Two-way unary temporal logic over trees

Mikołaj Bojańczyk, Warsaw University

***Abstract*— We consider a temporal logic $\mathsf{EF} + \mathsf{F}^{-1}$ for unranked, unordered finite trees. The logic has two operators: $\mathsf{EF}\varphi$, which says "in some proper descendant φ holds", and $\mathsf{F}^{-1}\varphi$, which says "in some proper ancestor φ holds". We present an algorithm for deciding if a regular language of unranked finite trees can be expressed in $\mathsf{EF} + \mathsf{F}^{-1}$. The algorithm uses a characterization expressed in terms of forest algebras.**

I. Introduction

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

This type of problem has been successfully studied for word languages. Arguably best known is the result of McNaughton, Papert and Schützenberger [10], [7], which says that the following three conditions on a regular word language L are equivalent: a) L can be defined in first-order logic; b) L can be defined using a star-free expression; and c) the syntactic semigroup of L does not contain a non-trivial group. Since condition c) can be effectively tested, the above theorem gives a decidable characterization of first-order logic. This result demonstrates two important features of work in this field: a decidable characterization not only gives us a better understanding of the logic in question, but it often reveals unexpected connections with algebraic concepts. During several decades of research, decidable characterizations have been found for fragments of first-order logic with restricted quantification and a large group of temporal logics, see [8] and [13] for references.

For trees, however, much less is known. No decidable characterization has been found for what is probably the most important tree logic, first-order logic with the descendant relation, despite some attempts [9], [5], [2]. Similarly open are chain logic [12] and the temporal logics CTL and PDL. However, there has been some recent progress. In [4], decidable characterizations were presented for the temporal logics EF and $\mathsf{EX} + \mathsf{EF}$; while Benedikt and Segoufin [1] characterized tree languages definable in first-order logic with the successor relation (but without the descendant relation).

In this paper, we continue the line of research started in [4], by focusing on a temporal logic for trees. We consider a logic called $\mathsf{EF}+\mathsf{F}^{-1}$. This logic has two operators: $\mathsf{EF}\varphi$, which says "in some proper descendant φ holds", and $\mathsf{F}^{-1}\varphi$, which says "in some proper ancestor φ holds". Thanks to the backward modality, $\mathsf{EF} + \mathsf{F}^{-1}$ is more expressive than EF alone. For instance, the formula

$$\mathsf{EF}(a \wedge \neg\mathsf{F}^{-1}\neg b)$$

is true in a tree where some node has label a, but all of its ancestors have label b. This is a property reminiscent of CTL, and cannot be expressed by only using EF.

The main result in this paper is Theorem V.1, which gives a decidable characterization of languages definable in $\mathsf{EF}+\mathsf{F}^{-1}$. Before we present this result, in Section II we try to justify the choice of the logic $\mathsf{EF} + \mathsf{F}^{-1}$. In Section III we present the algebraic formalism that will be used in the proofs. The rest of the paper is devoted to proving the main result.

I would like to thank Luc Segoufin. We spent a lot of time together trying to understand the expressive power of $\mathsf{EF} + \mathsf{F}^{-1}$; without his input this paper would not have been possible.

II. Why two-way unary temporal logic

There are two reasons to consider $\mathsf{EF}+\mathsf{F}^{-1}$. The first reason is that, over words, this logic corresponds to an important and well-studied class of regular languages. The second reason is that over trees, the logic is related to XML. We go over these reasons in Sections II-A and II-B respectively.

A. The word analogy

There is a very robust class of regular word languages that has several equivalent descriptions:

1) Word languages that can be defined in the temporal logic $\mathsf{F} + \mathsf{F}^{-1}$. Here $\mathsf{F}\varphi$ means "in some future position φ" and $\mathsf{F}^{-1}\varphi$ means "in some past position φ".
2) Word languages that can be defined by a first-order formula with two variables and the left-to-right ordering of positions (but without the successor relation).
3) Word languages that can be defined by a first-order formula (with many variables, the left-to-right ordering, but without the successor) with a $\forall^*\exists^*$ quantifier prefix, and also by one with an $\exists^*\forall^*$ quantifier prefix.
4) Word languages whose syntactic semigroup belongs to the semigroup variety DA.
5) Languages described by finite disjoint unions of unambiguous products (a form of regular expression).
6) Word languages that can be recognized by "turtle automata", a type of deterministic two-way word automaton.

An important corollary of property 4 is that membership of a regular language in the above class is decidable: it suffices to

*Supported by Polish goverment grant no. N206 008 32/0810.

check if the syntactic semigroup of the language satisfies the DA equation.

Some of the above classes have fairly natural tree counterparts, some don't. (We consider unranked, unordered finite trees here.)

The three logically defined classes – items 1, 2 and 3 – can be extended to trees as follows. A natural counterpart of class 1 is the logic $\mathsf{EF} + \mathsf{F}^{-1}$ considered in this paper. The classes 2 and 3 can be used to define tree languages if the order is interpreted as the ancestor/descendant ordering of tree nodes. (One could also consider variants where two partial orders of nodes are available instead of one: the descendant/ancestor order and also the left-to-right ordering of siblings. We keep to the simpler case, where siblings are unordered.)

We will not talk about classes 5 and 6: it is not clear what unambiguous expressions are for trees, nor do we have a notion of turtle automata. We will come back to the algebraic description in item 4 later on in the paper.

The logically defined classes diverge for trees:

- Two-variable logic is strictly stronger than the temporal logic. The translation from temporal to two-variable logic is fairly obvious. For the converse, the problem is that $x \not\leq y \wedge y \not\leq x$ cannot be expressed in the temporal logic. For instance, the language: "there are two a's" can be defined by a two-variable formula, but cannot be defined in the temporal logic. This is because the temporal logic is bisimulation invariant, and cannot see the difference between one child with a and two children with a. (Note however, that the language "there are two a's below some b" cannot be defined in two-variable logic.)
- As we will show at the end of this paper, the intersection of $\forall^*\exists^*$ and $\exists^*\forall^*$ is incomparable with both the two-variable and the temporal logic.

Why do we consider the temporal logic in the paper and not the other two logics? The short answer is: because we can. Giving decidable characterizations for the other two logics seems to be more complicated, and we leave it as future work.

B. XPath

XPath is a formalism used to describe paths and nodes in unranked trees. There is a strong connection between XPath and two-variable logics

A path is seen as a binary relation $P(x, y)$, which connects its source x with its target y. The basic idea in XPath is that one starts with atomic paths, called axes, such as "x is a descendant of y", or "x is a successor of y", and then constructs longer paths using mechanisms such as concatenation or iteration. Marx [6] has shown that a fragment of XPath called Core XPath (basically, iteration is not allowed) has exactly the same expressive power as two-variable first-order logic. Note however, that the axes considered by Marx include successor and next-child, which go beyond the fragments considered in this paper. When the only axis allowed is "descendant", Core XPath has exactly the same power as "our" logic $\mathsf{EF} + \mathsf{F}^{-1}$. A decidable characterization for Core XPath with the other axes is left as future work.

