Structural properties of trees and expressivity of logical formalisms

Mikolaj Bojańczyk and Igor Walukiewicz

1 Introduction

Logics for specifying properties of labeled trees play an important role in several areas of Computer Science. Recently, attention has turned to logics for *unranked* trees, in which there is no *a priori* bound on the number of children a node may have. Barceló and Libkin [1] and Libkin [12] catalogue a number of such logics and contrast their expressive power. Many fundamental problems in this domain remain unsolved. For example, we do not as yet possess an effective criterion for determining whether a given property of trees is expressible in the temporal logics CTL, or CTL*, or in first-order logic with the ancestor relation.

There have been a number of efforts to extend this algebraic theory from words to trees trees; a notable recent instance is in the work of Ésik and Weil on preclones [8, 9]. Recently, Bojańczyk and Walukiewicz [7] introduced *forest algebras*, a generalization of the syntactic monoid for languages of forests of unranked trees. This algebraic model is rather simple, and in contrast to others studied in the literature, has already yielded effective criteria for definability in a number of logics: see Bojańczyk [4], Bojańczyk-Segoufin-Straubing [6], Bojańczyk-Segoufin [5]. Forest algebras are also implicit in the work of Benedikt and Segoufin [3] on first-order logic with successor.

An important motivation for studying forest algebras is to develop tools necessary for deciding if a given regular language can be defined in first-order logic. This problem can be formalized in several ways depending on the type of trees and operations present in the signature. We will show in this document some dependencies between several natural formalisations of this problem in a sense that decidability for one setting would give decidability for the other.

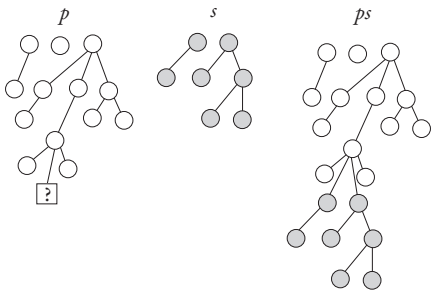
2 Trees, Forests and Contexts

Let A be a finite alphabet. Formally, forests and trees over A are expressions generated by the following rules: *(i)* if s is a forest and $a \in A$ then as is a tree; *(ii)* if $(t_1, \ldots, t_k)$ is a finite sequence of trees, then $t_1 + \cdots + t_k$ is a forest. We permit this summation to take place over an empty sequence, yielding the *empty forest,* which we denote 0. This gets the recursion started. So, for example, $s = a(b0 + c(a0 + ab0)) + a(a0 + b0)$ is a forest. Normally, when we

write such expressions, we delete the zeros. We depict s in the obvious fashion as an ordered forest of two ordered trees whose nodes are labeled by the letters a, b, c. In this example, the two root nodes are both labeled a, and there are five leaves altogether. The set of forests over A is a monoid with respect to forest concatenation $s + t$, with the empty forest 0 being the identity. We denote this set by H_A.

If x is a node in a forest, then the *subtree* of x is simply the tree rooted at x, and the *subforest* of x is the forest consisting of all subtrees of the children of x. In other words, if the subtree of x is as, with $a \in A$ and $s \in H_A$, then the subforest of x is s. Note that the subforest of x does not include the node x itself, and is empty if x is a leaf. A *forest language* over A is any subset of H_A.

A *context* p over A is formed by deleting a leaf from a nonempty forest and replacing it by a special symbol $\Box$. Think of $\Box$ as a kind of place-holder, or *hole*. Given a context p and a forest s, we form a forest ps upon substituting s for the hole in p.



In a similar manner, we can substitute another context q for the hole, and obtain a new context pq. The set of contexts over A forms a monoid, with respect to context composition pq, with the empty context $\Box$ being the identity. We denote this set by V_A.

Note that for all forests $s, t \in H_A$, V_A contains a context $s + \Box + t$, in which the hole has no parent, such that $(s + \Box + t)u = s + u + t$ for all $u \in H_A$.

Strictly speaking, our trees, forests and contexts are ordered, so that $s+t$ is a different forest from $t+s$ unless $s = t$ or one of s, t is 0. But all the applications in the present article effectively concern unordered trees, so there is no harm in thinking of $+$ as a commutative operation on forests.

