ELSEVIER Theoretical Computer Science 154 (1996) 85–106

Theoretical Computer Science

An algebraic characterization of frontier testable tree languages

Thomas Wilke*,[1]

Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 24098 Kiel, Germany

Abstract

The class of frontier testable (i.e., reverse definite) tree languages is characterized by a finite set of pseudoidentities for tree algebras, which are introduced here for this characterization. An efficient algorithm is presented that decides whether a given tree automaton recognizes a frontier testable tree language. The algorithm runs in time $O(mn^3 + m^2n^2)$, where m is the cardinality of the alphabet and n is the number of states of the automaton.

In this paper a tree language is said to be frontier testable[2] if there exists a natural number k such that

(*) any two trees with the same set of subtrees of depth at most k either both belong to the language in question or both do not belong to it.

A language is called k-frontier testable, if (*) is satisfied.

The first result of this paper is an algebraic characterization of the class of frontier testable tree languages. For this purpose, tree algebras[3] are introduced, and with every tree language a syntactic tree algebra is associated. The algebraic characterization of frontier testability then consists of four pseudoidentities defining a class of tree algebras which contains a syntactic tree algebra of a tree language if and only if this language is frontier testable. For each natural number k a set of identities is given that characterizes k-frontier testability, and from these sets of identities the pseudoidentities for the general case are derived.

The problem of characterizing classes of regular tree languages in algebraic terms has been addressed in several papers, see, e.g., [10, 12, 13, 16]. Two approaches have

* E-mail: tw@informatik.uni-kiel.d400.de.

[1] Supported by the ESPRIT BRA Working Group No. 6317, ASMICS 2.

[2] This is often called 'reverse definite', see [8] and [16].

[3] In the conference version of this paper [18], tree algebras were introduced as 'point-tree algebras'.

been pursued: a semigroup-based and a universal algebraic one. By introducing tree algebras, this paper suggests a combined approach. At least in the case of frontier testability this approach seems more appropriate than the universal-algebraic approach and is superior to the semigroup-theoretical approach, because frontier testability cannot be characterized using semigroup-based methods (see concluding remarks).

The second result of this paper is an efficient algorithm that decides whether the tree language recognized by a given tree automaton is frontier testable or not. The algorithm expects a deterministic frontier-to-root tree automaton as input and runs in time $\mathcal{O}(mn^3 + m^2n^2)$, where m is the cardinality of the alphabet and n is the number of states of the input automaton. This improves the exponential upper bound which is obtained by a straightforward implementation of the decision procedure described in [8].

The combinatorial properties of frontier testable tree languages used in Section 6 were, in a weaker form, established jointly with Scholz [15]. For background on universal algebra and many-sorted universal algebra, the reader is referred to [4] and [7], respectively.

1. Regular tree languages

Throughout this paper we fix an arbitrary finite alphabet and denote it by A. We assume that Γ is a finite signature that contains one constant symbol c_a and one binary function symbol f_a for every $a \in A$.

By a *tree over* A we mean a ground term in Γ. The set of all trees over A is denoted by T. It is the universe of a term algebra that we will denote by $\mathbf{T}$. This algebra is the initial Γ-algebra, i.e., for every Γ-algebra $\mathbf{B}$ there exists a unique homomorphism $\mathbf{T} \to \mathbf{B}$, which we will denote by $h_{\mathbf{B}}$.

By a *tree language over* A we mean a subset of T. A *tree automaton over* A is a pair $(\mathbf{B}, R)$ where $\mathbf{B}$ is a finite Γ-algebra and $R \subseteq B$. It is said that $(\mathbf{B}, R)$ *recognizes* the tree language $h_{\mathbf{B}}^{-1}(R)$. A tree language over A is called *regular* if there exists a tree automaton over A that recognizes the language.

If $(\mathbf{B}, R)$ is a tree automaton, the elements of B are called the *states of* $(\mathbf{B}, R)$. A state $q \in B$ is called *reachable*, if it belongs to the image of $h_{\mathbf{B}}$, i.e., if it belongs to the universe of the substructure $\mathbf{B}'$ of $\mathbf{B}$ which is generated by the empty set. Clearly, $(\mathbf{B}', R \cap B')$ is a tree automaton, it has only reachable states, and it recognizes the same language as $(\mathbf{B}, R)$. So, without loss of generality, we may assume that every tree automaton has only reachable states, and we will do so throughout this paper. Notice that the construction of $(\mathbf{B}', R \cap B')$ can be carried out in time $O(mn^2)$ where m is the cardinality of the alphabet and $n = |B|$.

