{\rho(I(t)\pi_{t_i}(v)) : v \text{ is a leaf in } t_i\} = \\ &= \{\rho(\pi_t(v)) : v \text{ is a leaf in } t\}, \end{aligned}$$

which shows the invariant (3). \square

The other logics

Here we comment on how the proof must be adapted for the other logics. Consider first the case of chain logic, or more conveniently, UETL. We have to deal with the more general operator $E^k \phi$. The solution is to use multisets (up to threshold k) in (H, V) instead of sets of elements of S . The semigroup of multisets (up to threshold k) is aperiodic commutative.

For first-order logic and CTL* we proceed the same way, but with the additional assumption that the semigroup S recognizing the maximal paths is aperiodic. This can be done by Schützenberger's Theorem.

4.4 Logics describe tree algebras

We refer the reader to the appendix for proofs of the right to left implications in Theorem 4.4. Here we sketch out the essential ideas. The first idea is that wreath product of tree algebras can be captured by composition of logic formulas: a formula consults its subformulas, just as the top-level tree algebra consults the lower level tree algebra in the wreath product. Therefore, the only difficulty is showing that the base classes can be captured in the appropriate logics. The key idea here is that in a distributive tree algebra (all our base classes are distributive), the value of a tree depends only on the paths in the tree (our logics can talk about paths).

5 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. Recently, a flaw has been uncovered in a characterization [2] of tree languages definable in first-order logic with the successor but without the order. Maybe tree algebras could give means to correct the construction presented in that paper.

Wherever there is an algebraic structure for recognizing languages, there is an Eilenberg theorem. It would be interesting to study varieties of tree algebras. A related topic concerns \mathcal{C} -varieties [14]. 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?

What classes of tree algebras can be defined using equations? What are the appropriate symbols that can be used in the equations (is the ω power enough)?

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] J. Cohen, D. Perrin, and J. Pin. On the expressive power of temporal logic. *Journal of Computer and System Sciences*, 46(3):271–294, 1993.
- [6] 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.
- [7] T. Hafer and W. Thomas. Computation tree logic CTL and path quantifiers in the monadic theory of the binary tree. In *International Colloquium on Automata, Languages and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 260–279, 1987.
- [8] 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.
- [9] R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.
- [10] J. Pin. Logic, semigroups and automata on words. *Annals of Mathematics and Artificial Intelligence*, 16:343–384, 1996.
- [11] 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.
- [12] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.
- [13] H. Straubing. *Finite Automata, Formal Languages, and Circuit Complexity*. Birkhäuser, Boston, 1994.
- [14] H. Straubing. On logical descriptions of regular languages. In *LATIN*, volume 2286 of *Lecture Notes in Computer Science*, pages 528 – 538, 2002.
- [15] D. Thérien and A. Weiss. Graph congruences and wreath products. *Journal of Pure and Applied Algebra*, 36:205–215, 1985.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] T. Wilke. Classifying discrete temporal properties. In *Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46, 1999.

A Wreath product of tree algebras

In this appendix we prove Lemma 2.5, which says that the wreath product of two tree algebras is a tree algebra.

The wreath product of two transformation semigroups is a transformation semigroup, therefore the wreath product of two tree algebras is a tree prealgebra. We need to verify that the inserting conditions are satisfied. By symmetry we will only verify the left inserting condition. We keep the notation from the definition above.

Let (f, w) belong to U and let (h, g) belong to I . We need to verify that for all $(h', g') \in H \times G$:

$$in_i^C((h, g), (f, w))(h', g') = (f, w)((h, g) \cdot_I (h', g')) \quad (5)$$

By definition

$$in_i^C((h, g), (f, w)) = (f', in_i^A(g, w)) \\ \text{where } f'(g) = in_i^B(h, f(gg'))$$

We get

$$in_i((h, g), (f, w))(h', g') = (f', in_i^A(g, w))(h', g') = \\ (in_i^B(h, f(gg'))h', in_i^A(g, w)g') = \\ (f(gg')(hh'), w(gg')) = (f, w)(hh', gg') = \\ (f, w)((h, g)(h', g'))$$

where the second and fourth equations come from the definition of action in \mathcal{C} . In a similar manner we can verify the right inserting condition.

B Logics recognize tree algebras

In this appendix we show the right to left implications in Theorem 4.4, i.e. we show that a language recognized by $\langle \mathbb{Y} \rangle$ (resp. the other classes) can be defined in first-order logic (resp. the other logics). We first do the proof for first-order logic and then only comment how the other cases differ from this one.

B.0.1 First-order logic

We will show that if a tree language L is recognized by a morphism (α, β) into $(H, V, act, in_l, in_r) \in \mathbb{Y}^n$, then for every $h \in H$ there is a formula φ_h of first-order logic that is true in exactly those forests t such that $\alpha(t) = h$. Recall that \mathbb{Y} are the distributive tree algebras where the horizontal semigroup is commutative aperiodic and the vertical semigroup is aperiodic.

The proof is by induction on n . The case of $n = 0$ is trivial, since \mathbb{Y}^0 contains only the tree algebra $(\{\epsilon\}, \{\epsilon\})$, which can only recognize the formulas false and true.

Case of $n = 1$

Before we proceed to the general case, we do these case of $n = 1$. For an a -labeled node x in a tree t , let $I_t(x) \in V$ be the value $\beta(C_a)$. Here, C_a is the context with a in the root and a hole below (one of the generators of the standard vertical semigroup). Given a leaf $x = a_1 \dots a_k$ of t , let

$$\pi_t(x) = I_t(\epsilon) \circ I_t(a_1) \circ \dots \circ I_t(a_1 \dots a_{k-1}) \in V$$

be the result of multiplying all the values $I_t(y)$ for nodes $y < x$. Since V is aperiodic, by Schützenberger's Theorem says that for every $v \in V$ there is a first order formula $\psi_v(x)$ that holds in a leaf x iff $\pi_t(x) = v$. Let $J_t(x)$ be the result of applying $\pi_t(x)$ to the value of α on the forest containing just one leaf labeled with the same letter as the label of x . Again, for every $h \in H$ there is a first-order formula $\phi_h(x)$ that holds in a leaf x if and only if $J_t(x) = h$.