3 Logics for Forest Languages

For a general treatment of predicate and temporal logics for unranked trees, see Libkin [12]. The logics we describe below are all fragments of monadic second-order logic, and thus the languages they define are all regular forest languages. These languages can be also represented by automata that are a minor modification of the standard bottom-up tree automata. The transition function is modified to cope with unbounded branching, and the definition of

acceptance needs to consider states in the roots of all the trees in the forest. See [7] for the definition.

3.1 First-order logic for trees and forests

Let A be a finite alphabet. Consider first-order logic equipped with unary predicates Q_a for each $a \in A$, and a single binary predicate $\prec$. Variables are interpreted as nodes in forests over A. $Q_a x$ is interpreted to mean that node x is labeled a, and $x \prec y$ to mean that node x is a (non-strict) ancestor of node y. A *sentence* ϕ – that is, a formula without free variables – consequently defines a language $L_\phi \subseteq H_A$ consisting of forests over A that satisfy ϕ. For example, the sentence

$$\exists x \exists y (Q_a x \wedge Q_a y \wedge \neg(x \prec y) \wedge \neg(y \prec x))$$

defines the set of forests containing two $\prec$-incomparable occurrences of a. We denote this logic by $FO[\prec]$. It is more traditional to consider logics over trees rather than over forests. For $FO[\prec]$ we need not worry too much about this distinction, since we can express in first-order logic the property that a forest has exactly one component ($\exists x \forall y (x \prec y)$). Thus the property that a set of trees is first-order definable does not depend on whether we choose to interpret sentences in trees or in forests.

3.2 Temporal logics

We describe here a general framework for temporal logics interpreted in trees and forests. By setting appropriate parameters in the framework we generate all sorts of temporal logics that are traditionally studied. The general framework is sometimes called *graded propositional dynamic logic* (graded PDL).

Syntax of temporal formulas. Temporal formulas are built starting with atomic *label formulas* a for $a \in A$. We combine temporal formulas with the usual boolean operations. We also define a temporal operator: if $k > 0$, Φ is a finite set of formulas, and $L \subseteq \Phi^+$ is a regular language of *words* over the alphabet Φ, then $\mathsf{E}^k L$ is a formula. The idea is that $\mathsf{E}^k L$ says there are at least k paths that satisfy L; the precise semantics are defined below. We place an additional restriction, called *unambiguity*, on the use of this operator: We require that the formulas $\Phi = \{\phi_1, \ldots, \phi_n\}$ are formally disjoint: that is, for all $i > 1$, ϕ_i has the form $\psi \wedge \neg \bigvee_{j<i} \phi_j$ for some formula ψ. We write $\mathsf{E}L$ for $\mathsf{E}^1 L$.

Semantics of temporal formulas. Usually, satisfaction for formulas is defined with respect to trees. For forests, the conventions are less well-established. Nevertheless, semantics for forests will be important for us, especially in the context of the wreath products. We will therefore use two notions of satisfaction: a *tree-satisfaction relation* $t \models_t \varphi$, which coincides with the usual notion of satisfaction, and a *forest-satisfaction relation* $t \models_f \varphi$, which is slightly unusual. The definition of the two will be mutually recursive. When t is a forest, then only the forest-satisfaction $t \models_f \varphi$ is defined, but when t is a tree, then both $t \models_f \varphi$ and $t \models_t \varphi$ are defined, with different meanings.

A label formula a is tree-satisfied by the trees whose root label is a, but is not forest-satisfied by any forest (even if the forest contains only a single tree). Whether a formula $\mathsf{E}^k L$ is tree-satisfied by a tree as depends only on the subforest of the root and not on the root node: that is, $as \models_t \psi$ if and only if $s \models_f \psi$. Finally, if $s \in H_A$, then $s \models_f \mathsf{E}^k L$ if and only if there are at least k distinct paths in s that satisfy L in the following sense: A path $x_1 \cdots x_n$ is a sequence of nodes connected by the child relation, beginning in one of the roots, and ending in some node, not necessarily a leaf. The path satisfies L if there is a sequence $\phi_1, \ldots, \phi_n \in \Phi$ such that the word $\phi_1 \cdots \phi_n$ belongs to L and for each $i = 1, \ldots, n$, the subtree of x_i tree-satisfies ϕ_i. Note that tree, and not forest, satisfaction is required in the subtree of x_j. Note also that the paths need not end in leaves, and that the sequence $\phi_1, \ldots, \phi_n$ is uniquely determined by the path, due to the unambiguity condition on Φ. As the paths need not end in leaves, some paths may be prefixes of others, for instance, the tree aaa forest-satisfies $\mathsf{E}^3 a^+$. Boolean operations have their usual interpretation.