III. Basic definitions

A. Trees and forests

We work with unranked finite labeled trees. We assume that an alphabet (A, B) contains two types of labels: one set of labels A that can be used in the leaves, and another set of labels B that can be used in inner nodes (i.e. not leaves). This division is convenient for the algebraic framework we use in general, and for the induction proof in this paper in particular. *Trees* are defined as follows: every leaf label $a \in A$ is a tree; if $t_1, \ldots, t_n$ are trees and $b \in B$ then $b(t_1 + \cdots + t_n)$ is a tree. We denote trees using letters s, t. A *forest* is a sequence of trees; we denote forests using vector notation $\bar{s}, \bar{t}$. As above, we concatenate forests using $+$. In particular every forest is of the form $\bar{t} = t_1 + \cdots + t_n$, for some trees $t_1, \ldots, t_n$. We do not allow empty forests, so $n \geq 1$. A *context* is a forest where exactly one leaf is labeled by a special label $*$; this leaf is interpreted as a hole. We denote contexts by p, q. The *main path* in a context consists of the ancestors of the hole. A forest $\bar{t}$ can be substituted in place of the hole of a context p, the resulting forest is denoted $p(\bar{t})$, or sometimes $p\bar{t}$. There is a natural composition operation on contexts: the context pq is the unique context such that $(pq)\bar{t} = p(q\bar{t})$ holds for all forests $\bar{t}$. We do allow the empty context, denoted by $*$; this is the context where the only node in the context is the hole $*$. The empty context satisfies $*\bar{t} = \bar{t}$. Nodes of trees, forests and contexts are defined the usual way. We will mostly order nodes using the descendant order.

The reader will notice that the trees and forests we defined are sibling-ordered (i.e. $s+t$ is not the same as $t+s$). However, properties definable in our logic $\mathsf{EF} + \mathsf{F}^{-1}$ are going to be invariant under this order.

B. The logic

The logic $\mathsf{EF} + \mathsf{F}^{-1}$ is defined as follows:

- Every label – both inner node label and leaf label – is a formula; this formula holds in nodes with that label.
- Formulas are closed under boolean combinations, including negation.
- If φ is a formula, then $\mathsf{EF}\varphi$ is also a formula; it is true in a node x if there is some proper descendant $y > x$ where φ is true. Likewise for $\mathsf{F}^{-1}\varphi$, but this time y must be a proper ancestor $y < x$.

C. Forest algebra

To represent languages of trees, we will be using forest algebra. We feel that using forest algebra instead of automata simplifies a lot of the combinatorics used in our characterization. Furthermore, when using forest algebra, the key properties from Theorem V.1 can be stated in terms of equations.

Here we only sketch out the definitions and basic properties; the reader is referred to [3] for more details. The algebras described in [3] differ slightly from those used here—mainly in that we allow empty contexts—but all the results carry over into this setting.

A forest algebra is to regular languages of unranked trees as a semigroup is to regular languages of words. Formally, a forest algebra is an algebra with two sorts (H, V), along with some operations that satisfy a number axioms. While defining the operations and axioms, we will illustrate them on an important example, called the free forest algebra, where H is the set of all nonempty forests, and V is the set of all, possibly empty, contexts.

The operations in forest algebra are defined below. (Elements of H will be denoted by h, g, f and elements of V will be denoted by v, w, u.)

- A composition operation $+$ on H. This operation is required to be associative, i.e. $h+(g+f) = (h+g)+f$. This makes H a semigroup, called the *horizontal semigroup*, and justifies the notation $h + g + f$. In the free forest algebra, $+$ is forest concatenation.
- A composition operation $\cdot$ on V. Again, this is required to be associative. We omit the $\cdot$ symbol, writing vw instead of $v \cdot w$. Furthermore, we require there to be a neutral element $* \in V$, i.e. an element satisfying $v* = *v = v$ for all $v \in V$. In particular, V is a monoid, called the *vertical monoid*. In the free forest algebra, $\cdot$ is context composition, while $*$ is the empty context. For context composition we use multiplicative notation, and sometimes omit the $\cdot$ symbol.
- A (left) action $V \to H \to H$. The result of this action is denoted by $vh \in H$. The action must satisfy: $(vw)h = v(wh)$, which justifies the notation vwh. In the free forest algebra, the left action is substituting a forest into a context. There is an extensionality requirement: each two different $v, w \in V$ must induce different functions.
- An operation $f_0 : H \times V \to V$. This operation must satisfy $f_0(h, v)g = h + vg$ for all $g \in H$. Thanks to this axiom, we can without ambiguity write $h + v$ to denote the element $f_0(h, v)$. In the free forest algebra, $h + v$ is the context obtained from v by prepending the forest h (next to the root, not the hole). In a similar way we define $v+h$. In [3], the operation corresponding to f_0 was defined differently; here we can have a simpler definition thanks to the empty context.

As demonstrated above, the free forest algebra is a forest algebra, which is denoted by $(A, B)^\Delta$. (This notion depends on the leaf labels A and inner node labels B). When describing a forest algebra, we usually only give names to the carrier sets H and V, leaving the operations implicit.

Let (H, V) and (G, W) be two forest algebras. A *forest algebra morphism*

$$\alpha : (H, V) \to (G, W)$$

is defined as in universal algebra: it is a pair of functions

$$\alpha = (\alpha_H, \alpha_V) \qquad \alpha_H : H \to G \qquad \alpha_V : V \to W$$

that preserve all operations in the signature, eg.

$$\alpha_H(h + g) = \alpha_H(h) + \alpha_H(g) \qquad \alpha_H(vh) = \alpha_V(v)\alpha_H(h)$$
$$\alpha_V(vw) = \alpha_V(v)\alpha_V(w) \qquad \alpha_V(*) = *$$

and likewise for $h+v$ and $v+h$. To avoid clutter, we omit the subscripts, and write $\alpha(h)$ instead of $\alpha_H(h)$, likewise for v.

The point of forest algebras is to recognize forest languages. Let L be a set of forests over labels (A, B) and let (H, V) be a finite forest algebra. We say a morphism

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

recognizes a forest language L if membership $\bar{t} \in L$ depends only on the value $\alpha(t)$. The following result shows that recognizable forest languages are exactly the regular forest languages (as defined by, say, finite automata):

Theorem III.1 ([3])
A forest language is regular if and only if it is recognized by a finite forest algebra.

