A Complexity Approach to Tree Algebras: the Bounded Case

Thomas Colcombet
Université de Paris, CNRS, IRIF, F-75006, Paris, France

Arthur Jaquard
Université de Paris, CNRS, IRIF, F-75006, Paris, France

Abstract

In this paper, we initiate a study of the expressive power of tree algebras, and more generally infinitely sorted algebras, based on their asymptotic complexity. We provide a characterization of the expressiveness of tree algebras of bounded complexity.

Tree algebras in many of their forms, such as clones, hyperclones, operads, etc, as well as other kind of algebras, are infinitely sorted: the carrier is a multi sorted set indexed by a parameter that can be interpreted as the number of variables or hole types. Finite such algebras – meaning when all sorts are finite – can be classified depending on the asymptotic size of the carrier sets as a function of the parameter, that we call the complexity of the algebra. This naturally defines the notions of algebras of bounded, linear, polynomial, exponential or doubly exponential complexity...

We initiate in this work a program of analysis of the complexity of infinitely sorted algebras. Our main result precisely characterizes the tree algebras of bounded complexity based on the languages that they recognize as Boolean closures of simple languages. Along the way, we prove that such algebras that are syntactic (minimal for a language) are exactly those in which, as soon as there are sufficiently many variables, the elements are invariant under permutation of the variables.

2012 ACM Subject Classification Theory of computation → Tree languages; Theory of computation → Regular languages

Keywords and phrases Tree algebra, infinite tree, language theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.127

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Thomas Colcombet*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624), and the DeLTA ANR project (ANR-16-CE40-0007).

1 Introduction

Infinitely sorted algebras occur naturally in many contexts of language theory, graph theory or logic. A typical example is the case of tree algebras (such as clones, hyperclones, operads): plugging a subtree into another one requires a mechanism for identifying the leaf/leaves in which the substitution has to be performed. Notions such as variables, hole types, or colors are used for that. Another example is the one of graphs (HR- and VR-algebras [8]) in which basic operations (a) glue graphs together using a set of colors (sometimes called ports) for identifying the glue-points, or (b) add all possible edges between vertices of fixed given colors. In these examples, the algebras are naturally sliced into infinitely many sorts based on the number of variables/hole types/colors that are used simultaneously.

However, a technical difficulty arises immediately when using such algebras. Even when all sorts are finite (what we call a finite algebra), these algebras are not really finite due to the infinite number of sorts. This forbids, for instance, to entirely and explicitly describe the whole algebra in a finite way. And this is of course a problem for describing and using these algebras in an algorithm. Indeed, a concrete algorithm can only maintain a subset of the

algebra in its memory, say up to some given sort, or resort to other forms of representations, which often means not really working with the algebra. This is particularly annoying since often these objects are used for analyzing properties that admit finite descriptions, such as tree automata or logical formulae (eg in monadic second-order logic for describing properties of graphs).

This hurdle to handle infinitely sorted algebras can, arguably, be seen as one of the causes of the many years that it took before having a good definition of an algebra for infinite trees [2], or the time that it took before it was possible to characterize logically the expressiveness of recognizable properties of graphs under bounded tree-width hypothesis [4]. Also, the long history of results characterizing language families by decidable algebraic properties (initiated by the famous Schützenberger result [11]) has proven hard to extend to these more complex objects, such as trees.

Classifying algebras based on their complexity. In this paper, we initiate a new approach in the study of such algebras, which is to try to understand infinitely sorted algebras in simpler cases. And we define these simpler cases using complexity considerations. Indeed, in each of the above cases, the sorts are naturally indexed by a natural number parameter: the number of variables, or hole types, or colors. Hence, an algebra $\mathcal{A}$ would have a carrier of the form

$$(A_n)_{n\in\mathbb{N}},$$

together with suitable operations that depend on the particular algebra type. This algebra will be called *finite* if all the A_n sets are finite, and, in this case, we naturally define the *complexity map* of the algebra $c_{\mathcal{A}} \colon \mathbb{N} \to \mathbb{N}$ as follows:

$$c_{\mathcal{A}}(n) = |A_n| \ , \qquad \text{for all } n \in \mathbb{N}.$$

Finite algebras can naturally be classified using this map c. Simple classes are then *algebras of bounded complexity* if $c_{\mathcal{A}}$ is bounded, of *polynomial complexity* if $c_{\mathcal{A}}$ is bounded from above by some polynomial in n, etc.

Other interesting complexity classes can be defined using orbits. Indeed, in all of the mentioned examples, there is a natural operation that performs a renaming of the variables/hole types/colors. This renaming is parameterized by a bijection over variables/hole types/colors, and this permutation acts on the corresponding sort. Said differently, in all examples, there is an action of the symmetric group over n elements, $\mathbf{Sym}(n)$, over A_n. It is thus natural to consider the *orbit complexity map* $c^{\circ}_{\mathcal{A}}$ as follows

$$c^{\circ}_{\mathcal{A}}(n) = |A_n/\mathbf{Sym}(n)| \ , \qquad \text{for all } n \in \mathbb{N},$$

and define accordingly what are finite algebras of *bounded orbit complexity*, or *polynomial orbit complexity*, etc.

Related works. As mentioned above, there is a long history of understanding the expressive power of regular languages of words based on algebraic properties. The first work in this direction [11], characterizing star-free languages, intiated a long list of deep results. It was natural to extend this approach to trees. Here, the notion of algebra was less obvious, and several definitions have been used. Some algebras for trees are one sorted, such as deterministic bottom up automata (that can be seen as algebras). Some are two sorted, such as forest algebras [7]. Some others, such as preclones [9], have infinitely many sorts.

Characterizations of classes have been obtained using these approaches [5, 10, 6], but remain very limited due to difficulties inherent to the tree case. The study of algebras for infinite trees renewed the interest in these questions [1, 2, 3]. This line of works also highlights the difficulty to work with tree algebras, and the poor understanding we have so far of the mechanism of recognition for infinite objects.

Contributions of the paper. In this paper we establish some first results in this complexity analysis of infinitely sorted algebras, for the simplest complexity class, bounded complexity. Our results are of two kinds: a characterization of algebras of bounded complexity; and a characterization of the languages that they recognize, meaning we give a syntactic description of the properties that can be recognized by algebras in this class. We more particularly prove:

- A characterization of syntactic finite tree algebras of bounded complexity as those syntactic algebras in which, as soon as there are sufficiently many variables, the elements are invariant under permutation of variables. See Theorem 5.
- A characterization of languages recognized by finite tree algebras of bounded complexity as Boolean closures of simple languages. See Theorem 14.

