Piecewise testable tree languages

Mikołaj Bojańczyk Luc Segoufin Howard Straubing
Warsaw university INRIA Boston college

***Abstract*—This paper presents a decidable characterization of tree languages that can be defined by a boolean combination of Σ_1 formulas. This is a tree extension of the Simon theorem, which says that a string language can be defined by a boolean combination of Σ_1 formulas if and only if its syntactic monoid is J-trivial.**

I. Introduction

Logics for expressing properties of labeled trees and forests figure importantly in several different areas of Computer Science. Research devoted to understanding and comparing the expressive power of such logics has raised a number of important questions that remain open: For instance, we do not possess an effective characterization of the properties of trees definable in first-order logic using a binary predicate $<$ that expresses the ancestor relation, nor of the properties definable in various temporal logics, such as CTL, CTL* or PDL.

In the case of logics for defining properties of *words,* such questions have been studied very successfully by applying ideas from algebra: A property of words over a finite alphabet A defines a set of words, that is a language $L \subseteq A^*$. As long as the logic in question is no more expressive than monadic second-order logic, L is a regular language, and definability in the logic often boils down to verifying a property of the *syntactic monoid* of L (the transition monoid of the minimal automaton of L). This approach dates back to the work of McNaughton and Papert [4] on first-order logic over $<$ (where $<$ denotes the usual linear ordering of positions within a word). A comprehensive survey, treating many extensions and restirictions of first-order logic, is given by Straubing [6]. Thérien and Wilke [9], [8], [7] similarly study temporal logics over words.

An important early discovery in this vein, due to I. Simon [5], treats word languages definable in first-order logic over $<$ with low quantifier complexity: A word language is definable by a boolean combination of Σ_1-sentences over $<$ if and only its syntactic monoid M is *$\mathcal{J}$-trivial*. This means that for all $m, m' \in M$, if $MmM = Mm'M$, then $m = m'$. (In other words, distinct elements generate distinct two-sided semigroup ideals.) Thus one can effectively decide, given an automaton for L, whether L is definable by such a sentence. (It should be noted that Simon did not discuss logic *per se*, but used piecewise testable languages, which are clearly equivalent to definability by a boolean combination of Σ_1-sentences.)

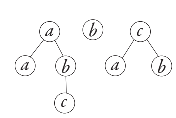
There has been some recent success in extending these methods to trees and forests. (We work here with unranked trees and forests, and not binary or ranked ones, since we believe that the definitions and proofs are cleaner in this setting.) The algebra is more complicated, because there are two multiplicative structures associated with trees and forests, both horizontal and a vertical concatenation. Benedikt and Segoufin [1] use these ideas to effectively characterize sets of trees definable by first-order logic with the parent-child relation. Bojańczyk [2] gives a characterization of properties definable in a temporal logic with unary ancestor and descendant operators. Bojańczyk and Walukiewicz [3] present the general theory of the 'forest algebras' that underlie these studies.

In the present paper we continue this line of inquiry, and provide a further illustration of the utility of these algebraic methods, by generalizing the theorem of Simon cited above. That is, we give necessary and sufficient conditions for a set L of unranked forests to be definable by a boolean combination of Σ_1-sentences (we consider various combinations of predicates, each with different conditions). These conditions are formulated as a collection of identities on the syntactic forest algebra of L, and thus are effectively verifiable if we have a deterministic tree automaton for L. We further generalize this result to the logic that includes a ternary 'closest common ancestor' relation. While we have to some extent drawn on Simon's original argument, the added complexity of the tree setting makes both formulating the correct condition and generalizing the proof quite nontrivial.

II. Notation

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

For example, the forest that we conventionally draw as



is the expression

$$a(a0 + bc0) + b0 + c(a0 + b0) \ .$$

Usually will not write the zeros and denote this forest by

$$t = a(a + bc) + b + c(a + b) \ .$$

This work was partially funded by the AutoMathA programme of the ESF and the PHC programme Polonium.

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

The notions of node, descendant and ancestor relations between nodes are defined in the usual way. We write $x < y$ to say that x is an ancestor or y or, equivalently, that y is a descendant of x. The *closest common ancestor* of two nodes x, y is a node z that is the unique node that is a descendant of all nodes that are ancestors of both x, y. As our forests are ordered, each forest induces a natural linear order between its nodes that we call the *forest-order* and which corresponds to the lexicographic order, or the depth-first left-first traversal of the forest.

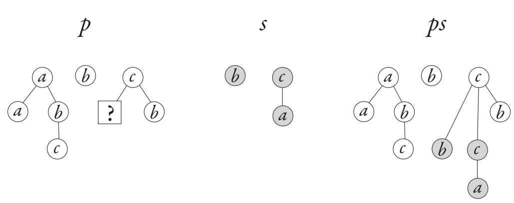
If we take a forest and replace one of the leaves by a special symbol $\Box$, we obtain a *context.* Contexts will be denoted using letters p, q, r. A forest s can be substituted in place of the hole of a context p, the resulting forest is denoted by ps. There is a natural composition operation on contexts: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

For example, from the forest t given above, we can obtain, among others, the context

$$p = a(a + bc) + b + c(\Box + b) \ .$$

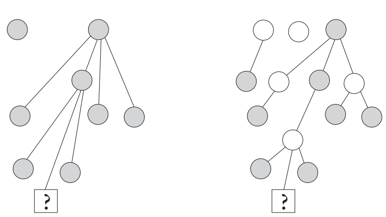
If $s = (b + ca)$, then

$$ps = a(a + bc) + b + c(b + ca + b) \ .$$



Piecewise testable languages: We say a forest s is an *immediate piece* of a forest s' is s, t can be decomposed as $s = pt$ and $s' = pqt$ for some contexts p, q and some forest t. The reflexive transitive closure of the immediate piece relation is called the *piece* relation. We write $s \preceq t$ to say that s is a piece of t. In other words, a piece of t is obtained by removing nodes from t.

We extend the notion of piece to contexts. In this case, the hole must be preserved while removing the nodes:



The notions of piece for forests and contexts are related, of course. For instance, if p, q are contexts with $p \preceq q$, then $p0 \preceq q0$. Also, conversely, if $s \preceq t$, then there are contexts $p \preceq q$ with $s = p0$ and $t = q0$.

A forest language L over A is called *piecewise testable* if there exists $n \geq 0$ such that membership of t in L is determined by the set of pieces of t of size n or less. The size of a piece is the size of the forest, i.e. the number of nodes. Equivalently, L is a finite boolean combination of languages $\{t : s \preceq t\}$, where s is a forest. Every piecewise testable forest language is regular, since given $n \geq 0$, there is a finite automaton that can calculate on input t the set of pieces of t of size no more than n.

Logic: Piecewise testability corresponds to definability in a logic, which we now describe. A forest can be seen as a logical relational structure. The domain of the structure is the set of nodes. The signature contains a unary predicate P_a for each symbol a of the label alphabet A, plus possibly some extra predicates, such as the descendant or lexicographic orders on nodes. Let Ω be a set of predicates. A $\Sigma_1(\Omega)$ formula is a formula $\exists x_1 \cdots x_n \ \gamma$, where the formula γ is quantifier free and uses predicates from Ω. The predicates Ω that we use always include $(P_a)_{a \in \Sigma}$ and equality, hence we do not explicitly mention them in the sequel. Initially we will be consider two predicates on nodes: the ancestor order $x < y$ and the lexicographic order $x <_{lex} y$. Later on, we will see what happens when the closest common ancestor is added, and the lexicographic order removed.

A forest language L can be defined by a $\Sigma_1(<, <_{lex})$ formula if and only if it is closed under adding nodes, i.e.

$$pt \in L \qquad \Rightarrow \qquad pqt \in L$$

holds for all contexts p, q and forests t. Furthermore, the above condition can be effectively decdied by inspecting any representation of the language L, e.g. a tree automaton. Far more interesting are the boolean combinations of properties definable in $\Sigma_1(<, <_{lex})$. It is easy to show that:

Proposition 1 A forest language is piecewise testable iff it is definable by a Boolean combination of $\Sigma_1(<, <_{lex})$ formulas.

But the above result does not give us effective characterization of either of the two equivalent descriptions. Such a characterization is the goal of this paper:

The problem: We want an algorithm deciding if a given regular forest language is piecewise testable.