The purpose of the next two sections is to develop an alternative algebraic framework for the treatment of regular tree languages, which we will gain advantage from when we will try to characterize the class of frontier testable tree languages.

2. Tree algebras

We are aiming at the definition of the notion of a tree algebra. For this purpose we introduce some operations on trees and related objects (special trees) and study the algebraic laws obeyed by these operations.

We start with the observation that $f_a(t,t')$ is a tree over A, if a is a letter of A and if t and t' are trees over A. We therefore obtain an operation $\kappa^A : A \times T \times T \to T$ by setting $\kappa^A(a,t,t') = f_a(t,t')$.

An even simpler way to produce a tree over A is to take a letter a of the alphabet and view this letter as the one-node tree c_a. According to this we define the operation $\iota^A : A \to T$ by $\iota^A(a) = c_a$.

For the rest of the paper we fix a variable x. By a *special tree over A* (see [17]) we mean a Γ-term in x with exactly one occurrence of x. The set of all special trees over A except x is denoted by S. We write S' for $S \cup \{x\}$, i.e., S' stands for the set of all special trees over A. If s denotes a special tree over A and if t stands for a special or an ordinary tree, then st denotes the special or ordinary tree which is obtained by substituting t for x in s.

On the set S we define a binary operation σ^A by $\sigma^A(s,s') = ss'$. Due to the associativity of the substitution operation, σ^A is associative, i.e., the following law is obeyed by σ^A.

$$\sigma^A(s,\sigma^A(s',s'')) = \sigma^A(\sigma^A(s,s'),s'') \tag{1}$$

In a similar way we define $\eta^A : S \times T \to T$ by setting $\eta^A(s,t) = st$ for $s \in S$ and $t \in T$. In accordance with (1) we have the following 'mixed' law of associativity for σ^A and η^A:

$$\eta^A(s,\eta^A(s',t)) = \eta^A(\sigma^A(s,s'),t) \tag{2}$$

To construct special trees we finally introduce two functions $\rho^A : A \times T \to S$ and $\lambda^A : A \times T \to S$ symmetric to each other by setting $\rho^A(a,t) = a(t,x)$ and $\lambda^A(a,t) = a(x,t)$, i.e., ρ^A produces special trees with x as the *right* successor of the root and λ^A with x as the *left* successor of the root. Obviously, the following two laws are obeyed by these new operations.

$$\eta^A(\lambda^A(a,t),t') = \kappa^A(a,t',t) \tag{3}$$

$$\eta^A(\rho^A(a,t),t') = \kappa^A(a,t,t') \tag{4}$$

In Fig. 1, a graphical illustration of the six functions introduced so far is given.

We now join together A, S, and T and ι^A, ρ^A, λ^A, κ^A, σ^A, and η^A to make a 3-sorted structure. As identifiers for the three sorts we use the letters A, S, and T, i.e., if a 3-sorted set is denoted by M, it is a $\{\mathrm{A},\mathrm{S},\mathrm{T}\}$-indexed family of disjoint sets, which are referred to by M_A, M_S, and M_T. We also write M in the form $(M_\mathrm{A},M_\mathrm{S},M_\mathrm{T})$. If a function symbol from a corresponding 3-sorted signature is to map, say, pairs of elements with first entry of sort S and second entry of sort T to elements of sort T,