The second result actually uses the first as a building block in its proof.

Structure of the paper. In Section 2, we recall some classical definitions, and introduce our notions of algebras. In Section 3, we look at the permutations of variables in finite tree algebras, and prove Theorem 5. In Section 4, we study in more depth the bounded complexity case for finite tree algebras, and establish our main result, Theorem 14. Section 5 is our conclusion.

2 Definitions

We denote by $\mathbb{N}$ the set of all non-negative integers. Given $n \in \mathbb{N}$, we write $[n] = \{0, 1, ..., n-1\}$. The symmetric group (resp. alternating group) over $[n]$ is denoted $\mathbf{Sym}(n)$ (resp. $\mathbf{Alt}(n)$), the symmetric group of any set X is denoted $\mathbf{Sym}(X)$. We denote by A^c the complement of a set A.

We fix a finite *ranked alphabet* Σ; the *arity* of a *symbol* $a \in \Sigma$ is denoted $\mathrm{ar}(a)$. It is a *constant* if $\mathrm{ar}(a) = 0$, and is *unary* if $\mathrm{ar}(a) = 1$. For $k \in \mathbb{N}$, we set $\Sigma_k = \{a \in \Sigma \mid \mathrm{ar}(a) = k\}$. A^* is the set of finite words over A, and $A^+ = A^* \setminus \{\varepsilon\}$.

2.1 Trees

In this section, we introduce notions and notations for trees.

We fix a countable set of *variables*. Given a finite set of variables X, a Σ, X-tree is, informally, a tree in which nodes are labelled by elements of Σ and leaves also possibly by variables. All variables have to appear at least once. Formally, a Σ, X*-tree* is a partial map $t \colon \mathbb{N}^* \to \Sigma \uplus X$ such that $\mathrm{dom}(t)$ is non-empty and prefix-closed, and furthermore:

- For all $u \in \mathrm{dom}(t)$ there exists $n \in \mathbb{N}$ such that $\{i \mid ui \in \mathrm{dom}(t)\} = [n]$, and
  - either $t(u) \in \Sigma_n$ (*symbol node*), or
  - $t(u) \in X$ and $n = 0$ (*variable node*). Note that a variable node is always a leaf.
- All variables from X appear in t, i.e. for all $x \in X$, $t(u) = x$ for some $u \in \mathrm{dom}(t)$.
- The root is not a variable, i.e. $t(\varepsilon) \notin X$.

$\Sigma, \emptyset$-trees are simply called Σ-*trees*. The elements in $\mathrm{dom}(t)$ are called *nodes*. The prefix relation over nodes is called the *ancestor relation*. The node ε is called the *root* of the tree. The tree t is *finite* if it has finitely many nodes. A *branch* of a tree t is a maximal set of nodes ordered under the ancestor relation. Let $\mathrm{FiniteTrees}(\Sigma, X)$ bet the set of finite Σ, X-trees, for all finite set of variables X.

Building trees. We introduce now some operations on trees. See Fig. 1.

- $a(x_0, \dots, x_{n-1})$, for $x_0, \dots, x_{n-1}$ variables and $a \in \Sigma_n$, denotes the $\Sigma, \{x_0, \dots, x_{n-1}\}$-tree consisting of a root labelled a, and children $0, \dots, n-1$ labelled with variables $x_0, \dots, x_{n-1}$ respectively.
- $s \cdot_x t$, for two trees $s \in \mathrm{FiniteTrees}(\Sigma, X)$, $t \in \mathrm{FiniteTrees}(\Sigma, Y)$ and a variable $x \in X$, is the $\Sigma, (X \setminus \{x\}) \cup Y$-tree s in which t is substituted for every occurrences of the variable x.
- $\widetilde{\sigma}(t)$, for a tree $t \in \mathrm{FiniteTrees}(\Sigma, X)$ and $\sigma \colon X \to Y$ a surjective map, is the Σ, Y-tree obtained as t in which variable $\sigma(x)$ has been substituted to x for all $x \in X$. Note that $\widetilde{\sigma} \circ \widetilde{\tau} = \widetilde{\sigma \circ \tau}$.
- $t[x_0 \leftarrow t_0, \dots, x_{n-1} \leftarrow t_{n-1}]$ denotes the tree of sort $X \setminus \{x_0, \dots, x_{n-1}\} \cup \bigcup_i Y_i$ obtained from t by simultaneously substituting the tree t_i for the variable x_i for all $i \in [n]$, where t is a tree of sort X, $x_0, \dots, x_{n-1} \in X$, and $t_0, \dots, t_{n-1}$ are trees of sort Y_i for all $i \in [n]$. Note that this operation is equivalent to a combination of the previous ones.
- $a(t_0, ..., t_{n-1})$, for $a \in \Sigma_n$, denotes the tree of root a and children $t_0, \dots, t_{n-1}$ at respective positions $0, \dots, n-1$. Again, this operation is equivalent to a combination of the previous ones.

▶ **Lemma 1.** *All finite trees can be obtained from the $a(x_0, \dots, x_{n-1})$'s using the operation $\cdot$.*

Expressions denoting finite trees. For X a finite set of variables, an FT_Σ-*expression of sort X (over the alphabet Σ)* is an expression built inductively as follows:

- $a(x_0, \dots, x_{n-1})$ is an FT_Σ-expression of sort $\{x_0, \dots, x_{n-1}\}$ for every symbol $a \in \Sigma_n$,
- $S \cdot_x T$ is an FT_Σ-expression of sort $X \setminus \{x\} \cup Y$ for all FT_Σ-expressions S of sort X, all FT_Σ-expressions T of sort Y, and all variables $x \in X$ (*substitution*),
- $\tilde{\sigma}(T)$ is an FT_Σ-expression of sort Y for all FT_Σ-expressions T of sort X, and surjective map $\sigma \colon X \to Y$ (*renaming*).

For an FT_Σ-expression T of sort X, $[\![T]\!]$ denotes its evaluation into a finite Σ, X-tree using the operations of substitution and renaming.