As noted earlier, the corresponding problem for words was solved by Simon, who showed that a word language L is piecewise testable if and only if its syntactic monoid $M(L)$ is $\mathcal{J}$-trivial. Note that one can test if a monoid M is $\mathcal{J}$-trivial in polynomial time: for each $m \neq m' \in M$, one calculates the ideals MmM and $Mm'M$ and then verifies that they are different. Therefore, it is decidable if a given regular word language is piecewise testable. We assume that the language L is given by its syntactic monoid and syntactic morphism, or by some other representation, such as a finite automaton, from which these can be effectively computed.

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

III. Forest algebras

Forest algebras were introduced by Bojanczyk and Walukiewicz as an algebraic formalism for studying regular tree languages [3]. Here we give a brief summary of the definition of these algebras and their important properties. A forest algebra consists of a pair (H, V) of finite monoids, subject to some additional requirements, which we describe below. We write the operation in V multiplicatively and the operation in H additively, although H is not assumed to be commutative. We accordingly denote the identity of V by $\Box$ and that of H by 0.

We require that V act on the left of H. That is, there is a map

$$(h, v) \mapsto vh \in H$$

such that

$$w(vh) = (wv)h$$

for all $h \in H$ and $v, w \in V$. We further require that this action be *monoidal,* that is,

$$h \cdot \Box = h$$

for all $h \in H$, and that it be *faithful,* that is, if $vh = wh$ for all $h \in H$, then $v = w$.

We further require that for every $g \in H$, V contains elements $(\Box + g)$ and $(g + \Box)$ defined by

$$(\Box + g)h = h + g, (g + \Box)h = g + h$$

for all $h \in H$.

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

Let A be a finite alphabet, and let us denote by H_A the set of forests over A, and by V_A the set of contexts over A. Clearly H_A forms a monoid under $+$, V_A forms a monoid under composition of contexts (the identity element is the empty context $\Box$), and substitution of a forest into a context defines a right action of V_A on H_A. It is straightforward to verify that this action makes (H_A, V_A) into a forest algebra, which we denote A^{Δ}. If (H, V) is a forest algebra, then every map f from H_A to H has a unique extension to a forest algebra morphism $\alpha : A^{\Delta} \to (H, V)$ such that $\alpha(a\Box) = f(a)$ for all $a \in A$. In view of this universal property, we call A^{Δ} the *free forest algebra* on A.

We say that a forest algebra (H, V) *recognizes* a forest language $L \subseteq H_A$ if there is a morphism $\alpha : A^{\Delta} \to (H, V)$ and a subset X of H such that $L = \alpha^{-1}(X)$. It is easy to show that a forest language is regular if and only if it is recognized by a finite forest algebra.

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

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

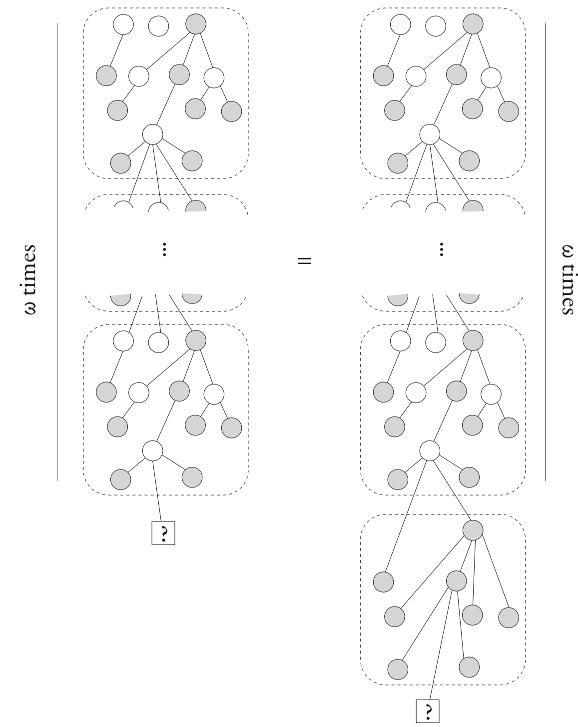


Fig. 1. The identity $u^\omega = u^\omega v$, with $v \preceq u$. The grey nodes are from v.

IV. Piecewise testable languages with only the descendant relation

The main result in this paper is the following characterization of piecewise testable languages:

Theorem 2 *A forest language is piecewise testable if and only if its syntactic algebra satisfies the identity*

$$u^\omega v = u^\omega = vu^\omega \qquad \textit{for } v \preceq u \; . \tag{1}$$

The identity (1) is illustrated in Figure 1.

Before we define the relation $v \preceq u$ used in identity (1), and also before we prove the theorem, we would like to show how it relates to the characterization of piecewise testable word languages given by Simon.

Let M be a monoid. For $m, n \in M$, we write $m \sqsubseteq n$ if m is a—not necessarily connected—subword of n, i.e. there are elements $n_1, \ldots, n_{2k} \in M$ such that

$$n = n_0 n_2 \cdots n_{2k} \qquad m = n_1 n_3 \cdots n_{2k-1} \; .$$

We claim that, using this relation, the word characterization can be written in a manner identical to Theorem 2:

Theorem 3 *A word language is piecewise testable if and only if its syntactic monoid satisfies the identity*

$$n^\omega m = n^\omega = mn^\omega \qquad \textit{for } m \sqsubseteq n \ . \tag{2}$$

Proof
Recall that Simon's theorem says a word language is piecewise testable if and only if its syntactic monoid is $\mathcal{J}$-trivial. Therefore, we need to show $\mathcal{J}$-triviality is equivalent to (2). We use an identity known to be equivalent to $\mathcal{J}$-triviality:

$$(nm)^\omega n = (nm)^\omega = m(nm)^\omega \ . \tag{3}$$

Since the above identity is an immediate consequence of (2), it suffices to derive (2) from the above. We only show $n^\omega m = n^\omega$. By assumption on $m \preceq n$, there are decompositions

$$n = n_0 n_1 \cdots n_{2k-1} n_{2k} \qquad m = n_1 n_3 \cdots n_{2k-3} n_{2k-1} \ .$$

By induction on i, we show

$$n^\omega n_1 n_3 \cdots n_{2i-3} n_{2i-1} = n^\omega \ .$$

The base $i = 0$, is immediate. In the induction step, we use the induction assumption to get:

$$n^\omega n_1 \cdots n_{2i-3} n_{2i-1} = n^\omega n_{2i-1} \ .$$

By applying (3), we have

$$n^\omega = n^\omega n_0 n_1 \cdots n_{2i-1} n_{2i}$$

and therefore

$$n^\omega n_{2i-1} = n^\omega n_0 n_1 \cdots n_{2i} n_{2i-1}$$

By applying (3) once again, the right side of the above becomes n^ω. □

Note that since the vertical monoid in a forest algebra is a monoid, it would make syntactic sense to have the relation $\sqsubseteq$ instead of $\preceq$ in Theorem 2. Unfortunately, the "if" part of such a statement would be false. That is why we need to have a different relation $\preceq$ on the vertical monoid, whose definition involves all parts of a forest algebra, and not just composition in the vertical monoid.

In Section IV-A, we define the $\preceq$ relation that is used in (1). We will also show that in a given finite forest algebra, $\preceq$ can be computed in polynomial time; an important corollary is that one can decide if a forest language is piecewise testable. Then, in Sections IV-B and IV-C, we prove both implications of Theorem 2. Finally, in Section IV-E, we give an equivalent statement of Theorem 2, where the relation $\preceq$ is not used.

A. The piece relation in a forest algebra

Recall that in Section II, we defined the piece relation for contexts in the free forest algebra. We now extend this definition to an arbitrary forest algebra (H, V). The general idea is that a context $v \in V$ is a piece of a context $w \in V$ if one can construct a term (using elements of H and V) which evaluates to w, and then take out some pieces of this term and get v. A more formal definition follows below.

Definition 4 Let (H, V) be a forest algebra. We say $v \in V$ *is a piece* of $w \in V$, denoted by $v \preceq w$, if $\alpha(p) = v$ and $\alpha(q) = w$ hold for some morphism

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

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

As we will see in the proof of Lemma 5, in the above definition, we can replace the term "some morphism" by "any surjective morphism". The following example shows that although the piece relation is transitive in the free algebra A^Δ, it may no longer be so in a finite forest algebra.