Given a temporal formula ψ, we write L_ψ for the set of forests that forest-satisfy ψ.

EF When ψ describes a property of words, then $\mathsf{F}\psi$ describes the words where ψ holds at some position, possibly the first. We obtain an analogous temporal operator for trees and forests by defining $\mathsf{EF}\psi$ to be $\mathsf{E}L$, where $L = (\neg\psi)^*\psi$. Thus, when s is a forest, $s \models_f \mathsf{EF}\psi$ if and only if some subtree of s, possibly rooted at a root of s, tree-satisfies ψ. When t is a tree, $t \models_t \mathsf{EF}\psi$ if some proper subtree of t tree-satisfies ψ. Note how the tree semantics of this temporal operator resembles the "strict semantics" of EF in which one ignores the current node, while the forest semantics resemble the non-strict semantics. We denote by EF the forest languages definable by a formula built from the label formulas $a \in A$ using boolean operations and the temporal operator EF.

CTL When ψ, ϕ describe properties of words, then $\psi \mathsf{U} \phi$ describes the set of words $a_1 \cdots a_n$ where for some $i = 1, \ldots, n$, the word beginning in position i satisfies ϕ, and all the words beginning in positions $1, \ldots, i-1$ satisfy ψ. The analogous operator for trees and forests is $\mathsf{E}(\psi \wedge \neg\phi)^*\phi$, which we denote $\mathsf{E}\psi\mathsf{U}\phi$. The forest semantics is that the subtree of some node x tree-satisfies the formula ϕ, and the subtree at every strict ancestor of x tree-satisfies ψ. For the tree semantics, the root of the tree is ignored. By nesting this operator we get the logic CTL. (The dual operator $\mathsf{E}\neg(\psi \mathsf{U} \phi)$ is redundant in finite trees.)

First-order logic We can use the same formalism to characterize the languages definable in $FO[\prec]$ in terms of a temporal logic.

Theorem 1 *A forest language is definable in $FO[\prec]$ if and only if it is definable by a formula in the fragment of the language of temporal formulas using the operator $\mathsf{E}^k L$ only for word languages L that are first-order definable.*

This is a slight adaptation of Hafer and Thomas [10], and Moller and Rabinovich [14]. We will give the proof in the full paper. Note that the theorem fails without the restriction on unambiguity of the alphabet Φ. For instance, if we took $A = \{a, b, c\}$, $\Phi = \{\phi_1, \phi_2\}$, where $\phi_1 = a \vee c$, $\phi_2 = b \vee c$, then $L = (\phi_1\phi_2)^+$ is first-order definable as a word language. However, the language defined by $\mathsf{E}L$ is not first-order definable. (If it were, we would be able to define in first-order logic the set of forests consisting of a single path with an even number of occurrences of c.)

CTL* and PDL Finally, we define two more temporal logics by modifying the definitions above. CTL* is like the fragment of temporal logic in Theorem 1, except that we only allow $k = 1$ in $\mathsf{E}^k L$. In particular, CTL* is a subset of $FO[\prec]$. We also consider PDL, which is obtained by restricting the temporal formulas $\mathsf{E}^k L$ to $k = 1$, but without the restriction on L being first-order definable. If we place no restriction on either the multiplicity k or the regular language L, we obtain *graded PDL.* (Actually, one can show, using methods similar to Theorem 1, that graded PDL has the same expressive power as chain logic, which is the fragment of monadic second order logic where set quantification is restricted to chains, i.e. subsets of paths.)

4 Various flavours of first-order definability

We will in this section we show some dependencies between several natural formalisations of this problem in a sense that decidability for one setting would give decidability for the other.

A tree t over an alphabet (A, B) can be represented as a first-order structure $\mathcal{M}_t = \langle S, \leq_H, \leq_V, \{P_a\}_{a\in A}, \{P_b\}_{b\in B}\rangle$ where: $\leq_H$ is the *horizontal order*, i.e. order between siblings; $\leq_V$ is the *vertical order*, i.e. ancestor-descendant order; and P_a, P_b are monadic predicates encoding the labels. We can consider first-order logic over the complete signature $FO[\leq_H, \leq_V]$, or the logic only with vertical order $FO[\leq_V]$. This gives us four problems to study: we can vary logic, and we can consider either binary or unranked trees.

We want to study the dependencies between the following four problems:

BHV Given a regular language L of binary trees decide if L can be defined in $FO[\leq_H, \leq_V]$.