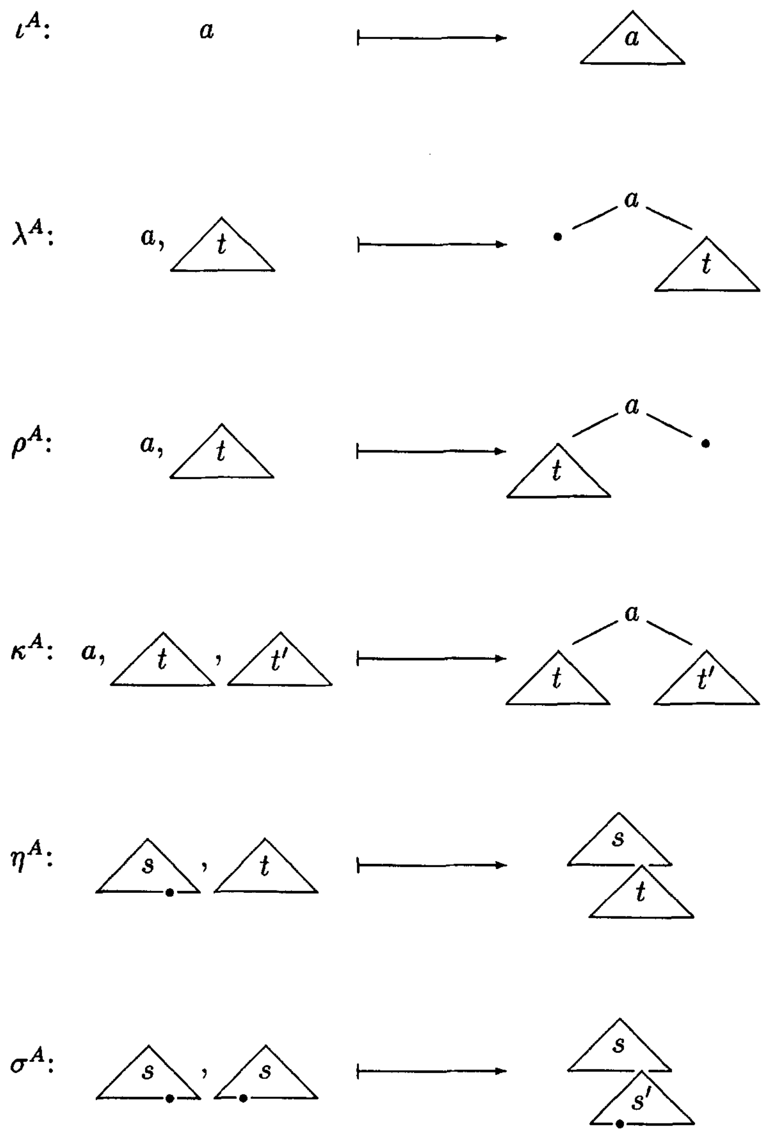

Fig. 1. The six operations on A, S, and T.

we indicate this by the subscript (ST,T). So $\eta_{(\mathrm{ST,T})}$, when interpreted in a corresponding structure $\mathbf{M}$ (with 3-sorted universe M), becomes a function $M_\mathrm{S} \times M_\mathrm{T} \to M_\mathrm{T}$.

The sets A, S, and T together with the six operations ι^A, ρ^A, λ^A, κ^A, σ^A, and η^A form a 3-sorted Σ-structure, where Σ is the 3-sorted signature given by

$$\Sigma = \{\iota_{(\mathrm{A,T})}, \lambda_{(\mathrm{AT,S})}, \rho_{(\mathrm{AT,S})}, \kappa_{(\mathrm{ATT,T})}, \sigma_{(\mathrm{SS,S})}, \eta_{(\mathrm{ST,T})}\}.$$

This structure is denoted by **F**, and as we have seen (see (1)–(4)) the following four identities hold in **F**:

$$\sigma(s,\sigma(s',s''))=\sigma(\sigma(s,s'),s'') \tag{5}$$

$$\eta(s,\eta(s',t))=\eta(\sigma(s,s'),t) \tag{6}$$

$$\eta(\lambda(a,t),t')=\kappa(a,t',t) \tag{7}$$

$$\eta(\rho(a,t),t')=\kappa(a,t,t') \tag{8}$$

Finally, we can give the definition which is fundamental for this paper.

Definition 1. A 3-sorted Σ-algebra satisfying (5)–(8) is called a *tree algebra.*

We observe the following.

Remark 1. If **B** is a tree algebra, then the set B_{S} together with the binary operation $\sigma^{\mathbf{B}}$ is a semigroup.

The following proposition, which we will need in the next subsection, states that **F** is fully determined by (5)–(8).