Contexts. We define now contexts, which are terms with a specific leaf called the hole. Since we work in a multi-sorted algebra, the hole itself has a sort. Essentially, to a hole of sort X will be substituted a term of sort X. Formally, for fixed finite set of variables Y, an *context of sort X with hole of sort Y* (or simply a FT_Σ-*context*) is defined inductively as an expression of sort X, using the extra construction $\Box_Y$ (the *hole of sort Y*) which is a context of sort Y with hole of sort Y. This new construction may appear multiple times in a context but has to appear at least once.

For C a context of sort X with hole of sort Y, $[\![C]\!] \colon \mathrm{FiniteTrees}(\Sigma, Y) \to \mathrm{FiniteTrees}(\Sigma, X)$ is the function which to a tree of sort Y t associates the tree of sort X obtained by evaluating the operations as above, interpreting $\Box_Y$ as t.

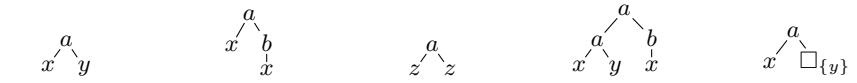


Figure 1 Trees and contexts with their notations. Here $\sigma(x) = \sigma(y) = z$.

2.2 Finite tree algebras

Our notion of tree algebra is the natural notion associated to finite trees equipped with the above operations. We give here a more formal definition, though the detail of identities is more for reference. What matters is that it is defined such that the free algebra coincides with finite trees.

An *FT_Σ-algebra* $\mathcal{A}$ consists of an infinite collection of carrier sets A_X indexed by finite sets of variables X, together with operations:

- $\sigma^{\mathcal{A}} \colon A_X \to A_Y$ for all surjective maps $\sigma \colon X \to Y$,
- $a(x_0, \dots, x_{n-1})^{\mathcal{A}} \in A_{\{x_0,\dots,x_{n-1}\}}$ for all $a \in \Sigma_n$ and variables $x_0, \dots, x_{n-1}$,
- $\cdot_x^{\mathcal{A}} \colon A_X \times A_Y \to A_{X \setminus \{x\} \cup Y}$ for all finite sets of variables X, Y and $x \in X$,

that satisfy the expected identities, i.e. the ones guaranteeing that several ways to describe the same tree yield the same evaluation in the algebra. Formally, for all s, t, u that belong to A_X, A_Y, A_Z respectively,

- $(s \cdot_x^{\mathcal{A}} t) \cdot_y^{\mathcal{A}} u = x \cdot_x^{\mathcal{A}} (t \cdot_y^{\mathcal{A}} u)$ for all $x \in X \cap Z$ and $y \in X \cap Y$ (horizontal associativity), and
- $(s \cdot_x^{\mathcal{A}} t) \cdot_y^{\mathcal{A}} u = x \cdot_x^{\mathcal{A}} (t \cdot_y^{\mathcal{A}} u)$ for all $x \in X$ and $y \in Y \setminus X \cup \{x\}$ (vertical associativity),

for all s, t that belong to A_X, A_Y, $x \in X$ and surjection $\sigma \colon X \to Y$,

- $\sigma^{\mathcal{A}}(s \cdot_x^{\mathcal{A}} t) = \sigma^{\mathcal{A}}(s) \cdot_x^{\mathcal{A}} t$ if $\sigma^{-1}(\sigma(x)) = \{x\}$ and $\sigma(y) = y$ for all $y \in X \cap Y \setminus \{x\}$,
- $\sigma^{\mathcal{A}}(s \cdot_x^{\mathcal{A}} t) = s \cdot_x \sigma^{\mathcal{A}}(t)$ if $\sigma(y) = y$ for all $y \in X \cap Y \setminus \{x\}$,

for all surjective maps $\sigma \colon X \to Y$ and $\tau \colon Y \to Z$, $(\tau \circ \sigma)^{\mathcal{A}} = \tau^{\mathcal{A}} \circ \sigma^{\mathcal{A}}$, and for all surjections $\sigma \colon \{x_0, \dots x_{n-1}\} \to Y$ and $a \in \Sigma_n$, $\sigma^{\mathcal{A}}(a(x_0, \dots, x_{n-1})^{\mathcal{A}}) = a(\sigma(x_0), \dots, \sigma(x_{n-1}))^{\mathcal{A}}$. In practice, we shall not explicitly use these identities, and simply write two elements of the algebra equal as soon as they obviously come from expressions denoting the same trees.

A *morphism of FT_Σ-algebras* from $\mathcal{A}$ to $\mathcal{B}$ is a family of maps $\alpha_X \colon A_X \to B_X$ for all finite sets of variables X which preserves all operations, i.e. $\alpha_Y(\sigma^{\mathcal{A}}(s)) = \sigma^{\mathcal{B}}(\alpha_X(s))$ for all surjective map $\sigma \colon X \to Y$, $\alpha(a(x_0, \dots, x_{n-1})^{\mathcal{A}}) = a(x_0, \dots, x_{n-1})^{\mathcal{B}}$, and $\alpha_{X \setminus \{x\} \cup Y}(s \cdot_x^{\mathcal{A}} t) = \alpha_X(s) \cdot_x^{\mathcal{B}} \alpha_Y(t)$ for all $s \in A_X$, $t \in A_Y$ and $x \in X$.

The FiniteTrees(Σ, X) sets equipped with the operations of substitution and renaming form an FT_Σ-algebra (it is the free FT_Σ-algebra generated by $\emptyset$). For $\mathcal{A}$ a FT_Σ-algebra, its associated *evaluation morphism* is the unique morphism from FiniteTrees(Σ) to $\mathcal{A}$.

A *congruence* $\sim$ over a FT_Σ-algebra $\mathcal{A}$ is a family $\sim$ of equivalence relations over the A_X's (each denoted $\sim$) such that, for any $a \sim b \in A_X$, $c \sim d \in A_Y$, $y \in Y$ and surjective $\sigma \colon X \to Y$: $c \cdot_y a \sim c \cdot_y b$; $c \cdot_y a \sim d \cdot_y a$ and $\widetilde{\sigma}(a) \sim \widetilde{\sigma}(b)$. From such a congruence, one can define the *quotient algebra* $\mathcal{A}/\sim$ in the natural way.

Compact presentation, and complexity. Our definition of algebras does not so far match the one used in the introduction. In the introduction, algebras were considered as using natural numbers as sorts, while here, our sorts are indexed by finite sets. What would be these algebras should be pretty clear. The presentation used here is simpler to present and to use. It is an exercise to show the equivalence.