Example: Consider the syntactic algebra of the language $\{abcd\}$, which contains only one forest, and this forest has just one path, labeled by $abcd$. The context part of the syntactic algebra has twelve elements: an error element ∞, and one element for each infix of $abcd$. We have

$$a \preceq aa = \infty = bd \preceq bcd$$

but we do not have $a \preceq bcd$.

We will now show that in a finite forest algebra, one can compute the relation $\preceq$ in polynomial time. The idea is to use a different but equivalent definition. Let R be the smallest relation on V that satisfies the following rules, for all $v, v', w, w' \in V$:

$$\begin{array}{rcll}
\Box & R & v & \\
v & R & v & \\
vw & R & v'w' & \text{if } v \mathrel{R} v' \text{ and } w \mathrel{R} w' \\
\Box + v0 & R & \Box + v'0 & \text{if } v \mathrel{R} v' \\
v0 + \Box & R & v'0 + \Box & \text{if } v \mathrel{R} v'
\end{array}$$

Lemma 5 The relations R and $\preceq$ are the same.

In any finite algebra, the relation R can be computed by applying the rules until no new relations can be added. This gives the following corollary:

Corollary 6 In any given forest algebra, the relation $\preceq$ on contexts (also on forests) can be calculated in polynomial time.

Proof (of Lemma 5)
Let $\alpha : A^\Delta \to (H, V)$ be any surjective morphism. By induction on the number of steps used to derive $v \mathrel{R} w$, one produces contexts $p \preceq q$ with $\alpha(p) = v$ and $\alpha(q) = w$. In particular, this proves the remark above that the term "some morphism" can be replaced by "any surjective morphism".

For the inclusion of $\preceq$ in R, we show that $\alpha(p) \mathrel{R} \alpha(q)$ holds for all contexts $p \preceq q$. The proof is by induction on the size of p:

- If p is the empty context, then the result follows thanks to the first rule in the definition of R. If p consists of a single letter a and the hole below, then we use the first three rules.
- If there is a decomposition $p = p_1 p_2$ then there must be a decomposition $q = q_1 q_2$ with $p_1 \preceq q_1$ and $p_2 \preceq q_2$. The

existence of such a decomposition is proved by induction on the size of p_1. Then $\alpha(p) \preceq \alpha(q)$ follows from the induction assumption by using the third rule.

- $p = s+\square$. We can assume that s is a tree, since otherwise the context p can be decomposed as $(s_1 + \square)(s_2 + \square)$. Since s is a tree, it can be decomposed as $ap'0$, with a being a context with a single letter and the hole below and p' a context smaller than p. By inspecting the definition of $\preceq$, there must be some decomposition $q = q_0(aq'0 + q_1)$ or $q = q_0(q_1 + aq'0)$, with $p' \preceq q'$. By induction assumption, $\alpha(p')\ R\ \alpha(q')$. From this the result follows by applying rules three, four and five.

$\square$

Corollary 7 It is decidable if a forest language is piecewise testable.

Proof
We assume the language is given by its syntactic forest algebra, which can be computed in polynomial time from any recognizing forest algebra. The new equations can easily be verified in polynomial time by enumerating all possible elements of H, V. $\square$

The above procedure gives an exponential upper bound for the complexity in case the language is represented by a deterministic or even nondeterministic automaton, since there is an exponential translation from automata into forest algebra. We do not know if this upper bound is optimal.

B. Correctness of the identities

In this section we show the easy implication in Theorem 2.

Proposition 8 If a language is piecewise testable, then its syntactic algebra satisfies identity (1).

We will use the following simple fact:

Fact 9 If $p \preceq q$ are contexts and t is a forest, then $pt \preceq qt$.

Proof (of Proposition 8)
Fix then a language L that is piecewise testable and let n be such that membership $t \in L$ only depends on the pieces of t with at most n nodes.

We only show the first part of the identity, i.e.

$$u^\omega v = u^\omega \qquad \text{for } v \preceq u$$

Fix now $v \preceq u$ as above. By definition of ω, we can write the equation as an implication: for $k \in \mathbb{N}$, if $u^k = u^k \cdot u^k$ then $u^k \cdot v = u^k$. Let then k be as above. Let $p \preceq q$ be contexts that are mapped to v, u by the syntactic morphism. By unraveling the definition of syntactic algebra, we need to show that

$$rq^k pt \in L \qquad \text{iff} \qquad rq^k t \in L$$

holds for any context r and forest t. Consider now the forests

$$rq^{ik}t \qquad rq^{ik}pt \qquad \text{for } i \in \mathbb{N} .$$

Thanks to Fact 9, we get

$$rq^{ik}t \ \preceq\ rq^{ik}pt \ \preceq\ rq^{(i+1)k}pt$$

Therefore, for sufficiently large i, the two forests have $rq^{ik}t$ and $rq^{ik}pt$ have the same pieces of size n, and either both belong to L, or both are outside L. However, by assumption on q^k being equivalent to $q^k q^k$ under the syntactic morphism, we get the desired result by:

$$\begin{array}{lcl} rq^{ik}t \ \in L & \text{iff} & rq^k t \in L \\ rq^{ik}pt \ \in L & \text{iff} & rq^k pt \in L \ . \end{array}$$

$\square$

C. Completeness of the equations

This section, as well as the next Section IV-D, is devoted to showing completeness of the equations: an algebra that satisfies identity (1) in Theorem 2 can only recognize piecewise testable languages. We fix an alphabet A, and a forest language L over this alphabet, whose syntactic forest algebra (H, V) satisfies identity. We will denote the syntactic morphism by α, and sometimes use the term "type of s" for the image $\alpha(s)$ (likewise for contexts).

We write $s \sim_n t$ if the two forests s, t have the same pieces of size n. Likewise for contexts. The completeness proof follows from the following two results.

Lemma 10 Let $n \in \mathbb{N}$. For k sufficiently large, if two forests satisfy $s \sim_k s'$, then they have a common piece t in the same $\sim_n$ class, i.e.

$$t \preceq s \quad t \preceq s' \quad t \sim_n s \quad t \sim_n s' \ .$$

Proposition 11 For n sufficiently large, $pat \sim_n pt$ entails $\alpha(pat) = \alpha(pt)$.

The completeness part of Theorem 2 clearly follows from the above to results. Indeed, take n as in Proposition 11, and then apply Lemma 10 to this n, yielding k. We show that $s \sim_k s'$ implies $s \in L \iff s' \in L$, which immediately shows that L is piecewise testable, by inspecting pieces of size k. Indeed, assume $s \sim_k s'$, and let t be their common piece as in Lemma 10. Since t is a piece of s with the same pieces of size n, it can be obtained from s by a sequence of steps where a single letter is removed without affecting the $\sim_n$-class. Each such step preserves the type thanks to Proposition 11. Applying the same argument to s', we get

$$\alpha(s) = \alpha(t) = \alpha(s') \ ,$$

which gives the desired conclusion.

We begin by showing Lemma 10, and then the rest of this section is devoted to proving Proposition 11, the more involved of the two results.

We begin with the following simple observation.

Fact 12 Let K be a regular language. There is some constant k, such that every $t \in K$ contains a piece $s \in K$ of size at most k.

Proof
When applying a pumping argument to t, we get a piece. $\square$

We are now ready to prove Lemma 10. Fix $n \in \mathbb{N}$. Notice that each $\sim_n$ class is a regular language and $\sim_n$ has finitely many classes. We claim the lemma holds for k the maximum of n and the numbers obtained by Fact 12 for each class of $\sim_n$. Indeed, take any two forests $s \sim_k s'$. Let t be the piece of s of size at most k with $s \sim_n t$, as given by Fact 12. Since $s \sim_k s'$, the forest t is also a piece of s'. Furthermore since $\sim_k$ implies $\sim_n$ (by $k \geq n$), we get $s' \sim_n s \sim_n t$, which implies $s' \sim_n t$ by transitivity of $\sim_n$.

D. Fractals

We now show Proposition 11. Let us fix a context p, a label a and a forest t as in the statement of the proposition. The context p may be empty, and so may be the forest t. We search for the appropriate n; the size of n will be independent of p, a, t. We also fix the types $v = \alpha(p)$, $h = \alpha(t)$ for the rest of this section. In terms of these types, our goal is to show that $vh = v\alpha(a)h$. To avoid clutter, we will sometimes identify a with its image $\alpha(a)$, and write $vh = vah$ instead of $vh = v\alpha(a)h$.