Proposition 1. *The tree algebra* **F** *is freely generated by the 3-sorted set* $(A,\emptyset,\emptyset)$ *in the variety of tree algebras.*

Proof. Clearly, **F** is generated by $(A,\emptyset,\emptyset)$; what remains to be shown is that for every tree algebra **B** every function $h_0: A \to B_{\mathrm{A}}$ can be extended to a tree algebra homomorphism $h: \mathbf{F} \to \mathbf{B}$.

Assume, therefore, that **B** is an arbitrary tree algebra and h_0 an arbitrary function $A \to B_{\mathrm{A}}$. We first define a 3-sorted function $h: F \to B$ that extends h_0 and then show that it is a homomorphism. On S and T the function h is defined by induction using the following rules:

$$h(c_a)=\iota^{\mathbf{B}}(h_0(a)) \tag{9}$$

$$h(f_a(t,t'))=\kappa^{\mathbf{B}}(h_0(a),h(t),h(t')) \tag{10}$$

$$h(f_a(x,t))=\lambda^{\mathbf{B}}(h_0(a),h(t)) \tag{11}$$

$$h(f_a(s,t))=\sigma^{\mathbf{B}}(\lambda^{\mathbf{B}}(h_0(a),h(t)),h(s)) \tag{12}$$

$$h(f_a(t,x))=\rho^{\mathbf{B}}(h_0(a),h(t)) \tag{13}$$

$$h(f_a(t,s))=\sigma^{\mathbf{B}}(\rho^{\mathbf{B}}(h_0(a),h(t)),h(s)) \tag{14}$$

The compatibility of h with ι, κ, λ, and ρ can immediately be read off from (9), (10), (11), and (13), respectively.

Proving the compatibility with σ is much more complicated. We do this by induction according to the inductive definition of h. Assume first that $s = f_a(x,t)$ with $a \in A$ and

$t \in T$ and that s' is an arbitrary element of S. Using the definition of $\sigma^{\mathbf{F}}$, the assumption about s, and (12) we obtain the following chain of equations:

$$\begin{aligned}
h(\sigma^{\mathbf{F}}(s,s')) &= h(ss') && \text{(definition of } \sigma^{\mathbf{F}}\text{)}\\
&= h(f_a(s',t)) && \text{(assumption about } s\text{)}\\
&= \sigma^{\mathbf{B}}(\lambda^{\mathbf{B}}(h_0(a),h(t)),h(s')) && \text{(because of (12))}\\
&= \sigma^{\mathbf{B}}(h(\lambda^{\mathbf{F}}(a,t)),h(s')) && \text{(compatibility with } \lambda\text{)}\\
&= \sigma^{\mathbf{B}}(h(s),h(s')) && \text{(definition of } \lambda^{\mathbf{F}}\text{)}
\end{aligned}$$

The proof is symmetric if s is of the form $f_a(t,x)$. For the induction step let $s = f_a(s'',t)$ and a, t, and s' as before. Then $h(\sigma^{\mathbf{F}}(s,s')) = \sigma^{\mathbf{B}}(h(s),h(s'))$ is shown by the following chain of equations.

$$\begin{aligned}
h(\sigma^{\mathbf{F}}(s,s')) &= h(ss') && \text{(definition of } \sigma^{\mathbf{F}}\text{)}\\
&= h(f_a(s''s',t)) && \text{(assumption about } s\text{)}\\
&= h(f_a(\sigma^{\mathbf{F}}(s'',s'),t)) && \text{(definition of } \sigma^{\mathbf{F}}\text{)}\\
&= \sigma^{\mathbf{B}}(\lambda^{\mathbf{B}}(h_0(a),h(t)),h(\sigma^{\mathbf{F}}(s'',s'))) && \text{(because of (12))}\\
&= \sigma^{\mathbf{B}}(\lambda^{\mathbf{B}}(h_0(a),h(t)),\sigma^{\mathbf{B}}(h(s''),h(s'))) && \text{(induction hypothesis)}\\
&= \sigma^{\mathbf{B}}(\sigma^{\mathbf{B}}(\lambda^{\mathbf{B}}(h_0(a),h(t)),h(s'')),h(s')) && \text{(because of (5))}\\
&= \sigma^{\mathbf{B}}(h(f_a(s'',t)),h(s')) && \text{(because of (12))}\\
&= \sigma^{\mathbf{B}}(h(s),h(s')) && \text{(assumption about } s\text{)}
\end{aligned}$$

The proof is symmetric if s is of the form $f_a(t,s'')$.