Note that in the above, we define recognition for languages of forests, and not languages of trees, as in the logic $\mathsf{EF} + \mathsf{F}^{-1}$. We will deal with this discrepancy in Section IV.

The *syntactic forest algebra* of a forest language L is a canonical forest algebra that recognizes the language. It is defined using a Myhill-Nerode equivalence over forests and contexts:

- Two forests $\bar{s}, \bar{t}$ are considered equivalent if

$$p\bar{s} \in L \qquad \text{iff} \qquad p\bar{t} \in L$$

  holds for every context p.
- Two contexts p, q are considered equivalent if for every forest $\bar{t}$, the forests $p\bar{t}$ and $q\bar{t}$ are equivalent in the above sense.

It turns out that the above defined equivalences are a forest algebra congruence; therefore a quotient forest algebra can be defined, where elements of H are equivalence classes of forests, and elements of V are equivalence classes of contexts. This quotient forest algebra is called the syntactic forest algebra of L. The morphism, which to each forest (resp. context) assigns its equivalence class is called the *syntactic morphism*. The syntactic morphism recognizes L, furthermore it is optimal in the sense that any morphism recognizing L can be uniquely extended to the syntactic morphism. In particular, the syntactic forest algebra is a morphic image of any forest algebra recognizing L.

IV. Tree-Definable vs Forest-Definable

A tree language L is *tree-definable* if there is a formula of $\mathsf{EF} + \mathsf{F}^{-1}$ that is true exactly (in the root of) trees in L. In this paper, it will sometimes be convenient to talk about $\mathsf{EF} + \mathsf{F}^{-1}$ formulas defining properties of forests (and not only trees). We say a forest language L is *forest-definable* if L is a boolean combination of languages of the form "some tree in the forest satisfies φ", with φ a formula of $\mathsf{EF} + \mathsf{F}^{-1}$. Such a boolean combination will be called a *forest formula*. For instance, the following property of a forest $t_1 + \cdots + t_n$ is forest-definable: all trees $t_1, \ldots, t_n$ contain a leaf with label a, and at least one of these trees has root label b.

Note that any nonempty tree language violates the following property (which is true for forest-definable languages)

$$t + t \in L \quad \text{iff} \quad t \in L \ ,$$

for the simple reason that $t + t$ is not a tree. Therefore no nonempty tree language is forest-definable. For similar reasons, no nonempty forest-definable language is tree-definable. The following Proposition relates the two concepts:

Proposition IV.1 Let L be a tree language over (A, B). The following conditions are equivalent:

- L is tree-definable.
- For each inner node label $b \in B$, the forest language $\{\bar{t} : b\bar{t} \in L\}$ is forest-definable.

In this paper, we will present a decidable characterization for forest-definable languages. Thanks to the above result, this will also give us a decidable characterization of tree-definable languages.

We only do bottom-up implication in the above proposition; the bottom-up can be shown using standard techniques.

Lemma IV.2 Let L be a forest-definable language and let $b \in B$ be an inner node label. The tree language $\{b\bar{t} : \bar{t} \in L\}$ is tree-definable.

Proof
The key observation is that if K is a language tree-definable in $\mathsf{EF} + \mathsf{F}^{-1}$, then the following tree language:

$$\mathsf{X}K = \{b(t_1 + \cdots + t_n) : b \in B, \exists i.\ t_i \in K\}$$

is also tree-definable. Once we demonstrate how to write a formula for $\mathsf{X}K$, the statement of the lemma immediately follows.

Note that definability of the language $\mathsf{X}K$ does not mean we can add the successor operator to the logic. This is because $\mathsf{X}K$ uses the successor only at the root. For instance, the property "some node at depth 4 has the same label as its parent" is tree-definable, contrary to the property "some node has the same label as its parent".

We now proceed to show the formula for the language $\mathsf{X}K$. Let φ be the formula defining K. We define $\hat{\varphi}$ to be the formula obtained from φ by replacing every subformula ψ by $\psi \wedge \mathsf{F}^{-1}\mathit{true}$. This way, quantification in $\hat{\varphi}$ is relativized to non-root nodes. Finally, the formula for $\mathsf{X}X$ is

$$\mathsf{EF}\big((\mathsf{F}^{-1}\mathit{true}) \wedge (\neg \mathsf{F}^{-1}\mathsf{F}^{-1}\mathit{true}) \wedge \hat{\varphi}\big) \ .$$

The above formula nondeterministically picks a successor x of the root, and then tests if $\hat{\varphi}$ holds in x. Since $\hat{\varphi}$ is relativized to non-root nodes, evaluation of $\hat{\varphi}$ will never leave the subtree of x. □

V. The equations and the main result

In this section we state our main result, the decidable characterization of the logic $\mathsf{EF} + \mathsf{F}^{-1}$.

Before we state the theorem, we define a relation $\dashv$ over contexts in a forest algebra. The idea is that $u \dashv w$ holds if u can be obtained from w by removing forests that are siblings of the main path (recall that the main path contains ancestors of the hole). Let (H, V) be a forest algebra. For $u, w \in V$, we write $u \dashv w$ if u, w can be decomposed as

$$u = v_0 v_1 \cdots v_n \qquad w = v_0(h_1 + v_1) \cdots (h_n + v_n)$$

for some $v_0, \ldots, v_n \in V$ and $h_1, \ldots, h_n \in H$. (In general this relation need not be transitive.)

Theorem V.1
A language is forest-definable in $\mathsf{EF} + \mathsf{F}^{-1}$ if and only if its syntactic algebra satisfies the following equations:

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

$$(vw)^\omega = (vw)^\omega w (vw)^\omega \ . \tag{2}$$

$$(u_1 w_1)^\omega (u_2 w_2)^\omega = (u_1 w_1)^\omega u_1 w_2 (u_2 w_2)^\omega \quad \textit{for } u_1 \dashv u_2, w_1 \dashv w_2 \ . \tag{3}$$

The equations in (1) say that the algebra is bisimulation invariant. The equation (2) says that the vertical monoid belongs to the variety DA (although the commonly used equation is different). Only the last equation is new.

The exponent ω in properties (2) and (3) stands for "for almost all n". In particular, equation (2) should be read as:

$$\exists m \forall n \geq m \qquad (vw)^n = (vw)^n w (vw)^n \ .$$

Before we prove this theorem, we present an important corollary:

Corollary V.2 It is decidable if a language is forest-definable (resp. tree-definable) in $\mathsf{EF} + \mathsf{F}^{-1}$.

Proof
To determine if a language is tree-definable, we calculate the languages $\{\bar{t} : b\bar{t} \in L\}$ and reduce to the characterization of forest-definable language thanks to Proposition IV.1. Therefore, we focus on deciding if a language is forest-definable.