What matters is that in what follows, a FT_Σ-algebra is of *bounded complexity* if there exists a bound K such that $|A_X| \leqslant K$ for all finite sets of variables X.

2.3 Languages and syntactic algebras

A *language of finite* Σ*-trees* L is a set of Σ-trees. It is *recognized by* an FT_Σ-algebra $\mathcal{A}$ if there is a set $P \subseteq A_\emptyset$ such that $L = \alpha^{-1}(P)$ in which α is the evaluation morphism of $\mathcal{A}$.

The *syntactic congruence* $\sim_L$ *of a language* L of finite Σ-trees is defined in the following way $s \sim_L t$ for s, t finite Σ-trees if, for all FT_Σ-contexts C, $[\![C]\!](s) \in L$ if and only if $[\![C]\!](t) \in L$. It is easy to prove that $\sim_L$ is indeed a congruence. The quotient algebra $\text{FiniteTrees}(\Sigma)/_{\sim_L}$ is called the *syntactic algebra* of L.

▶ **Example 2.** The language of all finite trees in which the symbol a appears has for syntactic FT_Σ-algebra the algebra with sorts $A_X = \{0, 1\}$, for all finite set of variables X, and whose evaluation morphism is such that $\alpha(t) = 1$ if and only if t is a tree in which a appears.

2.4 Tree automata

A *tree automaton* $\mathcal{B} = (Q, I, (\delta_a)_{a\in\Sigma})$ over Σ has a finite set Q of *states*, a set of *accepting states* $I \subseteq Q$ and a *transition relation* $\delta_a \subseteq Q \times Q^{\text{ar}(a)}$ for every symbol $a \in \Sigma$. A *run of* $\mathcal{B}$ over a finite tree t is a mapping $\rho\colon \text{dom}(t) \to Q$ such that, for any vertex $u \in \text{dom}(t)$ with $t(u) = a \in \Sigma$, $(\rho(u), (\rho(u0), ..., \rho(u(\text{ar}(a) - 1)))) \in \delta_a$. A run is *accepting* if $\rho(\varepsilon) \in I$. A language L of finite trees is called *regular* if it is recognized by a tree automaton $\mathcal{B}$, meaning the trees in L are exactly those for which there is an accepting run in $\mathcal{B}$.

Ex. 3 shows the translation from tree automata to tree algebra.

▶ **Example 3** (Automaton algebra)**.** Consider a regular language L of finite trees recognized by the tree automaton $\mathcal{B} = (Q, q_0, (\delta_a)_{a\in\Sigma})$.

Consider some finite set of variables X. An X*-run profile* is a tuple $\tau \in Q \times \mathcal{P}(Q)^X$. For a Σ, X-tree t, $\tau = (p, (U_x)_{x\in X})$ is a *run profile over* t if there exists a run ρ of the automaton over Q such that $\rho(\varepsilon) = p$ and for all variables $x \in X$, U_x is the set of states assumed by ρ at leaves labelled x. We define a tree algebra $\mathcal{A}$ that has as elements of sort X sets of X-run profiles. The definition of the operations is natural, and is such that the image of a Σ, X-tree t under the evaluation morphism yields the set of run profiles over t. It naturally recognizes the language L.

Note that this definition yields an algebra of doubly exponential complexity (and hence, this is an upper bound for regular languages). Of course, in practice, one can restrict the algebra to the reachable elements, and this may dramatically reduce the complexity.

The converse translation is also true, yielding the following classical result (it is for instance proved for preclones in [9]).

▶ **Proposition 4.** *A finite tree language is regular if and only if it is recognized by a finite* FT_Σ*-algebra.*

3 Fundamental results on permutations in tree algebras

This section studies some fundamental phenomena concerning the effect of variable permutations on tree algebras. Its main objective is to prove Theorem 5. It is a characterization of syntactic FT_Σ-algebras of bounded complexity which turns out to be key in the subsequent developments. Beyond that, several intermediate results in this section may also be relevant in the analysis of algebras of unbounded complexity.

In this section, given an FT_Σ-algebra $\mathcal{A}$, $\varphi_X^{\mathcal{A}}\colon \mathbf{Sym}(X) \to \mathbf{Sym}(A_X)$ (or simply φ_X when there is no ambiguity) denotes the group morphism $\sigma \mapsto \sigma^{\mathcal{A}}$, for all finite sets of variables X.

▶ **Theorem 5.** *A finite syntactic FT_Σ-algebra $\mathcal{A}$ is of bounded complexity if and only for all sufficiently large finite set of variables X, $\ker(\varphi_X^{\mathcal{A}}) = \boldsymbol{Sym}(X)$.*

The meaning of $\ker(\varphi_X) = \mathbf{Sym}(X)$ is that permuting the variables has no effect on A_X (i.e. $\sigma^{\mathcal{A}} = \mathrm{Id}_{A_X}$ for every $\sigma \in \mathbf{Sym}(X)$). We fix from now the FT_Σ-algebra $\mathcal{A}$.

Our first step is to define a relation $\equiv_{\mathcal{A}}$ which we show to be a congruence equivalent to the syntactic one (Proposition 6). We set for all $a \in A_X$:

$$\begin{aligned} \langle a \rangle : \quad (A_\emptyset)^X &\to A_\emptyset \\ b &\mapsto a[x_0 \leftarrow b(x_0), ..., x_{n-1} \leftarrow b(x_{n-1})] \ , \end{aligned}$$

in which $X = \{x_0, ..., x_{n-1}\}$. Define now $a \equiv_{\mathcal{A}} a'$, for $a, a' \in A_X$ to hold if $\langle a \rangle = \langle a' \rangle$ (note that it does not depend on the numbering of variables).

▶ **Proposition 6.** *$\equiv_{\mathcal{A}}$ is a congruence. If $\mathcal{A}$ is a syntactic FT_Σ-algebra, then $a = b$ if and only if $a \equiv_{\mathcal{A}} b$, for all a, b in $\mathcal{A}$.*

By a simple counting argument, we obtain the following corollary.