BV Given a regular language L of binary trees decide if L can be defined in $FO[\leq_V]$.

UHV As 1. but for unranked trees.

UV As 2. but for unranked trees.

Observe that the first case is just binary trees with left and right successors. The second case is for binary trees with no distinction between successors. Similarly, the forth case talks about unranked trees where the successors are not ordered.

We will show that UHV reduces BHV which reduces to BV which in turn reduces to UV.

A binary tree can be treated as a special case of an unranked tree. This suggest simple reductions form BHV to BV and from BV to UV.

The simplest case is the reduction BV to UV.

Lemma 2 A language L of binary trees can be defined in $FO[\leq_H]$ in the domain of binary trees iff it can be defined in $FO[\leq_H]$ in the domain of unranked trees.

Proof
There is a $FO[\leq_V]$ formula β saying that each internal node of the tree has rank 2. So if a formula $\varphi \in FO[\leq_V]$ defines L in the domain of binary trees, then $\varphi \wedge \beta$ defines the same set but in the domain of all unranked trees.

Reciprocally, if L is definable in $FO[\leq_V]$ in the domain of all unranked trees, then L is definable by the same formula in the domain of binary trees. □

For the reduction of BHV to UV we consider encoding of an order on successors into labels. This is possible as trees are binary so we need only two labels. For a binary tree t over an alphabet (A, B), let $enc(t)$ be the tree over the alphabet $(A \times \{0,1\}, B \times \{0,1\})$ where the label of each left successor gets additional tag 0 and that of each right successor gets tag 1 (the root gets tag 0 by convention).

Lemma 3 A language L of binary trees can be defined in $FO[\leq_H, \leq_V]$ iff $enc(L)$ can be defined in $FO[\leq_V]$.

Proof
Let γ be a formula $FO[\leq_V]$ saying that the root has tag 0 and that each internal node has precisely one son with tag 0 and one son with tag 1. It should be clear that $t \vDash \gamma$ iff $t = enc(t')$ for some binary tree t' over (A, B).

Now, given a formula φ from $FO[\leq_H, \leq_V]$ defining a set of binary trees L we can write a formula φ' such that $\varphi' \wedge \gamma$ defines $enc(L)$. We need just to change references to the alphabet (A, B) to corresponding references to the alphabet $(A \times \{0,1\}, B \times \{0,1\})$.

In the other direction. If the set $enc(L)$ is defined by a $FO[\leq_H, \leq_V]$ formula ψ then we can transform this formula to ψ' such that $t \vDash \psi'$ iff $enc(t) \vDash \psi$. For this it is enough to replace references to the extended alphabet with the references to the alphabet (A, B). For example a subformula $P_{b,0}(x)$ is changed to $P_b(x) \wedge \mathit{left}(x)$, where $\mathit{left}(x)$ is a formula saying that a node is a left son. Similarly for all other letters. □

Thanks to this lemma, to decide BHV, given an automaton for a language L of binary trees we can compute an automaton for $enc(L)$ and use a hypothetical decision procedure for UV.

Next we consider a reduction from UHV to BHV. This reduction is slightly more technical but very similar to those presented above. As the construction

is well-known we only sketch it briefly. To reduce the case of unranked trees to binary trees we use a standard encoding putting all successors of a node on the leftmost branch of the binary tree. Let $enc_{bin}(t)$ denote the binary tree encoding of t. The following lemma is a folklore.

Lemma 4 A language of unranked trees can be defined in $FO[\leq_H, \leq_V]$ iff $enc_{bin}(L)$ can be defined in $FO[\leq_H, \leq_V]$.

We can conclude since given an automaton for L we can compute and automaton for $enc_{bin}(L)$.

Finally we want to comment on the problems of reducing UV to UHV. A hypothetical reduction of this kind could for example show that horizontal order does not matter. This turns out not to be the case as the following two examples show.

Fact 5 The language "unranked trees with an even number of nodes" can be defined in monadic second-order logic with vertical and horizontal order; and its syntactic algebra satisfies $h + g = g + h$. However, this language cannot be defined in monadic second-order logic with only vertical order.

Fact 6 The language "unranked trees with all internal nodes of degree two and all paths of even length" can be defined in $FO[\leq_H, \leq_V]$ but not in $FO[\leq_V]$.

The later fact prevents us from having a straightforward reduction from UV to UHV. We conjecture that it may be still possible in the case of chain logic.