We begin by finding the syntactic forest algebra. The syntactic forest algebra can be effectively calculated based on any representation of the tree language, be it a tree automaton, or a formula of some rich logic, such as MSO. In general, the syntactic forest algebra can be exponentially larger than a nondeterministic tree automaton, not to mention a formula of MSO.

Once the syntactic forest algebra has been calculated, the properties (1), (2) and (3) can be verified in polynomial time (with respect to the algebra). The exponent ω is not a problem; using standard semigroup arguments one can show that testing for $\omega = |V|!$ is enough, and the factorial powers can be computed using fast multiplication. The relation $\dashv$ over V can be computed using a fix-point algorithm. □

The decidability result could be easily extended to languages of contexts. Unary queries, i.e. pairs consisting of a forest and a node, are left as future work.

The rest of this paper is devoted to showing Theorem V.1. The "only if" implication in the above theorem is proved in Section VI using a standard Ehrenfeucht-Fraisse approach. The difficult part is the proof of the "if" implication, which is found in Section VII.

In the following fact, we show that property (3) in Theorem V.1 is not redundant. In a similar way one can prove that neither (1) nor (2) are redundant.

Fact V.3 There exists a forest algebra satisfying properties (1) and (2) but not (3).

Proof
Consider the syntactic forest algebra of the following language:

> All inner nodes are labeled by b, leaves are labeled by a or a'. If a node has a successor with label a, then it has an ancestor with a successor with label a'.

The syntactic forest algebra of this language satisfies properties (1) and (2); but it does not satisfy (3), as witnessed by:

$$(bb)^\omega((b+a')(b+a))^\omega \neq (bb)^\omega b(b+a)((b+a')(b+a))^\omega \ .$$

□

VI. Correctness

In this section we show that any forest-definable language satisfies the equations from Theorem V.1. Validity of (1) can easily be shown. We omit the proof of (2) for two reasons: first, it is the same as in the word case; and second, it follows along similar lines as the proof of (3).

The rest of this section is devoted to showing the validity of equation (3). Let L be a language definable by an $\mathsf{EF}+\mathsf{F}^{-1}$ formula φ. We need to show that the syntactic algebra of L satisfies equation (3). Recall that elements of the syntactic algebra are equivalence classes of the Myhill-Nerode equivalence relation. Therefore, in order to show the validity of (3), we have to show that for all contexts

$$p_1 \dashv p_2 \qquad q_1 \dashv q_2 \ ,$$

every context p and every nonempty forest $\bar{t}$, the following equivalence holds for almost all $k \in \mathbb{N}$:

$$\begin{gathered} s_0 = p(p_1q_1)^k(p_2q_2)^k\bar{t} \models \varphi \\ \text{if and only if} \\ s_1 = p(p_1q_1)^k p_1q_2(p_2q_2)^k\bar{t} \models \varphi \end{gathered} \tag{4}$$

holds. To keep the notation simple, we will only consider the case when

- The context p is empty.
- The forest $\bar{t}$ is a single node a
- The contexts p_1, p_2, q_1, q_2 are

$$\begin{aligned} p_1 &= b_1 \cdots b_n \qquad p_2 = b_1(*+a_1)\cdots b_n(*+a_n) \\ q_1 &= b_{n+1}\cdots b_m \quad p_2 = b_{n+1}(*+a_{n+1})\cdots(b_m(*+a_m) \end{aligned}$$

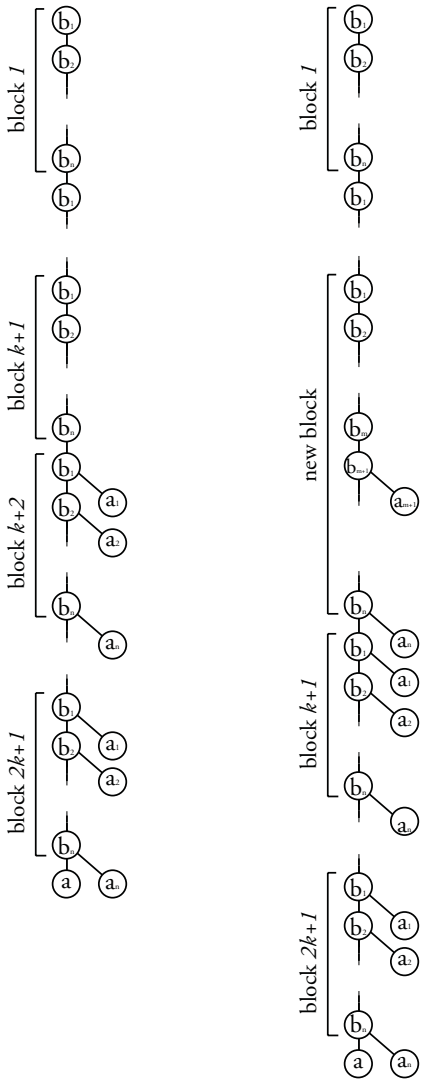

Fig. 1. The trees s_0 and s_1

for some $n < m$ and $b_1, \ldots, b_m \in B,\ a_1, \ldots, a_m \in A$.

- The labels $a_1, \ldots, a_m, b_1, \ldots, b_m$ and a are all distinct.

The trees s_0 and s_1 are shown in Figure 1. The general case can be shown in exactly the same manner.

Our proof uses a fairly standard Ehrenfeucht-Fraisse game technique. We first define the game and show that it corresponds to the logic $\mathsf{EF} + \mathsf{F}^{-1}$ over trees. Then, using this correspondence, we show in Lemma VI.2 that (4) holds once k is larger than the size of the formula φ.

We now proceed to define the Ehrenfeucht-Fraisse game. The game is played over two trees s_0, s_1 and consists of k consecutive rounds. There are two players, Spoiler and Duplicator. The aim of Spoiler is to show that the trees s_0, s_1 are different, while the aim of Duplicator is to show that they are similar. At the beginning of each round, a pebble x_0 is placed in a node of s_0 and a pebble x_1 is placed in a node of s_1. (At the beginning of the first round both pebbles are in the respective roots.)

A round is played as follows. If the labels of x_0, x_1 are

different, then player Spoiler is declared the winner and the game is terminated. Otherwise, Spoiler chooses one of the trees $s_i \in s_0, s_1$. He can then chose the type of move he makes:

- A descendant move. In a descendant move, Spoiler picks a new pebble position $x_i' > x_i$ that is a proper descendant of the old pebble position. Duplicator must respond by picking a new pebble position $x_{1-i}' > x_{1-i}$ in the other tree. If either player cannot find such a node, he loses and the game is terminated.
- An ancestor move. This is similar as above, except that a proper ancestor must be chosen.

If Duplicator successfully responds to Spoiler's move, the game continues on to the next round, with the pebbles updated to the new positions x_0', x_1'. If Duplicator has not lost after the last round, he is declared the winner.