▶ **Corollary 7.** *Let $\mathcal{A}$ be a finite syntactic FT_Σ-algebra, then*

$$|A_X| \leq |A_\emptyset|^{|A_\emptyset|^{|X|}} \ .$$

Recall the following result from finite group theory:

▶ **Proposition 8.** *Let $\varphi \colon \boldsymbol{Sym}(X) \to G$ be a group morphism. If $|X| \geq 5$, then $\ker(\varphi)$ equals either $\boldsymbol{Sym}(X)$, $\boldsymbol{Alt}(X)$ or $\{\mathrm{Id}_X\}$.*

Using Propositions 6 and 8, we prove that, whenever X is large enough, $\ker(\varphi_X)$ may only be $\mathbf{Sym}(X)$ or $\{\mathrm{Id}_X\}$.

▶ **Proposition 9.** *Let $\mathcal{A}$ be a finite syntactic FT_Σ-algebra. There is an integer M such that, for all X of cardinal at least M, either $\ker(\varphi_X) = \boldsymbol{Sym}(X)$ or $\ker(\varphi_X) = \{\mathrm{Id}_X\}$.*

Proof. Let $M = \max(5, |A_\emptyset| + 1)$. Let X be a finite set of variables such that $|X| \geq M$. By Prop. 8, $\ker(\varphi_X)$ is either $\mathbf{Sym}(X)$, $\mathbf{Alt}(X)$ or $\{\mathrm{Id}_X\}$. Assume, for the sake of contradiction, that $\ker(\varphi_X) = \mathbf{Alt}(X)$. This implies that the image of φ_X has exactly 2 elements: permutations of signature $+1$ are sent to Id_{A_X}, and those of signature -1 are sent to another (distinct) element. Let us call τ this permutation of A_X.

Let $t \in \mathbf{Sym}(X)$ be a transposition, let us show that $t^{\mathcal{A}} = \mathrm{Id}_{A_X}$. According to Proposition 6, we only need to prove that $\langle t^{\mathcal{A}}(a) \rangle = \langle a \rangle$ for all $a \in A_X$. Let $b \in (A_\emptyset)^X$, since $n \geq |A_\emptyset| + 1$, there must exists $x \neq y$ in X such that $b(x) = b(y)$, thus: $\langle t^{\mathcal{A}}(a) \rangle(b) = \langle \tau(a) \rangle(b) = \langle (x\ y)^{\mathcal{A}}(a) \rangle(b) = \langle a \rangle(b \circ (x\ y)) = \langle a \rangle(b)$. Since this holds for all $a \in A_X$, $t^{\mathcal{A}} = \mathrm{Id}_{A_X}$. This is a contradiction. ◀

According to Proposition 9, for X large enough, $\ker(\varphi_X)$ may only be $\mathbf{Sym}(X)$ or $\{\mathrm{Id}_X\}$. The next result shows that one of these two cases in fact holds for all sufficiently large X.

▶ **Proposition 10.** *Let $\mathcal{A}$ be a finite syntactic FT_Σ-algebra. There is an integer M such that, either $\ker(\varphi_X) = \boldsymbol{Sym}(X)$ for all X with $|X| \geq M$, or $\ker(\varphi_X) = \{\mathrm{Id}_X\}$ for all X with $|X| \geq M$.*

By simple counting, if $\ker(\varphi_X) = \{\mathrm{Id}_X\}$, then $|\mathbf{Sym}(A_X)| \geq |\mathbf{Sym}(X)|$ and hence $|A_X| \geq |X|$. This yields the following corollary, which is one direction of Theorem 5.

▶ **Corollary 11.** *Let $\mathcal{A}$ be a syntactic FT_Σ-algebra of bounded complexity. There is an integer M such that, for every X with $|X| \geq M$,* $\ker(\varphi_X) = \boldsymbol{Sym}(X)$.

Heading toward the converse implication, we now show that when $\ker(\varphi_X) = \mathbf{Sym}(X)$, then $\langle a \rangle$ can only take very simple forms.

▶ **Lemma 12.** *Let $\mathcal{A}$ be a finite syntactic FT_Σ-algebra, and n be such that* $\ker(\varphi_X) = \boldsymbol{Sym}(X)$ *whenever $|X| \in \{n-1, n\}$. Then, for all $a \in A_X$ with $|X| = n$,*

$$\langle a \rangle(b) = \langle a \rangle(b')$$

for all $b, b' \in A_\emptyset^X$ such that $\mathrm{Im}(b) = \mathrm{Im}(b')$.

Proof. Note first that for $|X| \in \{n-1, n\}$ we have $\ker(\varphi_X) = \mathbf{Sym}(X)$. Hence for for all permutations $\sigma \in \mathbf{Sym}(X)$,

$$\langle a \rangle(b \circ \sigma) = \langle \sigma^{\mathcal{A}}(a) \rangle(b) = \langle a \rangle(b) \ . \tag{$\star$}$$

As a consequence, what matters is to prove that we can "duplicate" some $b(x)$'s. For $Y \subseteq X$ where $Y = \{y_0, ..., y_{k-1}\}$ as well as $h \in (A_\emptyset)^Y$ and $a \in A_X$, we simplify the notation with

$$a[h] = a[y_0 \leftarrow h(y_0), \ldots, y_{k-1} \leftarrow h(y_{k-1})] \ .$$

Let $x, y, z \in X$ be distinct variables and $b, b' \in (A_\emptyset)^X$ be such that b and b' coincide everywhere but for $b(z) = b(y)$ and $b'(z) = b(x)$, we claim that

$$\langle a \rangle(b) = \langle a \rangle(b') \ . \tag{$\star\star$}$$

Indeed, for σ that maps y to z and leaves all other variables unchanged, we have

$$\begin{aligned}
a[x \leftarrow d, y \leftarrow d', z \leftarrow d'][h] &= \sigma^{\mathcal{A}}(a)[x \leftarrow d, z \leftarrow d'][h] \\
&= \sigma^{\mathcal{A}}(a)[x \leftarrow d', z \leftarrow d][h] && \text{(by } (\star) \text{ applied to } X \setminus \{y\}) \\
&= a[x \leftarrow d', y \leftarrow d, z \leftarrow d][h] && \text{(by } (\star) \text{ applied to } X) \\
&= a[x \leftarrow d, y \leftarrow d', z \leftarrow d][h] \ .
\end{aligned}$$

in which $d, d' \in A_\emptyset$ and $h \in (A_\emptyset)^{X \setminus \{x,y,z\}}$, from which the claim $(\star\star)$ follows.