Conjecture If a language L can be defined in $\mathrm{CL}[\leq_H, \leq_V]$ and its syntactic algebra satisfies $h + g = g + h$ then it can be defined in $\mathrm{CL}[[\leq_V]$.

5 UTL

It will be convenient to have a temporal logic instead of first-order logic or chain logic. In the following we define UTL (unranked temporal logic) which is equivalent to $FO[\leq_H, \leq_V]$. There are variants of it that define $FO[\leq_V]$ as well as $\mathrm{CL}[\leq_H, \leq_V]$ and $\mathrm{CL}[\leq_V]$.

Definition 7 (UTL) A formula of UTL specifies a property of a forest with a distinguished root node.

- Every label a is a formula; such a formula holds in forest if the distinguished root has label a.
- Boolean combinations, including negation, are allowed.
- Let Γ be a finite set of formulas, and let $L \subseteq \Gamma^*$ be a first-order definable word languages. Then both $\mathsf{L}L$, $\mathsf{R}L$, and $\mathsf{E}L$ are formulas. The semantics are defined as follows:

- The formula $\mathsf{L}L$ holds in a forest $t_1+\cdots+t_i+\cdots+t_n$ with the root of t_i distinguished if there is a sequence $\varphi_1\cdots\varphi_{i-1}\in L$ and such that φ_i holds in t_i for all $i=1,\ldots,i-1$.
- Similarly for $\mathsf{R}L$ but this time we consider $t_{i+1}\cdots t_n$.
- The formula $\mathsf{E}L$ holds in a tree t if there is a sequence $\varphi_1\cdots\varphi_n\in L$ and a maximal path $x_1,\ldots,x_n$ such that φ_i holds in the subtree $t|_i$ for all $i=1,\ldots,n$. Here a maximal path is a sequence of nodes that leads from the root to some leaf (including both).

This syntax is very close to CLT*. Construct $\mathsf{E}L$ is equivalent to $\mathsf{E}\alpha$ construct of CLT*. Both talk about existence of a path, one uses an aperiodic language the other an LTL formula; which comes to the same thing as the two formalisms are equivalent [13, 15, 11, 19]. The construct $\mathsf{L}L$ is a like $\mathsf{E}L$ but on siblings to the left of the current node. Similarly, $\mathsf{R}L$ talks about the sequence of siblings to the right of the current node. One could introduce more CLT*-like operators both for future and the past over siblings [2] but this will not be necessary for us here. The following theorem was proved in [2] (Theorem 3.4). The slight modification is that contrary to what is claimed in op. cit. a since operator on siblings is necessary.

Theorem 8 *UTL is equivalent to* $FO[\leq_H,\leq_V]$.

Proof
By definition UTL can be translated to $FO[\leq_H,\leq_V]$. To prove the opposite direction we use the composition method for trees (see [18] for an introduction). We start with some definitions. As often in the context of E-F games we will consider formulas in prenex normal form. The quantifier depth of a formula is the depth of nesting of quantifiers, for formulas in prenex normal form it is the same as the number of quantifiers.

A *n-type* of a forest is a set of first-order sentences in prenex normal form of quantifier depth n that are true in the forest. Let $Types_n$ denote the set of all possible n-types. Recall that equivalence of n-types is characterized by n-step Ehrenfeucht-Fraise games. We will also need a notion of a n-type of a node r in a forest t. This is a set of prenex formulas $\varphi(x)$ of quantifier depth n, and with one free variable x such that $\varphi(r)$ holds in t.

For a forest $t=t_1+\cdots+t_n$ we define $roots_n(t)$ to be the word $\theta_1\ldots\theta_n$ where θ_i is the n-type of t_i. So $roots_n(t)$ is a word over the alphabet $Types_n$. As a word is also a tree, the notion of n-type is applicable to words too. The following lemma says that in order to know n-type of a forest t it is enough to know n-type of the word $roots_n(t)$.

Lemma 9 The n-type of t is determined by the n-type of $roots_n(t)$.

Proof
By an E-F transfer argument. □

We will need another lemma of this kind but now talking about paths in a tree. First we give some definitions. For a tree t and its node s we denote by

- $\theta_n(t,s)$ the n-type of the subtree rooted in s;
- $\theta_n^{\leftarrow}(t,s)$ the n-type of the forest of trees rooted in nodes $s' <_H s$, i.e, siblings of s to the left of s without considering s;
- $\theta_n^{\rightarrow}(t,s)$ the n-type of the forest of trees rooted in nodes $s' >_H s$, i.e., siblings of s to the right of s without considering s.