Let s be a forest and X be a set of nodes in s. The *restriction of s to X*, denoted $s[X]$, is the piece of s obtained by only keeping the nodes in X.

Let s be a forest, X a set of nodes in s, and $x \in X$. We say that $x \in X$ is a vah-decomposition of s if: a) if we restrict s to X, remove descendants of x, and place the hole in x, the resulting context has type v; b) the node x has label a; c) if we restrict s to X and only keep nodes in X that are proper descendants of x, the resulting forest has type h.

Definition 13 A *fractal* of length k inside a forest s is a sequence $x_1 \in X_1 \ \cdots \ x_k \in X_k$ of vah-decompositions, where $X_i \subseteq X_{i+1} \setminus \{x_{i+1}\}$ holds for $i < k$.

A *subfractal* is extracted by only using a subsequence

$$x_{i_1} \in X_{i_1} \qquad \cdots \qquad x_{i_j} \in X_{i_j}$$

of the vah-decompositions.

Lemma 14 Let $k \in \mathbb{N}$. For n sufficiently large, $pat \sim_n pt$ entails the existence of a fractal of length k inside pat.

Proof
The proof is by induction on k. The case $k = 1$ is obvious.

Assume the lemma is proved for k and n and consider the case $k + 1$. Using a pumping argument as in Fact 12, we can show that for some m, if there is a fractal of length k, then this fractal has a piece of size at most m, which is also a fractal of length k. Without loss of generality we assume that $m > n$.

Assume now that $pat \sim_m pt$. By induction assumption, as $m > n$, we obtain a piece of pt which is a fractal of length k. From the previous observation, this piece can be assumed of size smaller than m. Clearly, this fractal can be extended to a fractal of length $k + 1$ by taking for X_{k+1} all the nodes of pat and for x_{k+1} the node a. $\square$

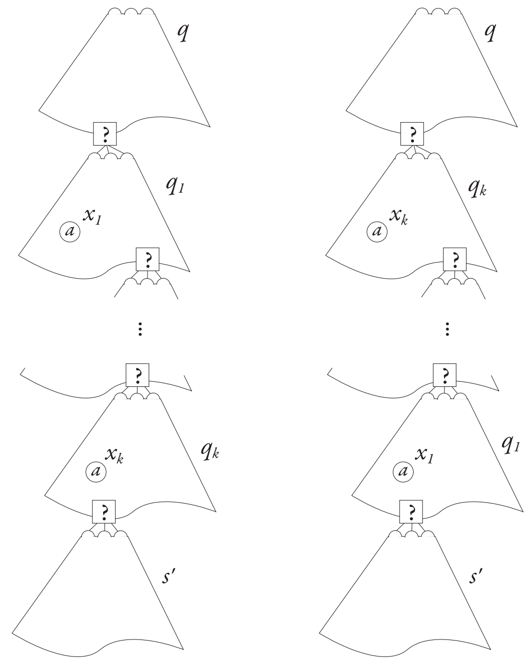


Fig. 2. Two types of tame fractal.

Thanks to the above lemma, Proposition 11 is a consequence of the following result:

Proposition 15 For k sufficiently large, the existence of a fractal of length k entails $vh = vah$.

The rest of this section is devoted to a proof of this proposition. The general idea is as follows. Using some simple combinatorial arguments, and also the Ramsey Theorem, we will show that there is also a large subfractal whose structure is very regular, or tame, as we call it. We will then apply identity (1) to this regular fractal, and show that a node with label a can be eliminated without affecting the type.

A fractal $x_1 \in X_1 \ \cdots \ x_k \in X_k$ inside a forest s is called *tame* if s can be decomposed as $s = qq_1 \cdots q_k s'$ (or $s = qq_k \cdots q_1 s'$) such that for each $i = 1, \ldots, k$, the node x_i is part of the context q_i, see Fig. 2. This does not necessarily mean that the nodes $x_1, \ldots, x_k$ form a chain, since some of the contexts q_i may be of the form $\square + t$.

Lemma 16 Let $k \in \mathbb{N}$. For n sufficiently large, if there is a fractal of length k, then there is a tame fractal of length k.

Proof
The main step is the following claim.

Claim 17 Let $m \in \mathbb{N}$. For sufficiently large n, for every forest s, and every set X of at least n nodes, there is a decomposition $s = qq_1 \cdots q_m s'$ where every context q_i contains at least one node from X.

Proof
Let Y be the set of nodes in s which are closest common ancestors of some two distinct nodes in X. The degree of a node in $y \in Y$ is defined to be the number of nodes $z \in Y \cup X$ such that all nodes in the path between y and z are outside Y. Take n to be m^m. Two cases may hold: either there is a node in Y with degree m, or Y contains a chain of length m. In both cases we get the conclusion of the lemma, but in the first case we need to use a decomposition where the contexts q_i have the hole in the root. □

We now come back to the proof of the lemma. For $k \in \mathbb{N}$ let n be the number defined by Lemma 17 for $m = k^2$. Let s, $x_1 \in X_1 \cdots x_k \in X_k$ be a fractal of length k. We apply Lemma 17, with $X = \{x_1, \dots, x_k\}$ and obtain a decomposition $s = qq_1 \cdots q_m s'$. For each $i = 1, \dots, m$ the context q_i contains at least one node of X. We chose arbitrarily one of them and denote it by x_{n_i}. Unfortunately, the function $i \mapsto n_i$ need not be monotone, as required in a tame fractal. However, we can always extract a monotone subsequence, since any number sequence of length k^2 is known to have a monotone subsequence of length k. □

We now assume there is a tame fractal $x_1 \in X_1 \cdots x_k \in X_k$ inside a forest s, which is decomposed as $s = qq_1 \cdots q_k s'$, with the node x_i belonging to the context q_i. The dual case when the decomposition is $s = qq_k \cdots q_1 s'$, corresponding to a decreasing sequence in the proof of Lemma 16, is treated analogously.

The general idea is as follows. We will define a notion of monochromatic tame fractal, and show that $vah = vh$ follows from the existence of large enough monochromatic tame fractal. Furthermore, a large monochromatic tame fractal can be extracted from any sufficiently large tame fractal thanks to the Ramsey Theorem.

Let i, j, l be such that $0 \leq i < j \leq l \leq k$. We define u_{ijl} to be the image under α of the context obtained from $q_{i+1} \cdots q_j$ by only keeping the nodes from X_l (with the hole staying where it is). We define w_{ijl} to be the image under α of the context obtained from $q_{i+1} \cdots q_j$ by only keeping the nodes from $X_l \setminus \{x_l\}$. Straight from this definition, we have

$$w_{ijl} \preceq u_{ij(l+1)} \text{ and } u_{ijl} \preceq u_{ij(l+1)} \tag{4}$$

A tame fractal is called *monochromatic* if for all $i < j < l$ and all $i' < j' < l'$ taken from $\{1, \dots, k\}$, we have

$$u_{ijl} = u_{i'j'l'} \ .$$

Note that in the above definition, we require $j < l$, even though u_{ijl} is defined even when $j \leq l$.

To show that monochromatic tame fractals exist, we use the Ramsey Theorem, in a hypergraph version. A *C-colored (undirected) d-dimensional (complete) hypergraph over nodes A* is a mapping f which associates a color from C to every d-element subset of A. A sub-hypergraph is defined by restricting the nodes to some subset $B \subseteq A$. A hypergraph is *monochromatic* if f uses only one color from c.

Theorem 18 (Ramsey Theorem) *Let $m, d \in \mathbb{N}$, and let C be a finite set of colors. If k is sufficiently large, then any C-colored d-dimensional hypergraph of k nodes has a monochromatic sub-hypergraph of m nodes.*

Lemma 19 If there is a tame fractal of sufficiently large size, then there is a monochromatic tame fractal of size $k = \omega + 2$.

Proof
Application of the Ramsey Theorem. □

We conclude by showing the following result:

Lemma 20 If there is a monochromatic tame fractal of size $k = \omega + 2$, then $vah = vh$.

Proof
Let $k = \omega + 2$, and fix a monochromatic tame fractal $x_1 \in X_1 \cdots x_k \in X_k$ inside a forest $s = qq_1 \cdots q_k s'$. Since $x_k \in X_k$ is a vah decomposition, the statement of the lemma follows if α assigns the same type to the two restrictions $s[X_k]$ and $s[X_k \setminus \{x_k\}]$.