The following lemma is standard. The size of a formula is the nesting depth of the operators EF and F^{-1}.

Lemma VI.1 If Duplicator can win the k-round game over trees s_0, s_1 then these trees cannot be distinguished by any $\mathsf{EF} + \mathsf{F}^{-1}$ formula of size k.

In the following lemma we show that (4) holds, under the assumptions on $p, p_1, p_2, q_1, q_2, \bar{t}$ stated previously:

Lemma VI.2 Duplicator can survive the k-round game over

$$s_0 = (p_1 q_1)^k (p_2 q_2)^k \bar{t}$$
$$s_1 = (p_1 q_1)^k p_1 q_2 (p_2 q_2)^k \bar{t}$$

Proof
In each of the trees s_0, s_1, the *main path* is the one leading to the $a = \bar{t}$. The *projection* of a node onto the main path is its closest ancestor (not necessarily proper) that is on the main path. Duplicator plays so as to preserve the following invariant:

> Assume there are $l \leq k$ rounds left. Let x_0, x_1 be the pebble positions, and let y_0, y_1 be their projections onto the main paths. The labels of x_0, x_1 are the same (this implies that the labels of y_0, y_1 are the same). Moreover, we have
>
> - The subtrees of y_0, y_1 are the same; or
> - The prefixes of y_0, y_1 are the same; or
> - Both the prefixes and suffixes contain at least l copies of $b_1, \ldots, b_m$.

□

VII. Completeness

This section is devoted to showing the more difficult implication in Theorem V.1. We will show:

Proposition VII.1 Any forest language recognized by an algebra satisfying (1), (2) and (3) can be forest-defined.

The above statement immediately implies the more difficult "if" part of V.1. Indeed, if L is recognized by an algebra satisfying (1), (2) and (3), then its syntactic algebra satisfies these equations. This is because the syntactic algebra is a morphic image of any algebra recognizing the language, and equations are preserved by morphic images.

If α is a morphism, then the *type under* α of a forest $\bar{t}$ is simply the value $\alpha(\bar{t})$. If the morphism α is clear, we omit the qualifier "under α". We will use the name *types* to refer to elements of H. Let $X \subseteq H$ be a set of types. We say a forest $\bar{t}$ is *X-trimmed* if the only subtrees of $\bar{t}$ that have a type in X are leaves. We say a tree language L is tree-definable *modulo* X if there is a formula φ such that

$$t \models \varphi \qquad \text{iff} \qquad t \in L$$

holds for all X-trimmed trees. In a similar fashion, we define forest-definable modulo X.

We will show a slightly more general result, which refers to parameters that can be used in an induction:

Proposition VII.2 Let $\alpha : (A, B)^\Delta \to (H, V)$ be a morphism, with (H, V) satisfying equations (1), (2) and (3). Let $X \subseteq H$ be a set of types, and let $v \in V$. For each $h \in H$ the following forest language is forest-definable modulo X

$$\{\bar{t} : v(\alpha(\bar{t})) = h\} \tag{5}$$

Clearly Proposition VII.1 follows from the above result, taking $X = \emptyset$, v to be the empty context, and doing a disjunction over all $h \in \alpha(L)$. The rest of Section VII is devoted to a proof of Proposition VII.2. The proof is by induction on several parameters:

- The size of H.
- The size of $H \setminus X$.
- The depth of v with respect to $\geq_{\mathcal{R}}$ (see below).
- The size of B, i.e. the number of inner node labels.

The order of these parameters is important: first we try to minimize H, then the other three parameters.

We would like to remark here that these parameters are properties of the whole morphism α, and not just its target (H, V). The notion of non-leaf types is dependent on α, as is being X-trimmed.

We say a morphism α into (H, V) is *leaf saturated* if for every $h \in H$, there is a representative leaf label a with $\alpha(a) = h$. In the rest of this paper, we will only consider such morphisms. Any morphism can be extended to one that is leaf saturated, without affecting the target forest algebra.

In VII-A, we define the concept of forest component, and the preorder $\geq_{\mathcal{R}}$. Then in Section VII-B, we lay out our proof strategy.

A. Green's relations for trees

We consider several preorders defined on H and V.

The preorders on V are obtained by simply treating V as a monoid and using Green's relations. As in any monoid, we

can define the following two relations on V:

$$v \leq_{\mathcal{R}} w \qquad \Longleftrightarrow \qquad v = wu \text{ for some } u \in V$$
$$v \leq_{\mathcal{L}} w \qquad \Longleftrightarrow \qquad v = uw \text{ for some } u \in V$$

(The perhaps counterintuitive direction of the order is traditional. The notation is supposed to suggest that more things can be reached from w than from v.) Both relations can easily be shown to be preorders (i.e. transitive and reflexive). Therefore, each induces an equivalence relation on V, these are called respectively $\sim_{\mathcal{L}}$ and $\sim_{\mathcal{R}}$. Equivalence classes of these equivalence relations are called $\mathcal{L}$-components and $\mathcal{R}$-components respectively. For our purposes, $\leq_{\mathcal{R}}$ will turn out to be more important than $\leq_{\mathcal{L}}$.

The preorder over H will need some interaction with V. We say a forest g is *reachable* from a forest h, if there is some context u such that $g = uh$. We write this as $g \sqsubseteq h$. By composing contexts, this is a preorder. The induced equivalence relation is denoted $\sim$ and its equivalence classes are called *(forest) components*. Moreover $g + h \sqsubseteq g$ holds for all $g, h \in H$; since $g + h = (h + *)g$.

The pre-order $\sqsubseteq$ has a "least" element. That is, the type $h = h_1 + \cdots + h_n$ that contains all types from H can be reached from any type of $h_i \in H$, via a context of the form

$$h_1 + \cdots + h_{i-1} + * + h_{i+1} + \cdots + h_n \ .$$

Note that in general, there may be more than one such least type; that is there may be some other types that are reachable from h. However, they all share the following property:

Lemma VII.3 Let $g \in H$ be a least type of H. There is a context that gives g for all arguments $f \in H$.

Proof
Let h be the concatenation of all types from H, as defined previously. By assumption on g, there is some context $v \in V$ such that $g = vh$. We claim that context $v(h + *)$ gives g for all arguments $f \in H$. Since h contains all forests from H, by (1), we have $f + h = h$ for all $f \in H$. In particular,

$$v(h + *)f = v(h + f) = vh = g \ .$$

□

B. Proof strategy

In our proof of Proposition VII.2, we consider three possible cases. First, in Section VII-D, we see what happens if $H \setminus X$ contains more than one forest component (i.e. at least two nonequivalent elements). Then, in Section VII-E, we see what happens when some inner node label $b \in B$ has the property that vb is strictly smaller than v in the preorder $\leq_{\mathcal{R}}$, in which case we say that the label b does not *preserve* v. Finally, in Section VII-F, we show that if neither of the above holds, then $vf = vg$ holds for all $f, g \in H$ and hence the formula φ in Proposition VII.2 need not depend on the forest $\bar{t}$ (it is either "true" or "false" depending on v and h). Section VII-F essentially corresponds to the induction base.