Finally, we define $path_n(t,s)$ which will be a word describing some type structure. Suppose that the path from the root of t to s is $s_0,\ldots,s_{i-1},s_i=s$. Then $path(t,s)$ is a word

$$(\theta_n^{\leftarrow}(t,s_0),\lambda(s_0),\theta_n^{\rightarrow}(t,s_0)),\ldots,(\theta_n^{\leftarrow}(t,s_{i-1}),\lambda(s_{i-1}),\theta_n^{\rightarrow}(t,s_{i-1})),\\ (\theta_n^{\leftarrow}(t,s_i),\theta_n(t,s_i),\theta_n^{\rightarrow}(t,s_i)),$$

That is, the letters are triples consisting of a n-type of the forest to the left of the node, the label of the node, and the n-type of the forest to the right. There is a small change in the last node where the label is replaced by n-type of the tree rooted in the node. We will use Δ_n to denote the set of letters that can appear in words of the form $path_n(t,s)$, i.e., $\Delta_n = \mathit{Types}_n \times \Sigma \times \mathit{Types}_n \cup \mathit{Types}_n^3$.

Lemma 10 The n-type of a node r in a forest t is determined by the n-type of $path_n(t,s)$.

Proof
The proof is easy using strategy transfer technique. □

By induction on the quantifier depth we show

> **(*)** for every $FO[\leq_H,\leq_V]$ sentence φ in a prenex normal form there is an UTL formula α_φ such that for every tree t: $t \vDash \varphi$ iff $t \vDash \alpha_\varphi$.

Notice that this statement talks only about trees. Before proving the statement we show how it implies a statement for forests. In case of forest UTL talks about forests with a distinguished root so instead of sentences of $FO[\leq_H,\leq_v]$ we rather take formulas with one free variable:

> **(**)** for every $FO[\leq,\leq_V]$ formula $\varphi(x)$ with one free variable, there is an UTL formula α_φ such that for every forest t with a distinguished root r we have: $t \vDash \varphi(r)$ iff $t,r \vDash \alpha_\varphi$.

Suppose that (*) holds for formulas of a quantifier depth n, we will show how to deduce (**) for formulas of the same quantifier depth. By Lemma 10 the n-type of r in t is determined by three types: $\theta_n^{\leftarrow}(t,r)$, $\theta_n(t,r)$, and $\theta_n^{\rightarrow}(t,r)$. By hypothesis we can calculate $\theta_n(t,r)$, as well types of all other trees in t. We show how to calculate $\theta_n^{\leftarrow}(t,r)$, the similar argument works for $\theta_n^{\rightarrow}(t,r)$. Let t_l be the forest of trees to the left of r. With this notation, $\theta^{\leftarrow}(t,r)$ is the type of t_l. By Lemma 9 $\theta^{\leftarrow}(t,r)$ is determined by the n-type of $\mathit{roots}(t_l)$. Since n-type is a first-order logic formula, for every n-type θ there is an aperiodic

language L_θ that describes the roots of forest having type θ. Language L_θ is over the alphabet $Types_n$, so it cannot be immediately used in UTL formula. By assumption, for each type θ we have an UTL formula which defines trees of type θ. Let L'_θ be obtained form L_θ by a homomorphism mapping each type to an UTL formula defining it. We get that $\theta^{\leftarrow}(t,r) = \theta$ iff $t, r \vDash \mathsf{L}L'_\theta$. Summarizing α_φ will be a conjunction of formulas of the form $\mathsf{L}L'_{\theta_l} \wedge \alpha_\theta \wedge \mathsf{R}L'_{\theta_r}$ for triples $(\theta_l, \theta, \theta_r)$ coming from trees satisfying φ.

Now we come back to the inductive proof of (*). The basic step is when the depth is 1. In this the sentence in question is either of the form $\exists x.\psi(x)$ or $\forall x.\psi(x)$ where $\psi(x)$ is quantifier free. It is enough to consider sentence of the first form because the later are obtained by negation. As there is only one variable, the truth of the sentence of this form depends only on the set of labels present in a tree. It is easy to express this kind of properties in UTL.