At this point, $(\star\star)$ allows to change the value $b(z)$ to another providing that its values before and after the change appear elsewhere in b, and $(\star)$ allows to permute all the $b(x)$'s. Hence, by iterative applications of them, we obtain of $(\star)$ and $(\star\star)$, $\langle a \rangle(b) = \langle a \rangle(b')$ providing that b and b' have same image. ◀

As a consequence of the above lemma, assuming that $\ker(\varphi_X) = \mathbf{Sym}(X)$ for all sufficiently large X, we can bound the number of possible elements in A_X. This yields Corollary 13 below, which is the second direction of Theorem 5.

▶ **Corollary 13.** *A finite syntactic FT_Σ-algebra such that* $\ker(\varphi_X) = \boldsymbol{Sym}(X)$ *for all sufficiently large set of variables X, has bounded complexity.*

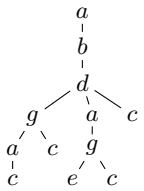
Proof. By Lemma 12, we know that, for $a \in A_X$, $\langle a \rangle$ must be chosen in a set of at most $|A_\emptyset|^{2^{|A_\emptyset|}}$ functions, this is an upper bound on the number of elements of A_X that does not depend on $|X|$ for X sufficiently large. ◀

4 Finite tree algebras of bounded complexity

The main theorem of this section, Theorem 14, provides a characterization of the languages recognized by FT_Σ-algebras of bounded complexity as Boolean combinations of simple languages. We proceed as follows. We state Theorem 14 and establish its easier parts in Section 4.1. In Section 4.2, we establish Lemma 22 which essentially amounts to the result for trees with "sufficiently many branches", which is the hardest part of the proof of Theorem 14.

4.1 Statement of the result

The main theorem of this section, Theorem 14, requires some preliminary definitions.



$\mathrm{upref}(t) = ab$ (unary prefix)
$\mathrm{fnu}(t) = d$ (first non-unary symbol)
$\mathrm{symb}(t) = \{a, b, c, d, e, g\}$ (symbols in t)
$\mathrm{pbsymb}(t) = \{a, c, e, g\}$ (post-branching symbols)

Figure 2 A finite tree t and its associated data.

For a given finite tree t, we associate to it some data (see Figure 2 for an illustration). Let n, be the depth of the first node labelled with a non unary symbol; formally, n is the least natural number such that $\mathrm{ar}(t(0^n)) \neq 1$. The *unary prefix* of t, denoted $\mathrm{upref}(t)$ is the word $t(0^1)...t(0^{n-1}) \in \Sigma_1^*$. The *first non-unary symbol of* t is $t(0^n)$, which we denote by $\mathrm{fnu}(t)$. The *set of symbols in* t is $\mathrm{symb}(t) = \{t(u) \mid u \in \mathrm{dom}(t)\}$ and its *set of post-branching symbols* is, if it exists, $\mathrm{pbsymb}(t) = \{t(0^n v) \mid 0^n v \in \mathrm{dom}(t), v \neq \varepsilon\}$.

▶ **Theorem 14.** *For a language of finite trees, the following properties are equivalent:*

1. *being recognized by a FT_Σ-algebra of bounded complexity,*
2. *having its syntactic FT_Σ-algebra of bounded complexity,*
3. *being equal to a Boolean combination of languages of the following kinds:*
   - **a.** *The language of finite trees with unary prefix in a given regular language of words $L \subseteq \Sigma_1^*$.*
   - **b.** *The language of finite trees with first non-unary symbol a for a fixed non-unary symbol a.*
   - **c.** *The language of finite trees with post-branching symbols B, for $B \subseteq \Sigma$.*
   - **d.** *A regular language K of* bounded branching*, meaning that there exists a natural number k such that all trees $t \in K$ have at most k branches.*

Let us establish the easiest parts of this statement. To start with, it is a generic fact– generic meaning independent of the type of algebras under consideration–that the syntactic FT_Σ-algebra of a language divides[1] any FT_Σ-algebra that recognizes the same language. Hence, each sort of the syntactic FT_Σ-algebra has a lesser size than the same sort in any other FT_Σ-algebra recognizing the same language. Thus 1 implies 2.

We now prove the second easiest implication, 3 implies 1. For this, it is sufficient to prove that the languages of kind 3a to 3d are recognized by FT_Σ-algebras of bounded complexity, and that FT_Σ-algebras of bounded complexity are closed under all the Boolean connectives. This is stated in Lemmas 15 to 17 below. All are straightforward.

[1] Divides in the morphism sense: being a quotient of a sub-algebra.

Given a regular language of words $L \subseteq \Sigma_1^*$, a non unary symbol a and a set $B \subseteq \Sigma$ of symbols, let us denote by $\mathrm{UPref}(L)$, $\mathrm{FNU}(a)$ and $\mathrm{PBSymb}(B)$, respectively the language of trees with unary prefix in L, the language of trees with first non-unary symbol c, and the language of trees t with $\mathrm{pbsymb}(t) = B$. Lemma 15 shows that these languages are recognized by algebras of bounded complexity, i.e. it treats Cases 3a to 3c.

▶ **Lemma 15.** *Given a regular language of words L, a non unary symbol $a \in \Sigma_{\neq 1}$ and a set of symbols $B \subseteq \Sigma$, the languages* $\mathrm{UPref}(L)$, $\mathrm{FNU}(a)$ *and* $\mathrm{PBSymb}(B)$ *are recognized by FT_Σ-algebras of bounded complexity.*

Proof. For space considerations, we only detail the case of $\mathrm{UPref}(L)$, which is arguably the hardest one. Let $\varphi\colon \Sigma_1^* \to M$ be a monoid morphism that recognizes L, meaning there exists $P \subseteq M$ such that $\varphi^{-1}(P) = L$. Define on $\mathrm{FiniteTrees}(\Sigma)$ the relation $\sim$ by $s \sim t$ if $\varphi(\mathrm{upref}(s)) = \varphi(\mathrm{upref}(t))$, this relation is easily seen to be a congruence, and $\mathrm{UPref}(L)$ is obviously recognized by the quotient algebra $\mathrm{FiniteTrees}(\Sigma)/\sim$. Because $\sim$ only has $|M|$ equivalence classes in every sort, we just described a FT_Σ-algebra recognizing $\mathrm{UPref}(L)$ of bounded complexity. ◀

Next we deal with the languages that have bounded branching, i.e. Case 3d. It is done with a modification of the automaton algebra of Example 3 so that it is also able to count the number of branches of a tree up to the bound k. The key observation is that a tree with k different variables must have at least k branches. This means that the sort A_X where X is a finite set of variables such that $|X| > k$ can be collapsed to one element only. Note that this is tightly related to our assumption that every variable from X must occur in all Σ, X-trees. We do not give any further detail here.