First however, we present a concatenation principle that will be used in Sections VII-D and VII-E. This principle is a generalization of the techniques used in Lemma IV.2.

C. A concatenation principle

We first present the principle for words, to give the appropriate intuition. A problem with $\mathsf{F} + \mathsf{F}^{-1}$ over words is that it is not closed under concatenation. For instance, the languages aa and $(a + b)^*$ are both $\mathsf{F} + \mathsf{F}^{-1}$ definable, but the language $(a + b)^*aa(a + b)^*$ is not. We claim however, that the concatenation LK is definable if the place in a word where L and K meet can be uniquely determined in $\mathsf{F} + \mathsf{F}^{-1}$:

Lemma VII.4 (Concatenation for words) Let L, K be two $\mathsf{F} + \mathsf{F}^{-1}$ definable word languages and let φ be a $\mathsf{F} + \mathsf{F}^{-1}$ formula with the semantic property that in every word, φ holds in at most one word position. The following word language is also definable in $\mathsf{F} + \mathsf{F}^{-1}$:

$$\{a_1 \dots a_n : a_1 \cdots a_i \in L, a_{i+1} \cdots a_n \in K, a_1 \cdots a_n, i \models \varphi\}$$

Proof
The same relativization as used in the proof of Lemma IV.2. We relativize the formula for L to positions where $\varphi \vee \mathsf{F}\varphi$ holds, and we relativize the formula for L to positions where $\mathsf{F}^{-1}\varphi$ holds. □

For trees, the situation is more complicated. First of all, there are two notions of concatenation: for forests and for contexts. We are interested in generalizing Lemma VII.4 to concatenation of contexts. In our generalization though, we may need to substitute many trees simultaneously. This leads to a slightly less appealing definition, which follows.

A formula is called *antichain* if in every tree, the set of nodes where it holds forms an antichain (not necessarily maximal). Recall that an antichain is a set of nodes incomparable with respect to the descendant relation. This is a semantic property, and may not be apparent just by looking at the syntax of the formula. For instance, the first two formulas are antichain, while the third is not:

- The node is a leaf: $\neg\mathsf{EF}\mathit{true}$.
- The node is a minimal occurrence of b: $b \wedge \neg\mathsf{F}^{-1}b$.
- The node has label b.

Using antichain formulas, we define our notion of concatenation. The ingredients are as follows:

- An antichain formula φ.
- Disjoint tree languages $L_1, \dots, L_n$ over (A, B).
- Leaf labels $a_1, \dots, a_n \in A$.

Let t be a tree over (A, B). We define the tree

$$t[(L_1, \varphi) \to a_1, \dots, (L_n, \varphi) \to a_n]$$

as follows. For each node x of t where the antichain formula φ holds, we determine the unique i such the tree language L_i contains the subtree of x. If such an i exists, we remove the subtree of x (including x), and replace x by a leaf labeled with a_i. Since φ is antichain, this can be done simultaneously

for all x. Note that the formula φ may depend also on ancestors of x, while the languages L_i only talk about the subtree of x.

Lemma VII.5 (Antichain concatenation for trees) Let φ, $L_1, \ldots, L_n$ and $a_1, \ldots, a_n$ be as above. For a tree-definable language K, the following tree language is tree-definable:

$$\{t : t[(L_1, \varphi) \to a_1, \ldots, (L_n, \varphi) \to a_n] \in K\} \ .$$

Proof
Relativization. □

The point of this lemma is that the languages L_i are taken out of their context inside the tree t. For instance L_i can say something like: "the root has label b and a child with label b",

$$L_i = \mathsf{EF}(b \ \wedge \ \mathsf{F}^{-1}b \ \wedge \ \neg\mathsf{F}^{-1}\mathsf{F}^{-1}\mathit{true}) \ ,$$

while in general the property "a node in the tree that has label b and a child with label b" cannot be expressed in $\mathsf{EF} + \mathsf{F}^{-1}$.

The above lemma also holds when K is forest-definable, in which case the language is forest-definable.

We now proceed to the proof of Proposition VII.2. As we said before, we consider three cases.

D. There is more than one forest component in $H \setminus X$

For the rest of Section VII-D, we fix some type $g \notin X$. Let $G \subseteq H$ be the forest component of g. We pick g so that no type in G can be reached from a type in $X \setminus G$. Intuitively speaking, types from G are close to the leaves. The essential idea is that we will add G to X, by squashing each subtree of type g to a single leaf with the g written in its label. This is done by applying the antichain concatenation lemma.

Let $W \subseteq V$ be the set of contexts that preserve G, i.e. contexts w such that holds $wg \sim g$ for some $g \in G$. The following lemma shows that thanks to equation (2), "some" in the above definition can equivalently be replaced by "all".

Lemma VII.6 Let $g \sim h \in H$ and $v \in V$. If $vg \sim g$ then $vh \sim g$.

Proof
Indeed, assume that $vg \sim g$ and $g \sim h$ hold. In this case we can find contexts u, w such that $uvg = h$ and $wh = g$. But then we have $wuvg = g$. In particular, we have $(wuv)^\omega g = g$. Using equation (2), we get

$$g = (wuv)^\omega g = (wuv)^\omega uv(wuv)^\omega g = (wuv)^\omega h \ ,$$

which shows g can be reached from vh. Since vh can be reached from g by assumption on h being reached from g, we obtain the desired $g \sim vh$. □

Let $F \subseteq H$ be the set of those types f from which a type in G can be reached. In particular, we have

$$G \subseteq F \subseteq H \ .$$

Note all types in $F \setminus X$ are from G by choice of G. Furthermore, the inclusion $F \subseteq H$ is proper, since $H \setminus X$ contains more than one forest component by assumption. The inclusion $G \subseteq F$ may also be proper, however all types in the difference $F \setminus G$ are from X.

We say $f \in H$ is a *bad brother* if $f + g \notin G$ holds for all $g \in G$. We say $f \in H$ is a *good brother* if $f + g \in G$ holds for all $g \in G$. Note that by definition of F, all good brothers are in F. An easy observation is that f is a good brother if and only if $f + *$ belongs to W. By Lemma VII.6, we see that every type in H is either a good brother or a bad brother. (In particular, all types of G are good brothers, since they can't be bad brothers by $g + g = g$.) Furthermore, since W is closed under context composition, good brothers are closed under forest concatenation $+$.