For the induction step we have take a sentence φ of the form $\exists x.\varphi'(x)$ where φ' has quantifier depth n. In this case, Lemma 10 gives us a first order sentence ψ such that for every tree t and its node s:

$$t \vDash \varphi'(s) \quad \text{iff} \quad path_n(t,s) \vDash \psi$$

Let L be the language of words over alphabet Δ_n which satisfy ψ. As ψ is a first-order sentence, L is aperiodic. So $\exists x.\varphi'(x)$ holds in t iff there is a node r in t such that $path(t,r) \in L$. The above discussion gives us for every triple of n-types $\theta_1, \theta_2, \theta_3$ an UTL formula that holds in r iff $(\theta^{\leftarrow}(t,r), \theta(t,r), \theta^{\rightarrow}(t,r)) = (\theta_1, \theta_2, \theta_3)$. Let L' be obtained form L by a homomorphism replacing each letter of Δ_n by an appropriate UTL formula. We claim that $\exists x.\varphi'(x)$ is equivalent to $\mathsf{E}L'(tt)^*$. Indeed, we have that $\exists x, \varphi'(x)$ holds iff there is r with $path_n(t,r) \in L$ which is equivalent to saying that the path from the root to some r is in L', which is equivalent to the fact that some maximal path is in $L'(tt)^*$, which in turn is expressed by $\mathsf{E}L'(tt)^*$. □

The proof of this theorem can be also done via translation to binary trees. Indeed, Hafer and Thomas [10] have shown that over binary trees $FO[\leq_H, \leq_V]$ is equivalent to CTL*. In [2] it is explained how to use this result to prove a result like Theorem 8. We preferred to reprove this result using composition method as it requires less encodings and writing complicated formulas.

Remark 11 In case of unranked trees, Moller and Rabinovich [14] show that $FO[\leq_V]$ is equivalent to CTL^* with counting modalities $\mathsf{D}^n\alpha$ which says that there are exactly n successors satisfying α. In our formalism CTL^* with D^n would correspond to replacing the operator $\mathsf{H}(L_0, L_1)$ with an operator $\mathsf{H}(L)$ having the obvious semantics. Moreover, we would need to require that L is not only aperiodic but also commutative, i.e., membership in L depends only on the number of occurrences of each letter.

Remark 12 Characterization of CL$[\leq_H, \leq_V]$ is obtained by dropping the requirement that L needs to be first-order definable in operator $\mathsf{E}L$. In a similar way CL$[\leq_V]$ is characterized by our logic where in $\mathsf{E}L$ all regular languages are

permitted and in $\mathsf{H}(L_0, L_1)$ the two languages should aperiodic and commutative.

6 Chain logic and CTL*

In this section we comment on extending Theorem 8. There are two directions.

The first direction is discussed in Section 6.1 and concerns CTL*. Using the same approach as in 4, we can show that definability for CTL* can be reduced to definability for first-order logic. Whether the opposite reduction holds is left as an open problem.

The second direction is discussed in Section 6.2 and concerns chain logic. It turns out that for chain logic, the picture is similar to the one for first-order logic. There are several possible variants, and a result similar to Theorem 8, which relates these variants, also holds.

6.1 CTL*

The logic CTL* is the restriction of CUTL (commutative unranked tree logic, defined in Section 5), where the next modality is only allowed in the form $\mathsf{X}\varphi$. A formula $\mathsf{X}\varphi$ holds in a tree if some successor subtree satisfies φ. In the style of Section 5, this is the same as saying that $H(K)$ is only allowed for word languages $K \subseteq A^*$ of the form A^*aA^*.

Proposition 13 A language of unranked trees can be defined in CTL* if and only if it can be defined in first-order logic without horizontal order, and its syntactic algebra satisfies $h + h = h$.

In particular, the above proposition shows that the problem: "can a given language of unranked trees be defined in CTL*?" can be reduced to any one of the first-order definability problems mentioned in Theorem 8. It seems, however, that the converse reduction requires some additional insight.

6.2 Chain logic

Chain logic is monadic second order logic, where set quantification is restricted to chains, i.e. sets of nodes linearly ordered by the vertical order. Chain logic has been studied mostly for binary (or unary, ternary etc.) trees [16, 17]. For unranked trees, it is tempting to consider an extension of chain logic, where quantification is also allowed for sets of siblings (i.e. chains with respect to the horizontal order). We call this logic *extended chain logic*. Since there are many parameters involved (binary / unranked, horizontal order / no horizontal order, extended / not extended), there is a proliferation of chain logics:

1. Chain logic over binary trees. Here, the extended model does not add expressive power; likewise for the horizontal order.