Recall the definition of u_{ijl} and w_{ijl} above. The type of the forest $s[X_k]$ can be decomposed as

$$\alpha(s[X_k]) = \alpha(q[X_k]) \cdot u_{12k} \cdot u_{23k} \cdots u_{(k-1)kk} \cdot \alpha(s'[X_k])$$

The type of $s[X_k \setminus \{x_k\}]$ is decomposed the same way, only $u_{(k-1)kk}$ is replaced by $w_{(k-1)kk}$. Therefore, the lemma will follow if

$$u_{12k} \cdot u_{23k} \cdots u_{(k-1)kk} = u_{12k} \cdot u_{23k} \cdots w_{(k-1)kk} \ .$$

Since the fractal is monochromatic, and since k is greater than ω, the above becomes

$$u_{12k}^{\omega} \cdot u_{(k-1)kk} = u_{12k}^{\omega} \cdot w_{(k-1)kk} \ .$$

By (4) and monochromaticity we have

$$\begin{aligned} w_{(k-1)kk} &\preceq u_{(k-1)k(k+1)} = u_{12k} \\ u_{(k-1)kk} &\preceq u_{(k-1)k(k+1)} = u_{12k} \ . \end{aligned}$$

Therefore equation (1) can be applied to show that both sides are equal to u_{12k}^{ω}. Note that we use only one side of equation (1), $u^{\omega} v = u^{\omega}$. We would have used the other side when considering the case when $s = qq_k \cdots q_1 s'$. □

E. An equivalent set of identities

In this section, we rephrase the identities used in Theorem 2. There are two reasons to rephrase these.

The first reason is that identity (1) refers to the relation $v \preceq w$. One consequence is that we need to prove Corollary 6 before concluding that identity (1) can be checked effectively.

The second reason is that we want to pinpoint how identity (1) diverges from $\mathcal{J}$-triviality of the context monoid V. As witnessed by the forest language "all trees in the forest are aa", $\mathcal{J}$-triviality of the syntactic context monoid is not sufficient for the language to be piecewise testable. The proposition below identifies the additional condition that must be added to $\mathcal{J}$-triviality.

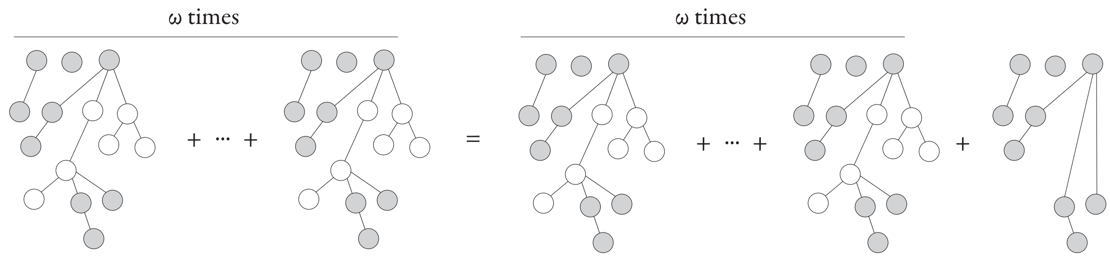


Fig. 3. The identity $\omega(vuh) = \omega(vuh) + vh$, with the white nodes belonging to u.

Proposition 21 Identity (1) is equivalent to $\mathcal{J}$-triviality of V, and the identity

$$vh + \omega(vuh) = \omega(vuh) = \omega(vuh) + vh \tag{5}$$

Proof
One implication is obvious: both $\mathcal{J}$-triviality and (5) follow from (1). For the other implication, we need to show that if $v \preceq u$, then

$$u^\omega v = u^\omega = vu^\omega \ .$$

We will only show the first equality, the other is done the same way. By unraveling the definition of $v \preceq u$, there is a morphism

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

and two contexts $p \preceq q$ over A such that $\alpha(p) = v$ and $\alpha(q) = u$. If p can be decomposed as p_1p_2, then we can reason separately for p_1 and p_2:

$$\alpha(q)^\omega \cdot \alpha(p_1) \cdot \alpha(p_2) = \alpha(q)^\omega \cdot \alpha(p_2) = \alpha(q)^\omega \ .$$

If p consists of single node with a hole below, then we have $q = q_0pq_1$ for some two contexts q_0, q_1, and therefore also $u = u_0vu_1$ for some u_0, u_1. The result then follows by:

$$u^\omega v = (u_0vu_1)^\omega v = (u_0vu_1)^\omega u_0v = (u_0vu_1)^\omega = u^\omega \ .$$

In the above, we used twice the assumption on J-triviality of V. Once when adding u_0 to the ω-power, and then when removing u_0v from after the ω-power.

The interesting case is when $p = \square + s$ for some tree s. In this case, the forest q can be decomposed as $q_1(\square + t)q_2$, with $s \preceq t$. We have

$$u^\omega v = \alpha(q_1(\square + t)q_2)^\omega \alpha(\square + s) \ .$$

Thanks to J-triviality, the above can be rewritten as

$$\alpha(q_1(\square + t)q_2)^\omega(\alpha(\square + t))^\omega \alpha(\square + s) =$$
$$\alpha(q_1(\square + t)q_2)^\omega(\square + \alpha(s) + \omega \cdot \alpha(t)) \ .$$

It is therefore sufficient to show that $s \preceq t$ implies

$$\omega\alpha(t) = \alpha(s) + \omega\alpha(t)$$

The proof of the above equality is by induction on the number of nodes that need to be removed from t to get s. The base case $s = t$ follows by aperiodicity of H, which follows by aperiodicity of V, itself a consequence of J-triviality. Consider now the case when t is bigger than s. In particular, we can remove a node from t and still have s as a piece. In other words, there is a decomposition $t = q_0q_1t'$ such that $s \preceq q_0t'$. Applying the induction assumption, we get

$$\omega\alpha(q_0t') = \alpha(s) + \omega\alpha(q_0t') \ .$$

Furthermore, applying equation (5), we get

$$\omega\alpha(t) = \alpha(q_0t') + \omega\alpha(t) = \omega\alpha(q_0t') + \omega\alpha(t) \ .$$

Combining the two equalities, we get the desired result. □

V. Commutative languages

In this section we talk about forest languages that are commutative, i.e. closed under rearrainging siblings.

A forest t' is called a *reordering* of a forest t if it is obtained from t by rearranging the order of siblings. In other words, reordering is the least equivalence relation on trees that identifies each two forests $p(s+t)$ and $p(t+s)$. A forest language is called *commutative* if it is closed under reordering. A forest language is *commutative* if and only if its syntactic algebra satisfies the identitiy

$$g + h = h + g \ .$$

We say a forest s is a *commutative piece* of t, if s is a piece of some reordering of t. A forest language L is called *commutative piecewise testable* if for some $n \in \mathbb{N}$, membership $t \in L$ depends only on the set of commutative pieces of t that have n nodes. This definition also has a counterpart in logic, by removing the lexicographic order from the signature:

Proposition 22 A forest language is commutative piecewise testable iff it is definable by a Boolean combination of $\Sigma_1(<)$ formulas.

If a language is commutative piecewise testable, then it is clearly commutative and piecewise testable (in the more powerful, noncommutative, sense). Below we show that the converse implication is also true:

Theorem 23 *A forest language is commutative piecewise testable if and only if it is commutative and piecewise testable.*

Corollary 24 It is decidable if a forest language is definable by a Boolean combination of $\Sigma_1(<)$ formulas.

The theorem above follows immediately from:

Lemma 25 Let $n \in \mathbb{N}$. For k sufficiently large, if two forests have the same commutative pieces of size at most k, then they can be both reordered so that they have the same pieces of size at most n.