A *twig* is a tree of depth exactly two, i.e. a root and some leaves. A *twig node* is a node whose subtree is a twig.

Lemma VII.7 There is an formula ψ such that in any X-trimmed tree, ψ holds in nodes with a subtree of type in G.

Lemma VII.8 For each $g \in G$, the set of trees with type g is tree-definable modulo X.

The general idea is that (G, W) is a (smaller) forest algebra, and therefore the induction assumption can be applied to languages recognized by (G, W). However, thanks to bad brothers and such, (G, W) does not recognize the language in the lemma.

We will now use Lemmas VII.7 and VII.8 along with the concatenation principle to conclude the case considered in this section. The idea is that we add all types from G to X.

Let h, v be as in the statement of Proposition VII.2. We need to show that the language

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

is forest-definable modulo X. By induction assumption, we know that this language is forest-definable modulo $X \cup G$. In other words, there is some forest-definable set of forests K that agrees with L over $(X \cup G)$-trimmed forests. To describe L modulo X, we will use the antichain concatenation principle.

Let ψ be the formula from Lemma VII.7. Let

$$\varphi = \psi \wedge \neg\mathsf{F}^{-1}\psi \ .$$

This formulas holds in a node whose subtree has a type in G, and the node is closest to the root for this property. Thanks to the last clause, φ is an antichain formula. Let $G = \{g_1, \ldots, g_n\}$. By assumption that α is leaf saturated, for each g_i there is a leaf label $a_i \in A$ with $\alpha(a_i) = g_i$. For each g_i, let L_i be the set of trees with type g_i. Thanks to Lemma VII.8, each L_i is tree-definable modulo X.

It is easy to see that squashing a subtree with type g_i into a single leaf with label a_i does not change the type of the whole tree. More precisely, a forest $\bar{t}$ has the same value as

$$\bar{t}[(L_1, \varphi) \to a_1, \ldots, (L_n, \varphi) \to a_n] \ .$$

Furthermore, the above forest is $(X \cup G)$-trimmed, at least as long as $\bar{t}$ was X-trimmed. It follows that over X-trimmed forests, L agrees with

$$\{\bar{t} : \bar{t}[(L_1, \varphi) \to a_1, \ldots, (L_n, \varphi) \to a_n] \in K\} \ ,$$

which is forest-definable thanks to the antichain concatenation principle.

E. For some $b \in B$, we have $v\alpha(b) >_{\mathcal{R}} v$

Let $C \subseteq B$ be the set of those labels b that satisfy $v\alpha(b) \sim_{\mathcal{R}} v$, i.e. those labels that preserve v. In this section we show that if C is a proper subset of B, then the induction assumption can be applied.

Using the same technique as in Lemma VII.6, we can show:

Lemma VII.9 If $v \sim_{\mathcal{R}} w$ and $vu \sim_{\mathcal{R}} v$ then $wu \sim_{\mathcal{R}} v$.

Consider now the following equivalence relation:

$$h \equiv_{v+} g \qquad \Longleftrightarrow \qquad uh = ug \text{ for all } u <_{\mathcal{R}} v.$$

Lemma VII.10 Each equivalence class of $\equiv_{v+}$ is forest-definable modulo X.

Proof
By induction assumption; the third parameter is decreased. □

Lemma VII.11 Let $\bar{t}$ be a forest and let x be a node with a label outside C. For any subforest below x, replacing the subforest with an $\equiv_{v+}$ equivalent one does not change the value of $v\alpha(\bar{t})$.

Lemmas VII.10 and VII.11, together with the concatenation principle complete the proof of the case in this section. The idea is that for each minimal node x with a label outside C, we can use Lemma VII.10 to calculate the $\equiv_{v+}$-equivalence class of its subtree. By Lemma VII.11, this is enough to determine the value of $v\alpha(\bar{t})$.

F. The induction base

In this section, we assume that the techniques from the previous two sections cannot be applied. That is:

- All types of $H \setminus X$ are in a single forest component.
- For all $b \in B$, we have $v\alpha(b) \sim_{\mathcal{R}} v$.

We will show

$$vf = vg \qquad \text{for all } f, g \in H \setminus X \ . \tag{6}$$

Before we do this, we show how this completes the proof of Proposition VII.2. For every $h \in H$, we need to show that

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

is forest definable modulo X. By (6), there is some $h_0 \in H$ be such that $vf = h_0$ holds for all $f \in H \setminus X$.

- If an X-trimmed forest $\bar{t}$ contains an inner node label – which can easily be tested by the logic – then $\alpha(\bar{t})$ must be in the single forest component $H \setminus X$. In particular, $v\alpha(\bar{t}) = h_0$. So in this case, φ is either "true" or "false" depending on whether $h_0 = h$ or not.
- Otherwise, the forest $\bar{t}$ is the concatenation of some leaves $a_1 + \cdots + a_n$. In this case, the type of

$$v\alpha(a_1 + \cdots + a_n)$$

can be calculated based on the set of leaf labels in $\bar{t}$.

The rest of this section is devoted to showing (6).

The following Lemma is the key step in our proof (6). It says that not only any two types $h, g \in H \setminus X$ can be reached from each other (which follows from the assumption on there being one forest component), but they can also be reached from each other by only using contexts without any branching. Furthermore, the context that goes from g to h can be chosen independently of g. However, all these statements are relative to contexts $w \sim_{\mathcal{R}} v$.

Lemma VII.12 Let $h \in H \setminus X$. There are inner node labels $b_1, \ldots, b_n \in B$ such that $wh = w\alpha(b_1 \cdots b_n)g$ holds for all $w \sim_{\mathcal{R}} v$ and all $g \in H \setminus X$.

Proof
Let $g \in H \setminus X$. By Lemma VII.3, there are some $u_g, u_h \in V$ such that $g = u_g f$ and $h = u_h f$ for all $f \in H$. We assume that these contexts are of the form:

$$u_h = (f_1 + \alpha(b_1)) \cdots (f_n + \alpha(b_n))$$
$$u_g = (f_{n+1} + \alpha(b_{n+1})) \cdots (f_m + \alpha(b_m))$$

for some $n \leq m$ and $f_1, \ldots, f_m \in H$ and $b_1, \ldots, b_m \in B$. (In general, some of the f_i may be empty; but the proof follows the same lines.) Let us denote $\alpha(b_i)$ by v_i. We will show that

$$wh = wv_1 \cdots v_n g$$

holds for any $w \sim_{\mathcal{R}} v$. Since $b_1, \ldots, b_n$ were chosen independently of g, this will establish the statement of the Lemma.