The compatibility of h with η is shown in a similar way. By induction on the structure of s one proves $h(\eta^{\mathbf{F}}(s,t)) = \eta^{\mathbf{B}}(h(s),h(t))$. Instead of (5) one uses (6); depending on whether x occurs in the left or right subtree of s one uses either (7) or (8). □

3. A characterization of regularity in terms of tree algebras

A tree algebra $\mathbf{B}$ is said to *recognize* a tree language L over A, if there exists a homomorphism $h \colon \mathbf{F} \to \mathbf{B}$ and a set $R \subset B_{\mathrm{T}}$ such that $h^{-1}(R) = L$. We will prove in this section that a tree language is regular if and only if it is recognized by a finite tree algebra.

For the direction from right to left assume that $\mathbf{B}$ is a finite tree algebra that recognizes a tree language L over A, say, $h \colon \mathbf{F} \to \mathbf{B}$ is a homomorphism such that

$h^{-1}(R) = L$ for some $R \subset B_{\mathrm{T}}$. From h and $\mathbf{B}$ we obtain a tree automaton $(\mathbf{C}, R)$ by the following definitions:

$$C = B_{\mathrm{T}} \tag{15}$$

$$c_a^{\mathbf{C}} = h(c_a) \tag{16}$$

$$f_a^{\mathbf{C}}(q,q') = \kappa^{\mathbf{B}}(h(a),q,q') \tag{17}$$

By induction on the set T, using the fact that h is a homomorphism, one proves $h_{\mathbf{C}}(t) = h(t)$ for every $t \in T$, hence $h_{\mathbf{C}}^{-1}(R) = h^{-1}(R) = L$, thus $(\mathbf{C}, R)$ recognizes L. What we have proved is the following.

Remark 2. If L is a tree language over A recognized by a finite tree algebra, L is regular.

To proof the converse of this remark let us assume that L is a tree language over A which is recognized by a tree automaton $(\mathbf{B}, R)$. We will construct a finite tree algebra $\mathbf{C}$ that recognizes L.

Consider the 3-sorted set $C = (A, B^B, B)$, which is turned into a 3-sorted Σ-algebra $\mathbf{C}$ by the following definitions:

$$\iota^{\mathbf{C}}(a) = h_{\mathbf{B}}(c_a) \tag{18}$$

$$\kappa^{\mathbf{C}}(a,q,q') = f_a^{\mathbf{B}}(q,q') \tag{19}$$

$$\sigma^{\mathbf{C}}(f,f') = f \circ f' \tag{20}$$

$$\eta^{\mathbf{C}}(f,q) = f(q) \tag{21}$$

$$\lambda^{\mathbf{C}}(a,q) = \{(q', f_a^{\mathbf{B}}(q',q)) \mid q' \in B\} \tag{22}$$

$$\rho^{\mathbf{C}}(a,q) = \{(q', f_a^{\mathbf{B}}(q,q')) \mid q' \in B\} \tag{23}$$

In (20), $f \circ f'$ denotes the composition of functions, i.e., $f \circ f'(q) = f(f'(q))$ for every $q \in B$.

The algebra $\mathbf{C}$ satisfies (5) and (6) because of the associativity of the composition of functions; it satisfies (7), since we have $\eta^{\mathbf{B}}(\lambda^{\mathbf{B}}(a,q),q') = f_a^{\mathbf{B}}(q',q) = \kappa^{\mathbf{C}}(a,q',q)$, where the first equality is due to (22) and the second is due to (19). A symmetric argument would prove that $\mathbf{C}$ obeys (8). Thus $\mathbf{C}$ is a tree algebra. Therefore, using Proposition 1, there exists a homomorphism $h \colon \mathbf{F} \to \mathbf{C}$ which extends h_0 with $h_0(a) = a$ for every $a \in A$. By induction on the structure of the trees over A one can show that $h(t) = h_{\mathbf{B}}(t)$ for every $t \in T$. This implies $h^{-1}(R) = L$, i.e., $\mathbf{C}$ recognizes L. This yields the following.