2. Chain logic over unranked trees without horizontal order.

3. Chain logic over unranked trees with horizontal order.

4. Extended chain logic over unranked trees without horizontal order.

5. Extended chain logic over unranked trees with horizontal order.

One can easily see that the logics 2,3,4,5 have different expressive powers. Using techniques as in the proof of Theorem 8, one can show that the definability problem is just as difficult for logics 1,2,4 and 5 (we will comment on these reductions later on). It is not clear how the definability problem for logic 3 is related to the other four, though.

We will not write out the entire reductions between the logics 1,2,4 and 5, but restrain ourselves to highlighting the differences with respect to Theorem 8. The reduction of 1 to 2 works the same way the reduction in Section 4. The reduction of 5 to 1 is done as in Section 4; we also use the same encoding. The only difficulty comes in showing the "if" direction in the analogue of Lemma 4:

Lemma 14 If $enc(L)$ is definable in chain logic over binary trees (with vertical order and left/right successors), then L is definable in extended chain logic over unranked trees (with horizontal and vertical order).

Proof
The difficulty here is that a chain in the binary tree $enc(t)$ does not correspond to a chain in the original tree t; it corresponds to an extended chain. An *extended chain* is a set of nodes $\{x_1, \ldots, x_n\}$ such that for each $i = 1, \ldots, n-1$ the node x_i is either an ancestor of x_{i+1}, or it is a sibling to the left of x_{i+1} (but not necessarily immediately to the left). One can show that quantification over extended chains does not add to the power of extended chain logic with horizontal and vertical order. □

Both the reductions from 2 to 4 and from 4 to 5 are done along the same lines as in Section 4. The key results are, respectively:

Proposition 15 A language of unranked trees can be defined in chain logic without horizontal order if and only if it can be defined in extended chain logic without horizontal order, and its syntactic algebra satisfies $h^\omega = h^{\omega+1}$.

Proposition 16 A language of unranked trees can be defined in extended chain logic without horizontal order if and only if it can be defined in extended chain logic with horizontal order, and its syntactic algebra satisfies $h + g = g + h$.

For the reduction of 4 to 3 we use a similar approach as in the previous section. The important result is the following lemma:

Lemma 17 Let L be tree language definable in first-order logic with horizontal order. If the syntactic forest algebra of L satisfies $h + h = h$, then L is definable in CTL*.

References

[1] Pablo Barceló and Leonid Libkin. Temporal logics over unranked trees. In *LICS*, pages 31–40. IEEE Computer Society, 2005.

[2] Pablo Barceló and Leonid Libkin. Temporal logics over unranked trees. In *LICS*, pages 31–40, 2005.

[3] M. Benedikt and L. Segoufin. Regular languages definable in FO. In *STACS'05*, volume 3404 of *LNCS*, pages 327 – 339, 2005. See the corrected version on the authors web page.

[4] Mikolaj Bojanczyk. Two-way unary temporal logic over trees. In *LICS*, pages 121–130, 2007.

[5] Mikolaj Bojanczyk and Luc Segoufin. Tree languages defined in first-order logic with one quantifier alternation. In *ICALP*, pages 233–245, 2008.

[6] Mikolaj Bojanczyk, Luc Segoufin, and Howard Straubing. Piecewise testable tree languages. In *LICS*, pages 442–451. IEEE Computer Society, 2008.

[7] Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In J. Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2007.

[8] Z. Ésik and P. Weil. On logically defined recognizable tree languages. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 2003.

[9] Z. Ésik and P. Weil. Algebraic recognizability of regular tree languages. *Theor. Comput. Sci*, 340(1):291–321, 2005.

[10] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *14th Internat. Coll. on Automata, Languages and Programming (ICALP'87)*, volume 267 of *LNCS*, pages 269–279, 1987.

[11] J. A. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Univ. of California, Los Angeles, 1968.

[12] L. Libkin. Logics for unranked trees: an overview. In *Automata, languages and programming*, volume 3580 of *Lecture Notes in Comput. Sci.*, pages 35–50. Springer, Berlin, 2005.

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

[14] Faron Moller and Alexander Moshe Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Inf. Comput.*, 184(1):147–159, 2003.

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

[16] W. Thomas. Logical aspects in the study of tree languages. In *Colloquium on Trees and Algebra in Programming (ICALP'84)*, pages 31–50, 1984.

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

[18] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1997.

[19] T. Wilke. Classifying discrete temporal properties. In *STACS'99*, volume 1563 of *LNCS*, pages 32–46, 1999.