Proof
Let $P(t)$ be the set of pieces of t that have size at most n. By a pumping argument as in Lemma 10, there is some k such that any forest t has a piece $s \preceq t$ of size at most k with $P(s) = P(t)$. Let now s_1, s_2 be two forests with the same commutative pieces of size k. For $i = 1, 2$, consider the families

$$\mathcal{P}_i = \{P(s'_i) : s'_i \text{ is a reordering of } s_1\} .$$

To prove the lemma, we need to show that the families $\mathcal{P}_1$ and $\mathcal{P}_2$ share a common element, itself a set of pieces. To this end, we show that for any $X \in \mathcal{P}_1$, there is some $Y \in \mathcal{P}_2$ with $X \subseteq Y$, and vice versa; in particular, the families share the same maximal elements. Let then $X = P(s'_1) \in \mathcal{P}_1$. By choice of k, the forest s'_1—and therefore also s_1—has a commutative piece t of size at most k with $P(t) = X$. By assumption, the forest t is also a commutative piece of some reordering s'_2 of s_2, and therefore $X \subseteq P(s'_2) \in \mathcal{P}_2$. □

VI. Tree languages

Theorem 2 characterizes piecewise testable *forest* languages, and in fact the algebraic theory used here works best when forests, rather than trees, are treated as the fundamental object. Traditionally, though, interest has focused on trees rather than forests. Thus we want to give a decidable characterization of the piecewise testable tree languages: that is, the sets of *trees* that result when we interpret a boolean combination of Σ_1 sentences in trees over a finite alphabet.

For certain logics, like first-order logic over the descendant relation, or first-order logic over successor, one can write a sentence that says "this forest is a tree", and thus there is no need to treat tree and forest languages separately. For piecewise testability, we need to do something more, since the set of all trees over a finite alphabet A is not piecewise testable as a forest language.

We define a *tree piecewise testable language* over a finite alphabet A to be the intersection of a piecewise testable forest language with the set of all trees over A. (This is preferable to defining a tree piecewise testable language to be a tree language that is piecewise testable as a forest language, the latter definition would give exactly the finite tree languages.) We will obtain our decidability result by an entirely general method for translating algebraic characterizations of classes of forest languages to characterizations of the corresponding classes of tree languages. First, suppose

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

is a forest algebra morphism that maps onto (H, V). We define an equivalence relation on H_A: We write $s \sim t$ if for all contexts p such that ps and pt are both trees, we have $\alpha(ps) = \alpha(pt)$. It is clear that if $s \sim t$ then for any context q, $qs \sim qt$. Thus $\sim$ defines a forest algebra congruence on A^{Δ}. Let

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

be the projection morphism onto the quotient by this congruence. Obviously α' factors through α; that is, $\alpha' = \beta\alpha$ for some morphism β from (H, V) onto (H', V'). We call α' the *tree reduction* of α.

Let $\mathbf{F}$ be a family of forest languages over A and let $\mathcal{F}$ be a family of surjective forest algebra morphisms with domain A^{Δ} that characterizes $\mathbf{F}$ in the following sense: A forest language L belongs to $\mathbf{F}$ if and only if L is recognized by some morphism in $\mathcal{F}$. Observe that if $\alpha : A^{\Delta} \to (H, V)$ belongs to such a family $\mathcal{F}$, and if $\beta : (H, V) \to (H'V')$ is any surjective morphism, then every language recognized by $\beta\alpha$ also belongs to $\mathbf{F}$. Thus we will suppose that $\mathcal{F}$ is closed in this way: if α belongs to $\mathcal{F}$, then $\beta\alpha$ belongs to $\mathcal{F}$.

Theorem 26 *Let $\mathbf{F}$ and $\mathcal{F}$ be as above, and let $L \subseteq H_A$ be a set of trees. Then there is a forest language $K \in \mathbf{F}$ such that L consists of all the trees in K if and only if the tree reduction of the syntactic morphism α_L of L belongs to $\mathcal{F}$.*

Proof: We merely sketch the proof: If such a forest language K exists, then it is straightforward to verify that the tree reduction α' of α_L factors through α_K, and thus belongs to $\mathcal{F}$. Conversely, suppose that α' belongs to $\mathcal{F}$. Let T be the set of all trees over A. The syntactic morphism α_T of this language can assign three possible values to a forest: 0 for empty the empty forest, x for a tree, and $x + x$ for a forest with at least two trees. The key property is that α_L factors through $\alpha_T \times \alpha'$, and thus L is recognized by $\alpha_T \times \alpha'$. Since L consists entirely of trees, this implies that there exists $X \subseteq H'$ such that

$$L = (\alpha_T \times \alpha')^{-1}(\{x\} \times X) = T \cap (\alpha')^{-1}(X) = T \cap K,$$

where $K \in \mathbf{F}$. ■

As a result we have:

Corollary 27 It is decidable if a regular tree language is tree piecewise testable.

Proof: Let $\mathbf{F}$ be the family of piecewise testable forest languages over A, and let $\mathcal{F}$ be the family of morphisms from A^{Δ} onto finite forest algebras that satisfy the identities of Theorem 2. Since this family of algebras is closed under quotient, $\mathbf{F}$ and $\mathcal{F}$ satisfy the hypotheses of Theorem 26.

Consequently, a regular tree language L is tree piecewise testable if and only if the tree reduction of α_L belongs to $\mathcal{F}$. It remains to show that we can effectively compute the image of the tree reduction given α_L. Since the tree reduction factors through α_L, this amounts to deciding which pairs of elements of the syntactic forest algebra are identified under the reduction, which we can do as long as we know which elements are images under α_L of trees. It is easy to see that if an element of H_L is the image of a tree, then it is the image of a tree of depth at most $|V_L|$ in which each node has at most $|H_L|$ children, so we can effectively decide this as well. ■

VII. Closest common ancestor

According to our definition of piece, $t = d(a+b)$ is a piece of the forest $s = dc(a+b)$. In this section we consider a notion of piece which does not allow removing the closest common ancestor of two nodes, in particular removing the node c in the example above. The logical correspondent of this notion is a signature where the closest common ancestor (a three argument predicate) is added.

Given a forest s and three nodes x, y, z of s we say that z is the *closest common ancestor* of x and y if z is an ancestor of both x and y and all other nodes of s with this property are ancestors of z. We now say that a forest s *is a cca-piece* of a forest t if there is an injective mapping from nodes of s to nodes of t that preserves the ancestor order and the closest common ancestor relationship. An equivalent definition is that the cca-piece relation is the reflexive transitive closure of the relation

$$\{(pt, pat): \ t \text{ is a tree or empty}\}$$

A forest language L is called *cca-piecewise testable* if membership in L depends only on the set of cca-pieces of t up to some fixed size n.

As before, every cca-piecewise testable language is regular. The analogue of Proposition 1 holds as well. The cca-piecewise testable languages are exactly those definable by boolean combinations of Σ_1-sentences over a signature that includes predicates for lexicographic order, the ancestor order and the closest common ancestor relation.

A first remark is that there are more cca-piecewise testable languages than there are piecewise testable ones. Hence the equations that characterize piecewise testable languages are no longer valid. In particular, in the syntactic algebra of a cca-piecewise testable language, the context monoid V may no longer be J-trivial. To see this consider the language L of forests over $\{a, b, c\}$ that contain the cca-piece $a(b+c)$, this is the language "some a is the closest common ancestor of some b and c". Then the context $p = (ab)^\omega\Box$ is not the same as the context $q = (ba)^\omega\Box$ as $p(b+c) \notin L$ while $q(b+c) \in L$. Note however that p and q satisfy the equivalence $pt \in L$ iff $qt \in L$ for all *trees* t. The characterization below is a generalization of this idea.

With the closest common ancestor, also the algebraic situation is more complicated: the cca-piecewise testable languages no longer form a variety of languages and cca-piecewise testability of a forest language L is not determined by the syntactic forest algebra alone. Indeed it is not difficult to see that they are not closed under inverse images of homomorphisms that are either i) erasing (the image of some $a\Box$ is the empty context $\Box$) or, ii) $a\Box$ is sent to $u + f$ for some context u and some non-empty forest f. However cca-piecewise testable languages satisfy all the other properties of varieties of languages and in particular they are closed under the inverse image of homomorphisms that are "tree-preserving", i.e., the image of $a\Box$ is a tree-context u for all a. Therefore, to obtain a characterization of cca-piecewise testable languages, it is necessary to look at the *syntactic morphism* $\alpha_L : A^\Delta \to (H_L, V_L)$ that maps each (h, v) to its $\sim_L$-class, and not just the algebra found in the image of this morphism.

We call a context a *tree context* if it is nonempty and has one node that is the ancestor of all other nodes, including the hole.