Remark 3. If L is a regular tree language over A, then L is recognized by a finite tree algebra.

Putting Remarks 2 and 3 together we obtain the desired result.

Proposition 2. *A tree language L over A is regular if and only if L is recognized by a finite tree algebra.*

Algebraic characterizations of regular tree languages in terms of monoids and semigroups have been given in, e.g., [5, 8, 13].

4. The notion of a syntactic tree algebra

For every regular tree language L there exists, up to isomorphism, a smallest (with respect to the number of states) tree automaton that recognizes L. This tree automaton is usually called *the* minimal automaton of L and considered to be a canonic object for L. The Γ-algebra of this automaton is obtained as a quotient of $\mathbf{T}$ by an appropriate congruence, which is called the syntactic congruence of L.

The concept of syntactic congruence was first introduced in the theory of regular word languages and was later adapted to the tree case. Meanwhile, it has been recognized as a natural concept when dealing with recognizable sets (inverse images of homomorphisms into finite algebra) in any context, see, e.g., [2] for a treatment in a universal algebraic but one-sorted framework. The concept generalizes in a natural way to the many-sorted case; the basic properties of the notion are preserved. The application to the case of tree algebras and tree languages leads to the following definition.

The *syntactic tree algebra congruence* of a tree language L over A is the 3-sorted binary relation $\sim^L$ defined as follows:

$$\begin{aligned}\sim^L_{\mathrm{A}} = \{(a,a') \in A \times A \mid (a \in L \leftrightarrow a' \in L) \\ \wedge \forall s \in S'\ \forall t,t' \in T\ (sf_a(t,t') \in L \leftrightarrow sf_{a'}(t,t') \in L)\}\end{aligned} \tag{24}$$

$$\sim^L_{\mathrm{S}} = \{(s,s') \in S \times S \mid \forall s'' \in S'\ \forall t \in T\ (s''st \in L \leftrightarrow s''s't \in L)\} \tag{25}$$

$$\sim^L_{\mathrm{T}} = \{(t,t') \in T \times T \mid \forall s \in S'\ (st \in L \leftrightarrow st' \in L)\} \tag{26}$$

So two elements are equivalent if and only if they (are of the same sort and) relate to L in the same way in every possible context.

The basic properties of $\sim^L$, as they can be derived from the general results on syntactic congruences are summed up in the following lemma.

Lemma 1. *Let L be a tree language over A.*

1. *$\sim^L$ is a congruence relation on* $\mathbf{F}$.
2. *$\sim^L$ is the largest (with respect to set inclusion) congruence such that L is a union of classes.*
3. *L is recognized by a finite tree algebra iff $\sim^L$ has finite index (i.e., iff $\sim^L$ has a finite number of congruence classes).*

As a consequence we note the following.

Proposition 3. *Let L be a tree language over A. Then the following conditions are equivalent*:

(A) *L is regular*

(B) *$\mathbf{F}/{\sim^L}$ is finite*

(C) *$\sim^L$ has finite index*

Proof. The equivalence of (B) and (C) is trivial. For the implication from (A) to (B) let L be a regular tree language over A. Then, by Proposition 2, it is recognized by a finite tree algebra, say, $L = h^{-1}(R)$ for a homomorphism $h\colon \mathbf{F} \to \mathbf{B}$ into a finite tree algebra $\mathbf{B}$ and some $R \subseteq B_{\mathrm{T}}$. Since $\mathbf{B}$ is finite, $\ker(h)$ has a finite index; since $h^{-1}(R) = L$, the language L is a union of classes of $\ker(h)$. Then, by Lemma 1, we know that $\sim^L$ is at least as large as $\ker(h)$, hence $\sim^L$ is of finite index too.

For the implication from (B) to (A) let $\sim^L$ have finite index. Then $\mathbf{F}/{\sim^L}$ is a finite tree algebra, which, by Lemma 1, recognizes L. (Consider the natural homomorphism from $\mathbf{F}$ to $\mathbf{F}/{\sim^L}$.) □

Definition 2. For every tree language L over A the quotient structure $\mathbf{F}/{\sim^L}$ is called the *syntactic tree algebra of L over A*.