▶ **Lemma 16.** *The regular languages of finite trees of bounded branching are recognized by FT_Σ-algebras of bounded complexity.*

Finally, using standard constructions, we can provide the last ingredient of the proof that 3 implies 1 in Theorem 14:

▶ **Lemma 17.** *The languages recognized by FT_Σ-algebras of bounded complexity are closed under Boolean operations.*

4.2 Trees with many branches

In this section, we fill the missing gap in the proof of Theorem 14, namely the implication from 2 to 3.

Given two finite trees s and t and a FT_Σ-algebra $\mathcal{A}$ with evaluation morphism α, let $s \simeq_{\mathcal{A}} t$ hold if $\alpha(s) = \alpha(t)$. We omit the sub- and superscript $\mathcal{A}$ when it is clear from the context. Our goal is to prove Lemma 22 which states that if $\mathcal{A}$ is of bounded complexity, then for all trees s and t with sufficiently many branches, if $\mathrm{upref}(s) = \mathrm{upref}(t)$, $\mathrm{fnu}(s) = \mathrm{fnu}(t)$ and $\mathrm{pbsymb}(s) = \mathrm{pbsymb}(t)$, then $s \simeq_{\mathcal{A}} t$

Given an $\Sigma, X \uplus \{\gamma_1, \ldots, \gamma_n\}$-tree t in which the γ_i's only appear once, and given trees $t_1, \ldots, t_n$, we denote $t(t_1, \ldots, t_n)$ the tree $t[\gamma_1 \leftarrow t_1, \ldots, \gamma_n \leftarrow t_n]$. We will use these distinguished variables γ_i's in this section.

The results we prove throughout this section are consequences of the properties of permutations in FT_Σ-algebras of bounded complexity and, more particularly, consequences of Corollary 11. Our first objective is to show that if there are sufficiently many branches in a tree, it is possible to exchange any two subtrees which are not related by the ancestor

relationwithout changing evaluation in the algebra (see Lemma 20). Lemma 18 is a first step towards this goal: it says that, in a syntactic algebra, whenever a tree has many branches, it is possible to exchange *some* of its subtrees. More precisely:

▶ **Lemma 18.** *For every syntactic FT_Σ-algebras $\mathcal{A}$ of bounded complexity, there exists an integer N_0 such that, for all finite trees $t(t_1, t_2)$ such that t has at least N_0 branches,*

$$t(t_1, t_2) \simeq_{\mathcal{A}} t(t_2, t_1) \ .$$

Proof. Let N_0 be the integer introduced in Corollary 11. Assume that t is a $\Sigma, X \uplus \{\gamma_1, \gamma_2\}$-tree that has at least N_0 branches. Let s be a $\Sigma, X \uplus \{\gamma_1, \gamma_2\} \uplus \{x_1, ..., x_{N_0-2}\}$-tree and let s_0, ..., s_{N_0-2} be trees such that $s[x_1 \leftarrow s_1, ..., x_{N_0-2} \leftarrow s_{N_0-2}] = t$. As such, s has at least N_0 variables. Hence by Corollary 11, $t' \simeq_{\mathcal{A}} \sigma^{\mathcal{A}}(t')$ in which σ is the transposition of γ_1 and γ_2. We now compute:

$$\begin{aligned} t(t_1, t_2) &\simeq_{\mathcal{A}} s[\gamma_1 \leftarrow t_1, \gamma_2 \leftarrow t_2][x_1 \leftarrow s_1, ..., x_{N_0-2}] \\ &\simeq_{\mathcal{A}} \sigma^{\mathcal{A}}(s)[\gamma_1 \leftarrow t_1, \gamma_2 \leftarrow t_2][x_1 \leftarrow s_1, ..., x_{N_0-2}] \\ &= s[\gamma_1 \leftarrow t_2, \gamma_2 \leftarrow t_1][x_1 \leftarrow s_1, ..., x_{N_0-2}] = t(t_2, t_1) \ . \end{aligned}$$

◀

Here, notice the similarity between this proof and the observations we made in the proof of Lemma 12: this is the exact same argument, we just used the fact that a tree with many branches can always be obtained from some tree with many variables. Taking this similarity further, we may apply the other arguments we used in the proof of Lemma 12 to prove Lemma 19, and change the number of occurrences of plugged subtrees.

▶ **Lemma 19.** *For every syntactic FT_Σ-algebras $\mathcal{A}$ of bounded complexity, there exists an integer N_1 such that, for any finite tree $t(t_1, t_2, t_2)$ such that t has at least N_1 branches,*

$$t(t_1, t_2, t_2) \simeq_{\mathcal{A}} t(t_1, t_1, t_2) \ .$$

The next step is to establish Lemma 20, which is very similar to the previous Lemma 18 but for the fact that it is sufficient to have many branches in $t(t_1, t_2)$ instead of many branches in t to be allowed to swap t_1 and t_2. It is obtained by repeated and careful applications of Lemma 18.

▶ **Lemma 20.** *For all syntactic FT_Σ-algebras $\mathcal{A}$ of bounded complexity, there is an integer N_3 such that, for any finite tree $t(t_1, t_2)$ where $t(t_1, t_2)$ has at least N_3 branches,*

$$t(t_1, t_2) \simeq_{\mathcal{A}} t(t_2, t_1) \ .$$

As such, we may exchange two subtrees of a tree with many branches without changing evaluation in the algebra. We will use this result to prove that two trees with the same unary prefix, first non-unary symbol and set of post-branching symbols are not distinguished by the algebra (Lemma 22).

Using Lemmas 19 and 20, we first establish Lemma 21 that allows in some situation to make a symbol "appear" or "disappear".

▶ **Lemma 21.** *For all syntactic FT_Σ-algebras $\mathcal{A}$ of bounded complexity, there exists an integer N_4 that has the following property. For all finite trees $s(\gamma_1, \gamma_2)$, all finite trees t with at least N_4 branches, and all symbols $c, d \in \mathrm{symb}(t)$, c constant,*

$$s(t, c) \simeq_{\mathcal{A}} s(t, d(c, \ldots, c)) \ .$$

In combination with Lemmas 19 and 20, it means that under the same assumptions, $s(t, c) \simeq_{\mathcal{A}} s(t, t')$ for all finite trees t' that use only symbols appearing in t. This building block, used iteratively, allows to shuffle and change the number of occurrences of all symbols that appear below a first non-unary symbol as soon as there are sufficiently many branches. This is our key Lemma 22.