Given a set X and function $f : X \rightarrow H$, we write $\prod_{x \in X} f(x)$ for the multiplication in H of the values $f(x)$ for all elements $x \in X$. This notation makes sense, since H is commutative and therefore no order is needed over X . Let $\text{leaves}(t)$ be the set of leaves of the tree t .

We claim that the value $\alpha(t)$ can be obtained by multiplying in H the values $J_t(x)$ for all the leaves v in t , i.e.

$$\alpha(t) = \prod_{x \in \text{leaves}(t)} J_t(x) \quad (6)$$

Before we show this claim, we remark how it yields the desired result. Indeed, by commutativity of H , the value $\alpha(t)$ depends only on the number of occurrences of $\phi_h(x)$

for different $h \in H$. Moreover, by aperiodicity of H , the occurrences are counted only up to a certain finite threshold, which is the number m such that $h^{m+1} = h^m$ holds for all $h \in H$. Counting the number of nodes up to a finite threshold can be done by a first-order formula.

We now proceed to demonstrate (6). The proof is by induction on the depth of t . Let t be a tree with a in the root and successor forest $t_1 \dots t_k$.

$$\alpha(t) = \beta(C_a)(\alpha(t_1 \dots t_k)) = \beta(C_a) \prod_{i=1 \dots k} \alpha(t_i).$$

By using $\beta(C_a) = I_t(\epsilon)$, the induction assumption and distributivity this becomes

$$I_t(\epsilon) \prod_{i=1}^k \prod_{x \in \text{leaves}(t_i)} J_{t_i}(x) = \prod_{i=1}^k \prod_{x \in \text{leaves}(t_i)} I_t(\epsilon) J_{t_i}(x).$$

This yields the desired result, since for a leaf y of t of the form $k \cdot x$ we have

$$J_t(x) = I_t(\epsilon) J_{t_k}(x).$$

Case of $n > 1$

Let then L be a language recognized by a morphism (α, β) into $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$ where

$$\begin{aligned} \mathcal{C} &= (I, U, act'', in_l'', in_r''), \\ \mathcal{B} &= (H, V, act', in_l', in_r'), \quad \mathcal{A} = (G, W, act, in_l, in_r), \\ &\text{with } \mathcal{B} \in \mathbb{Y} \text{ and } \mathcal{A} \in \mathbb{Y}^{n-1}. \end{aligned}$$

We want to show that L is definable in first-order logic. We will reduce this case to the case of $n = 1$. Recall that by the definition of wreath product

$$I = H \times G, \quad U = V^G \times W$$

therefore (α, β) may be decomposed into two pairs (α_1, β_1) and (α_2, β_2) into (H, V^G) and (G, W) respectively. Moreover, the pair (α_2, β_2) is a tree algebra morphism. By induction assumption, for every $g \in G$ there is a formula φ_g of first order logic that is true in exactly those trees t where $\alpha_2(t) = g$.

For a letter $a \in \Sigma$, let v_a be the value $\beta_1(C_a) \in V^G$ of the context C_a under β_1 . For an a -labeled inner node x in a tree t we define

$$I_t(x) = v_a(\alpha_2(s_1 \dots s_m)) \in V.$$

where $s_1 \dots s_m$ is the successor forest of x in t . One can show that wreath product preserves aperiodicity and commutativity of the horizontal semigroup. In particular, the value $\alpha_2(s_1 \dots s_m)$ depends on the possible values

$\alpha_2(s_1), \dots, \alpha_2(s_m)$ and the number of times they occur (up to some given threshold). Therefore we can use the induction assumption to calculate the value $\alpha_2(s_1 \cdots s_m)$ — and consequently $I_t(x)$ — with a first-order formula.

Let I be the function which to a Σ -tree t assigns a tree $I(t)$ of the same domain but over the alphabet $H \cup V$ such that every inner node x of $I(t)$ is labeled by $I_t(x)$, while every a -labeled leaf is labeled by $\alpha_1(a)$. This is a tree whose inner nodes are labeled by V and whose leaves are labeled by H . Finally, let (γ, δ) be the unique morphism which to such a tree assigns an element of (H, V) and satisfies

$$\delta(C_v) = v \quad \gamma(t_h) = h .$$

Here t_h is a tree with a single node with label h . This is the morphism that simply evaluates the tree.

Let t be a Σ -tree. One can show by induction on the depth of t that

$$\alpha(t) = (\gamma(I(t)), \alpha_2(t)) .$$

It is therefore enough to show that we can calculate the value $\gamma(I(t))$ using a first-order formula. Since the labeling in the tree $I(t)$ can be calculated using a first-order formula, we have thus reduced our problem to calculating the value of the morphism (γ, δ) into $(H, V) \in \mathbb{Y}$, i.e. the case of $n = 1$.

The other logics

Finally, we comment on how the proof needs to be modified for the other logics. For CTL^* , the only difference is that the logic cannot count the number of leaves satisfying some formula (which is done in the paragraph after equation (6)). However, we never need to do this, since the base class \mathbb{Y}' is idempotent, hence it is only necessary to verify if there *exists* a node.

For the chain logic and PDL, the argument is analogous, except that we cannot use the assumption on aperiodicity of V (in the first paragraph of the case $n = 1$). However, since the path component of these logics can evaluate arbitrary regular expressions, this is not a problem.