We extend the cca-piece relation to elements of a forest algebra (H, V) as follows: we write $v \preceq w$ if there are contexts $p \preceq q$ that are mapped to v and w respectively by the morphism α. There is a subtle difference here: the $\preceq$ relation on V depends on the particular syntactic morphism α_L! By abuse of notation, the elements of V_L that are image by the syntactic morphism α_L of a tree context are also called tree contexts. Similarly, the elements of H_L that are images of a tree are also called trees (it is possible for an element to be an image of both a tree and a non-tree, but it is still called a tree here). Note that the notions of tree and of tree context for the elements of H_l and V_L are relative to α_L.

Theorem 28 *A forest language L is cca-piecewise testable if and only if its syntactic algebra satisfies the following identities:*

$$u^\omega h = u^\omega v h = v u^\omega h \tag{6}$$

whenever h is a tree or empty, and $v \preceq u$ are tree-contexts, and

$$\omega(h) = \omega(h) + g = g + \omega(h) \qquad \text{if } g \preceq h \tag{7}$$

Because of the finiteness of H_L and V_L, one can effectively decide whether an element of one of these monoids is the image of a tree context or a tree. Whether or not $v \preceq u$ or $g \preceq h$ holds can be decided in polynomial time using an algorithm as in Corollary 6. Thus the theorem yields a decidable characterization of the cca-piecewise testable languages. The proof follows the same outline as that of the proof of Theorem 2, but the details are somewhat complicated. We omit it for reasons of space, the proof can be found in the appendix. In the appendix, we also include an equivalent set of identities, where the conditions $v \preceq u$ and $g \preceq h$ are not used.

References

[1] 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.

[2] M. Bojańczyk. Two-way unary temporal logic over trees. In *Logic in Computer Science*, pages 121–130, 2007.

[3] M. Bojańczyk and I. Walukiewicz. Characterizing EF and EX tree logics. *Theoretical Computer Science*, 358(2-3):255–273, 2006.

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

[5] I. Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222, 1975.

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

[7] 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.

[8] 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.

[9] 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.

APPENDIX

This appendix is devoted to the proof of Theorem 28. We first recall the statement of the theorem:

Theorem 28 *A forest language L is cca-piecewise testable if and only if its syntactic algebra satisfies the following identities:*

$$u^\omega h = u^\omega v h = v u^\omega h \tag{6}$$

whenever h is a tree or empty, and $v \preceq u$ are tree-contexts, and

$$\omega(h) = \omega(h) + g = g + \omega(h) \qquad \text{if } g \preceq h \tag{7}$$

The proof strategy is essentially the same as for Theorem 2; but there is some tedium due to the closest common ancestor.

The proof that (6) and (7) are correct is the same as Section IV-B. The only difference is that instead of Fact 9, we use the following.

Fact 29 If r is any context, $p \preceq q$ are tree contexts, and t is a tree or empty, then $rpt \preceq rqt$.

We now turn to the completeness proof in Theorem 28. The proof is very similar to the one of the previous section, with some subtle differences.

As before, we fix a language L whose syntactic forest tree algebra (V, H) satisfies all the equations of Proposition 35. We write α for the syntactic morphism.

We now write $s \sim_n t$ if the two forests s, t have the same cca-pieces of size n. Likewise for contexts.

The main step is to show the following proposition.

Proposition 30 For n sufficiently large, if t is a tree or empty, then $pat \sim_n pt$ entails $\alpha(pat) = \alpha(pt)$.

Theorem 28 follows from the above proposition in the same way as Theorem 2 follows from Proposition 11 in the previous section. The reason why we assume that t is either a tree or empty is as follows: if s is an cca-piece of s', then s can be obtained from s' by iterating one of the following two operations: removing a leaf, or removing a node which has only one successor. We thus concentrate on showing Proposition 30.

We will now redefine the concept of fractal for our new, closest common ancestor setting. The key change is in the concept of a vah-decomposition. We change the notion of $x \in X$ being a vah-decomposition as follows: all conditions of the old definition hold, but a new condition is added: either x has no descendants in X; or there is a minimal element of X that has x as a proper ancestor. In other words, the part of $s[X]$ that corresponds to h is either empty, or is a tree. In particular, $s[X \setminus \{x\}]$ is a closest common ancestor piece of $s[X]$; which is the key property required below. From now on, when referring to a vah-decomposition, we use the new definition.

The concept of a fractal $x_1 \in X_1, \dots, x_k \in X_k$ inside s is also redefined to reflect the new definition: for each i, $x_i \in X_i$ is a vah-decomposition of s in the new sense.

Using the same technique as without closest common ancestors, we show

Lemma 31 Let $k \in \mathbb{N}$. For n sufficiently large, if t is a tree or empty, then $pat \sim_n pt$ entails the existence of a fractal of length k.

A fractal $x_1 \in X_1 \ \cdots \ x_k \in X_k$ inside s is called *cca-tame* if s can be decomposed as $s = qq_1 \cdots q_k s'$ (or $s = qq_k \cdots q_1 s'$) such that $x_1 \in q_1, \cdots, x_k \in q_k$ and such that either:

- Each q_i is a tree context whose root node belongs to $X_i \setminus \{x_i\}$.
- Each q_i is a context of the form $\square + t_i$, with t_i a forest.

Lemma 32 Let $k \in \mathbb{N}$. For n sufficiently large, if there is a fractal of length k, then there is a cca-tame fractal of length k.

Proof
The proof is essentially the same as for Lemma 11; only this time we need to be more careful to satisfy the more stringent requirements in a cca-tame fractal.

Let $m = 2k + 2$. Using the same technique as in the case without closest common ancestor, if n is large enough then we may extract a subfractal of length m where either:

- All the nodes $x_1, \dots, x_m$ have the same closest common ancestor. In this case, we can extract a cca-tame subfractal, where each context is of the form $\square + t_i$.
- The set $Y = \{y : y \text{ is a closest common ancestor of some } x_i, x_j\}$ is a chain. Using the same sort of reasoning as in Lemma 16, we may assume that Y enumerated as $y_1 < \cdots < y_{m-1}$, and that between y_i and y_{i+1} we can find the node x_i. (There is a second case, where the nodes $y_1, \dots, y_m$ are ordered the other way: with y_{i+1} an ancestor of y_i. This case is treated analogously.) In particular, y_i is the closest common ancestor of x_i and any of the nodes $x_{i+1}, \dots, x_m$. Since X_{i+1} contains both x_i and x_{i+1}, each node y_i belongs to the set X_{i+1}. This allows us to get the cca-tame fractal. We use $x_2 \in X_2, x_4 \in X_4, \dots, x_{2k} \in X_{2k}$ as the fractal (recall that $m = 2k$); while the decomposition $qq_1 \dots q_k s'$ is chosen so that q_i has its root in y_{2i-1}, and its hole in y_{2i+1}.

$\square$

We will now take a cca-tame fractal, and show that $\alpha(pat) = \alpha(pt)$.

Recall the definition of u_{ijl} and w_{ijl} as the image under α of the context obtained from $q_{i+1} \cdots q_j$ by restricting s to X_l and $X_l \setminus \{x_l\}$, respectively. Note that the latter is a cca-piece of the former, by the new definition of fractals. This way, we get:

$$w_{ijl} \preceq u_{ij(l+1)} \qquad u_{ijl} \preceq u_{ij(l+1)} \tag{8}$$

if the q_i are tree-contexts then u_{ijl}, w_{ijl} are a tree-contexts (9)

The definition of monochromaticity is the same as in the previous section and Ramsey's Theorem gives.

Lemma 33 If there is a cca-tame fractal of sufficiently large size, then there is a monochromatic cca-tame fractal of size $m = \omega + 2$.

We conclude by showing the following result:

Lemma 34 If there is a monochromatic cca-tame fractal of size $\omega + 2$, then $vah = vh$.

Proof
Fix a monochromatic cca-tame fractal of size $m \geq \omega + 1$. Since $x_m \in X_m$ is a vah decomposition, the statement of the lemma follows once we show that α assigns the same type to the forest $s[X_m]$ and $s[X_m \setminus \{x_m\}]$.