▶ **Lemma 22.** *Let $\mathcal{A}$ be a syntactic FT_Σ-algebra of bounded complexity. There is an integer N, such that, for all finite trees s and t, both of which have at least N branches, if* $\mathrm{upref}(s) = \mathrm{upref}(t)$, $\mathrm{fnu}(s) = \mathrm{fnu}(t)$ *and* $\mathrm{pbsymb}(t) = \mathrm{pbsymb}(t)$, *then $t \simeq_{\mathcal{A}} s$.*

Putting everything together we establish the last implication in Theorem 14:

▶ **Lemma 23.** *A language of finite trees L recognized by a FT_Σ-algebra of bounded complexity can be written as a Boolean combination of languages of kinds 3a-3d in Theorem 14.*

The idea behind this last proof is as follows. Given a regular word language $K \subseteq \Sigma_1^*$, a non-unary symbol a, and a set of symbols B, let $L_{K,a,B}$ be the set of trees t such that $\mathrm{upref}(t) \in K$, $\mathrm{fnu}(t) = a$ and $\mathrm{pbsymb}(t) = B$. Such languages can be written as the intersection of $\mathrm{UPref}(K)$, $\mathrm{FNU}(a)$ and $\mathrm{PBSymb}(B)$ which are of the kinds 3a-3d in Theorem 14. Let N be the value from Lemma 22.

In a first step, we construct finitely many tuples (K_i, a_i, B_i), such that L and $\bigcup_i L_{K_i,a_i,B_i}$ coincide over all trees with sufficiently many branches. For this, for all $t \in L$ with at least N branches, consider the least language $K_t \subseteq \Sigma_1^*$ recognized by $A_{\{x\}}$ such that $\mathrm{upref}(t) \in K_t$ (in which $\mathrm{upref}(t)$ is recognized by $A_{\{x\}}$ when seen as a tree made of unary symbols and the variable x). The language K_t is regular and has the property that exchanging the unary prefix of t for any other word in K_t leaves the tree in L. We use as the (K_i, a_i, B_i)'s all the tuples $(K_t, \mathrm{fnu}(t), \mathrm{pbsymb}(t))$ for t ranging over the trees in L with at least N branches (note that since all the K_t's are recognized by $A_{\{x\}}$, there are finitely many of them). One can show using Lemma 22 that, as claimed, L and $\bigcup_i L_{K_i,a_i,B_i}$ coincide over all trees with at least N branches. In a second step one defines L_i to be the set of trees in L_{K_i,a_i,B_i} that have at least N branches. Let also L' be L restricted to trees with less than N branches. One gets

$$L = L' \cup \bigcup_i L_i \ ,$$

and this is by construction a Boolean combination of languages of the kinds 3a-3d.

This concludes our proof of Lemma 23, and hence Theorem 14.

5 Conclusion

In this paper, we initiated a complexity analysis of the expressiveness of infinitely sorted algebras. Our main result gives a descriptive characterization of the languages of finite trees recognized by algebras of bounded complexity, Theorem 14. In this work, we made a design choice in the definition of tree algebras. Indeed, we require that in a tree of sort X, every variable occurs at least once. Removing this assumption would change our bounded complexity characterization result, yielding only Boolean combinations of languages of the form "the root symbol is a". Another possible variant is to allow trees restricted to a single variable: in this case our results remain unchanged.

Extensions to infinite trees. We also obtained a similar characterization for algebras for infinite trees. We did not include it in this short abstract for space considerations (this was in fact our original question). In this case, the algebras have to include an extra iterating construct that allows to build all infinite regular trees (i.e. unfolding of finite graphs). By Rabin's lemma, regular languages of infinite trees are entirely characterized by the regular trees they contain, and as a consequence such algebras describe regular languages of infinite trees. Our result characterizes such algebras of bounded complexity along the same line as Theorem 14. We show that over infinite trees, such algebras can express two extra things (1) the existence of subtrees of unary shape that belong to a prescribed prefix-invariant regular language of infinite words, and (2) the existence of subtrees in which a set of letters C appears densely, i.e. every letter in C appears in every subtree.

Future work. The next simplest cases seem to be the algebras of polynomial complexity and of bounded orbit complexity. An example in this case is the language of Σ-trees such that the leftmost branch does not contain the symbol a (we assume the existence of other symbols of arity 0, and at least 2). It is recognized by an algebra which is both of polynomial complexity and bounded orbit complexity. So far, we do not even know whether these two classes differ.

References

1 Achim Blumensath. Recognisability for algebras of infinite trees. *Theor. Comput. Sci.*, 412(29):3463–3486, 2011. doi:10.1016/j.tcs.2011.02.037.

2 Achim Blumensath. Regular tree algebras. *Logical Methods in Computer Science*, 16, 2020.

3 Mikolaj Bojanczyk and Bartek Klin. A non-regular language of infinite trees that is recognizable by a sort-wise finite algebra. *Log. Methods Comput. Sci.*, 15(4), 2019. URL: https://lmcs.episciences.org/5927, doi:10.23638/LMCS-15(4:11)2019.

4 Mikolaj Bojanczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 407–416. ACM, 2016. doi:10.1145/2933575.2934508.

5 Mikolaj Bojanczyk and Luc Segoufin. Tree languages defined in first-order logic with one quantifier alternation. *Log. Methods Comput. Sci.*, 6(4), 2010. doi:10.2168/LMCS-6(4:1)2010.

6 Mikolaj Bojanczyk, Luc Segoufin, and Howard Straubing. Piecewise testable tree languages. *Log. Methods Comput. Sci.*, 8(3), 2012. doi:10.2168/LMCS-8(3:26)2012.

7 Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2008.

8 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

9 Zoltán Ésik and Pascal Weil. Algebraic recognizability of regular tree languages. *Theor. Comput. Sci.*, 340(1):291–321, 2005. doi:10.1016/j.tcs.2005.03.038.

10 Zoltán Ésik and Pascal Weil. Algebraic characterization of logically defined tree languages. *Int. J. Algebra Comput.*, 20(2):195–239, 2010. doi:10.1142/S0218196710005595.

11 Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.