By definition, we have

$$v_1 \cdots v_n \dashv u_h \qquad v_{n+1} \cdots v_m \dashv u_g \tag{7}$$

Let now $w \in V$ be such that $v \sim_{\mathcal{R}} w$. By assumption on w and Lemma VII.9, we have

$$wv_1 \cdots v_m \sim_{\mathcal{R}} w \ .$$

In particular, there is some $\overline{w} \in V$ such that

$$wv_1 \cdots v_m \overline{w} = w \ .$$

By iterating the above ω times, and appending h, we get

$$wh = w(v_1 \cdots v_m \overline{w})^\omega h \ .$$

Since $u_h f = h$ for all f, the above can be rewritten as

$$w(v_1 \cdots v_m \overline{v})^\omega (u_h u_g \overline{v})^\omega h \ .$$

Using the property from equation (3), we get

$$w(v_1 \cdots v_m \overline{w})^\omega (u_h u_g \overline{w})^\omega h =$$
$$w(v_1 \cdots v_m \overline{w})^\omega v_1 \cdots v_n u_g \overline{w} (u_h u_g \overline{v})^\omega h =$$
$$= w(v_1 \cdots v_m \overline{w})^\omega v_1 \cdots v_n g = wv_1 \cdots v_n g \ ,$$

which concludes the proof of the lemma. □

We now use the above Lemma to conclude the proof of (6). Indeed, let f, g be in $H \setminus X$. By the above lemma, there are $b_1, \ldots, b_m \in B$ such that

$$f = w\alpha(b_1 \cdots b_n)h \qquad g = w\alpha(b_{n+1} \cdots b_m)h$$

holds for all $w \sim_{\mathcal{R}} v$ and all $h \in H \setminus X$. Let $v_i = \alpha(b_i)$. By assumption on the equivalence class of v and by Lemma VII.9, there must be some $\overline{v} \in V$ such that

$$vv_1 \cdots v_m \overline{v} = v \ .$$

But then we have

$$vf = v(v_1 \cdots v_m \overline{v})^\omega f =$$
$$v(v_1 \cdots v_m \overline{v})^\omega v_{n+1} \cdots v_m \overline{v}(v_1 \cdots v_m \overline{v})^\omega f =$$
$$v(v_1 \cdots v_m \overline{v})^\omega g = vg \ .$$

The second equality follows from (2).

VIII. One quantifier alternation

In [11], it was shown that over words, $\mathsf{F}+\mathsf{F}^{-1}$ has the same expressive power as $\Sigma_2 \cap \Pi_2$, where

- Σ_2 are word properties definable by a first-order formula with quantifier prefix $\exists^*\forall^*$; the signature contains label tests and the left-to-right order on word positions.
- Π_2 are complements of Σ_2.

Both classes Σ_2, Π_2 can be extended to trees (using the descendant order on tree nodes). We show here that the result from [11] fails for trees:

Proposition VIII.1 Over trees, the classes $\mathsf{EF}+\mathsf{F}^{-1}$ and $\Sigma_2 \cap \Pi_2$ are incomparable.

The inequality

$$\mathsf{EF} + \mathsf{F}^{-1} \supsetneq \Sigma_2 \cap \Pi_2$$

is witnessed by the language "three nodes with label a", which cannot be defined in $\mathsf{EF} + \mathsf{F}^{-1}$ by virtue of (1). To show the remaining inequality

$$\mathsf{EF} + \mathsf{F}^{-1} \subsetneq \Sigma_2 \cap \Pi_2 \ ,$$

we will demonstrate in Lemma VIII.2 an invariant of Σ_2 that is not satisfied by the property "all successors of the root have label a".

Let s, t be two trees. An *embedding* of s in t is an injective function that assigns nodes of s to nodes of t, and preserves the labels and vertical order. If $n \in \mathbb{N}$ and s is a tree, we use ns to denote the n-fold concatenation of s.

Lemma VIII.2 Let φ be a formula of the form

$$\exists x_1 \dots x_i \forall y_1 \dots y_j \psi(x_1 \dots x_i, y_1 \dots y_j) \ ,$$

with φ quantifier-free. Let $n > i + j$, and let a be an inner node label. If t embeds in s, and $a(ns)$ satisfies φ, then so does $a(ns + t)$.

Proof
Assume then that $a(ns)$ satisfies φ. We need to show that $a(ns)+t$ does too. For $x_1, \dots, x_i$, we pick the same nodes in $a(ns)+t$ as the nodes in $a(ns)$ that witnessed $a(ns) \models \varphi$. We need to show that for any assignment of the nodes $y_1, \dots, y_j$ in $a(ns)+t$ that makes ψ false, we also can find an assignment in $a(ns)$ that makes ψ false. The key point is that any assignment of $x_1, \dots, x_i, y_1, \dots, y_j$ in $a(ns) + t$ must leave at least one copy of s without any variables; this copy can be used in $a(ns)$ to simulate t via the embedding. □

IX. Closing remarks

The main contribution of this paper is a characterization of languages definable in $\mathsf{EF} + \mathsf{F}^{-1}$, which is expressed in terms of equations that must be satisfied in the syntactic algebra. A corollary of this characterization is an algorithm for deciding if a given regular language can be expressed in $\mathsf{EF} + \mathsf{F}^{-1}$.

As mentioned in the introduction, there are many open problems waiting to be solved in this field. Of those closely related to $\mathsf{EF} + \mathsf{F}^{-1}$, the following look interesting:

- What are the equations for two-variable first-order logic with the descendant relation? The question boils down to: what equation should replace idempotency $h + h = h$? Here is one candidate: $v(h + h) + vh = vh + vh$.
- What are the equations for an extension of $\mathsf{EF} + \mathsf{F}^{-1}$, where we allow operators of the form $\mathsf{EF}^k\varphi$, with the meaning: "the current node has k incomparable descendants were φ holds". This seems to be a reasonable extension of $\mathsf{EF} + \mathsf{F}^{-1}$ that is capable of counting.

It is conceivable that a modification of the techniques developed in this paper can be sufficient to solve the above two logics. For other logics mentioned in this paper, such as full first-order logic, or even variants of $\mathsf{EF} + \mathsf{F}^{-1}$ with horizontal order, new techniques need to be developed.

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. *Decidable Properties of Tree Languages*. PhD thesis, Warsaw University, 2004.

[3] M. Bojańczyk and I. Walukiewicz. Forest algebras (unpublished manuscript). http://hal.archives-ouvertes.fr/ccsd-00105796.

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

[5] 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.

[6] M. Marx. Conditional XPath. *ACM Transactions on Database Systems*, 30:929–959, 2005.

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

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

[9] 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.

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

[11] 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.

[12] W. Thomas. On chain logic, path logic, and first-order logic over infinite trees. In *Logic in Computer Science*, pages 245–256, 1987.

[13] 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.