Recall the definition of u_{ijl} and w_{ijl} in the previous section. Recall that the type of the forest $s[X_m]$ can be decomposed as

$$\alpha(s[X_m]) = \alpha(q[X_m]) \cdot u_{12m} \cdot u_{23m} \cdots u_{(m-1)mm} \cdot \alpha(s'[X_m])$$

The type of $s[X_k \setminus \{x_k\}]$ is decomposed the same way, only $u_{(k-1)kk}$ is replaced by $w_{(k-1)kk}$. Therefore, the lemma will follow if

$$u_{12m} \cdot u_{23m} \cdots u_{(m-1)mm} = u_{12m} \cdot u_{23m} \cdots w_{(m-1)mm} \ .$$

Since the fractal is monochromatic, and since m is greater than ω, the above becomes

$$u^{\omega}_{12m} \cdot u_{(m-1)mm} = u^{\omega}_{12m} \cdot w_{(m-1)mm} \ .$$

By (8) and monochromaticity, we have

$$w_{(m-1)mm} \ , \ u_{(m-1)mm} \quad \preceq \quad u_{(m-1)m(m+1)} = u_{12m} \ ,$$

We now have two cases. If all the q_i are tree-contexts, we conclude using equation (6) which can be applied because of the above and (9). If all the q_i are contexts of the form $\square + f_i$, we conclude using equation (7) which can be applied because of (8). $\square$

A. An equivalent set of equations.

In this section, we give a set of identities that is equivalent to the one used in Theorem 28. The rationale is the same as in Proposition 21: we want to avoid the use of $v \preceq w$ in the identities.

Proposition 35 The conditions on the syntactic morphism stated in Theorem 28 are equivalent to the following equalities:

$$(uv)^{\omega} h = (uv)^{\omega} uh \tag{10}$$

whenever h is a tree or empty, and

$$(uv)^{\omega} = v(uv)^{\omega} \tag{11}$$

whenever u and v are tree context elements, and

$$(u(\square + vwh))^{\omega} g = (u(\square + vwh))^{\omega} u(\square + vh) g \tag{12}$$

whenever u is a tree context and g, h are trees or empty.

The rest of Section A is devoted to showing the above proposition.

It is immediate to see that equation (6) implies equation (11) and that equation (6) implies equation (12). We now show that equations (6) and (7) implies equation (10). Let u and v be two contexts and h be a tree. We want to show that $(uv)^{\omega} h = (uv)^{\omega} uh$.

We consider several cases.

• In the first case we assume that $u = u_1 u_2$ for some tree-context u_2. In that case we have, using aperiodicity:

$$(uv)^{\omega} h = (u_1 u_2 v)^{\omega} h = (u_1 u_2 v)(u_1 u_2 v u_1 u_2 v)^{\omega} h$$

Therefore,

$$(uv)^{\omega} h = u_1 (u_2 v u_1 u_2 v u_1)^{\omega} u_2 v h$$

Notice now that $u_2 v \preceq u_2 v u_1 u_2 v u_1$ and that $u_2 v u_1 u_2 \preceq u_2 v u_1 u_2 v u_1$. As u_2 is a tree-context, all the contexts involved are tree-contexts and we can use equation (6) twice and replace $u_2 v$ by $u_2 v u_1 u_2$. This yields:

$$(uv)^{\omega} h = u_1 (u_2 v u_1 u_2 v u_1)^{\omega} u_2 v u_1 u_2 h$$

And we have

$$(uv)^{\omega} h = (u_1 u_2 v u_1 u_2 v)^{\omega} u_1 u_2 v u_1 u_2 h$$

By aperiodicity we get the desired result:

$$(uv)^{\omega} h = (uvuv)^{\omega} uvuh = (uv)^{\omega} uh$$

• The second case assume that $v = v_1 v_2$ for some tree-context v_2 and is treated similarly.

$$(uv)^{\omega} h = (uv_1 v_2)^{\omega} h = (uv_1 v_2)(uv_1 v_2 u)^{\omega} h$$

Therefore,

$$(uv)^{\omega} h = uv_1 (v_2 u v_1)^{\omega} v_2 h$$

Notice now that $v_2 \preceq v_2 u v_1$ and that $v_2 u \preceq v_2 u v_1$. As v_2 is a tree-context, all the contexts involved are tree-contexts and we can use equation (6) twice and replace v_2 by $v_2 u$. This yields:

$$(uv)^{\omega} h = uv_1 (v_2 u v_1)^{\omega} v_2 u h$$

And we have

$$(uv)^{\omega} h = (uv)^{\omega} uvuh = (uv)^{\omega} uh$$

• When none of the above cases works, we must have $u = \square + f$ and $v = \square + g$. In that case we have $(uv)^{\omega} h = \omega(f + g) + h$, and we conclude using equation (7) as $f \preceq (f + g)$.

We now consider the converse implication in Proposition 35. Assume that equations (10)-(12) hold. We only need to show that equations (6) and (7) are satisfied, since commutativity of H is in both sets of equations.

We first show the following lemma:

Lemma 36 If u is a tree context, v, w, w' are (not necessarily tree) contexts with $w' \preceq w$, and g, h are either a tree or empty, then the following identity holds

$$(u(\square + vwh))^\omega g = (u(\square + vwh))^\omega (u(\square + vw'h)g \quad (13)$$

Note that the identity (7) is a direct consequence of the above, by taking u, v to be the empty context, and g, h to be the empty tree. We will also use the above lemma to show (6), but this will require some more work.

Proof

The proof is by induction on the number of steps used to derive $w' \preceq w$.

- Consider first the case when w, w' can be decomposed as

  $$w = w_1 w_2 \quad w' = w_1' w_2' \qquad w_1' \preceq w_1, w_2' \preceq w_2$$

  Two applications of the induction assumption give us:

  $$(u(\square + vw_1w_2h))^\omega g = (u(\square + vw_1w_2h))^\omega (u(\square + vw_1w_2'h)g$$
  $$(u(\square + vw_1w_2'h))^\omega g = (u(\square + vw_1w_2'h))^\omega (u(\square + vw_1'w_2'h)g$$

  As u is a tree context, these can then be combined to give the desired result.
- Consider now the case when w, w' can be decomposed as

  $$w = w_1 w_2 w_3 \qquad w' = w_1' w_3' \qquad w_1' \preceq w_1, w_3' \preceq w_3'$$

  with w_3' a tree context or empty. We first use the induction assumption to get

  $$(u(\square + vw_1w_2w_3h))^\omega g = (u(\square + vw_1w_2w_3))^\omega (u(\square + vw_1w_2w_3'h)g$$

  By applying the equation (12), we get

  $$(u(\square + vw_1w_2w_3'h))^\omega g = (u(\square + vw_1w_2w_3'h))^\omega (u(\square + vw_1w_3'h)g$$

  Note that it is important here that $w_3'h$ is going to be either a tree context or empty. Finally, we apply once again the induction assumption to get

  $$(u(\square + vw_1w_3'h))^\omega g = (u(\square + vw_1w_3'))^\omega (u(\square + vw_1'w_3'h)g$$

  Again, as u is a tree context those three identities can be combined sequentially and give us the desired result.
- Finally, consider the case when w, w' can be decomposed as

  $$w = \square + w_1 0 \qquad w' = \square + w_1' 0 \qquad w_1' \preceq w_1$$

  In this case, the identity becomes:

  $$(u(\square + v(h + w_1 0))^\omega g = (u(\square + v(h + w_1 0)))^\omega (u(\square + v(h + w_1' 0))g$$

  The result easily follows by induction assumption, by collapsing $v(h + \square)$ into v.

$\square$

We now derive the first part of equation (6). Let u, v be tree contexts such that $v \preceq u$, and let h be a tree. We will show that $u^\omega h = u^\omega vh$. If $v = v_1 v_2$ where both v_1 and v_2 are tree-contexts then we consider v_2 first and v_1 next:

$$u^\omega h = u^\omega v_2 h = u^\omega v_1 v_2 h \ .$$

It is important here that $v_2 h$ is a tree.

Therefore it is enough to consider the case where v is of the form $a(\square + f)$ for some letter a and some forest f. From $v \preceq u$ we get $u = u_1 a(\square + g) u_2$ where u_1 and u_2 are tree-contexts and $f \preceq g$. Then we have

$$u^\omega h = u^\omega a(\square + g)h = u^\omega a(\square + g)a(\square + g)h = \cdots = u^\omega (a(\square + g))^\omega h \ .$$

It will therefore be enough to show

$$(a(\square + g))^\omega h = (a(\square + g))^\omega a(\square + f)h$$

for $f \preceq g$. This, however, is a consequence of (13).

The second part of equation (6) is shown the same way. This time however, we need a symmetric variant of (13), which is shown the same way:

$$(u(\square + vwh))^\omega = (u(\square + vw'h)(u(\square + vwh))^\omega \ .$$