5. Frontier testable tree languages

A language of finite words is reverse definite if membership is determined by the prefix of a fixed maximal length of a given word. In the case of tree languages (reading trees from front to root) this corresponds to the set of frontier trees of a fixed maximal depth.

The set of *frontier trees* of a given tree t (either ordinary or special) is defined as follows. If $t \in \{c_a \mid a \in A\} \cup \{x\}$, then $\mathrm{front}(t) = t$. If $t = f_a(t', t'')$, then

$$\mathrm{front}(t) = \mathrm{front}(t') \cup \mathrm{front}(t'') \cup \{f_a(t', t'')\}.$$

The *depth* of a tree t is inductively defined too. If $t \in \{c_a \mid a \in A\} \cup \{x\}$, then $\mathrm{depth}(t) = 1$. If $t = f_a(t', t'')$, then

$$\mathrm{depth}(t) = \max\{\mathrm{depth}(t'), \mathrm{depth}(t'')\} + 1.$$

Let k be a natural number. The set of frontier trees of depth less than or equal to k of a tree t is now defined by $\mathrm{front}_k(t) = \{t' \in \mathrm{front}(t) \mid \mathrm{depth}(t') \leqslant k\}$.

We observe the following.

Remark 4. Let t be an arbitrary tree.

(1) The tree t has depth $\geqslant k$ iff there is a tree of depth k in $\mathrm{front}_k(t)$.

(2) If the tree t has depth $< k$, there is a unique tree of maximal depth in $\mathrm{front}_k(t)$, namely t itself, and $\mathrm{front}_k(t) = \mathrm{front}(t)$.

Definition 3. A tree language L over A is *k-frontier testable* if, for $t, t' \in T$ with $\mathrm{front}_k(t) = \mathrm{front}_k(t')$, either $t, t' \in L$ or $t, t' \notin L$. The language L is *frontier testable* if it is k-frontier testable for some k.

6. An equational characterization of k-frontier testability

In this section let k be a fixed positive natural number.

We introduce some conventions concerning the notation of terms in Σ. For $\kappa(a, \phi_0, \phi_1)$ we write $a(\phi_0, \phi_1)$, for $\sigma(\psi, \psi')$ and for $\eta(\psi, \phi)$ we write $\psi\psi'$ and $\psi\phi$, respectively. (Note that because of the sorts it is always clear whether σ or η is meant.)

For notational convenience the term $s_k \ldots s_2 t_1$ (or, to be precise, the term $\eta(s_k, \eta(s_{k-1}, \ldots \eta(s_2, t_1) \ldots)))$ is denoted by τ_k; it captures exactly all trees of depth $\geqslant k$. This is expressed in the following remark, in which $\hat{\mathbf{F}}$ stands for the free 3-sorted Σ-term algebra over the (countable) set of all variables that are used in this paper to write equations in Σ. In particular, this set of variables includes $s_2, s_3, \ldots$ and t_1, so F includes τ_k for every k.

Remark 5. An ordinary tree $t \in T$ has $\mathrm{depth}(t) \geqslant k$ iff there is a homomorphism $h : \hat{\mathbf{F}} \to \mathbf{F}$ such that $h(\tau_k) = t$.

An equational characterization of k-frontier testability is the main result of this section.

Theorem 1. *Let L be a regular tree language over the alphabet A. Then the following conditions are equivalent.*

(A) *L is k-frontier testable,*

(B) *The syntactic tree algebra of L satisfies the following equations, which are denoted by* (FT_k):

$$a(\tau_k, t) = b(t, \tau_k) \qquad (\mathrm{Sym}_k)$$

$$a(\tau_k, \tau_k) = \tau_k \qquad (\mathrm{Idp}_k)$$

$$a(t, a(st, \tau_k)) = a(st, \tau_k) \qquad (\mathrm{Can}_k)$$

$$a(a(t, \tau_k), t') = a(t, a(\tau_k, t')) \qquad (\mathrm{Rot}_k)$$

(C) *L is recognized by a finite tree algebra satisfying* (FT_k).

An illustration of the four equations is given in Fig. 2. These pictures also explain the names of the equations: (Sym_k) stands for 'symmetry', (Idp_k) for 'idempotence', (Can_k) for 'cancellation', and (Rot_k) for 'rotation'. Equation (Sym_k) says that a tree

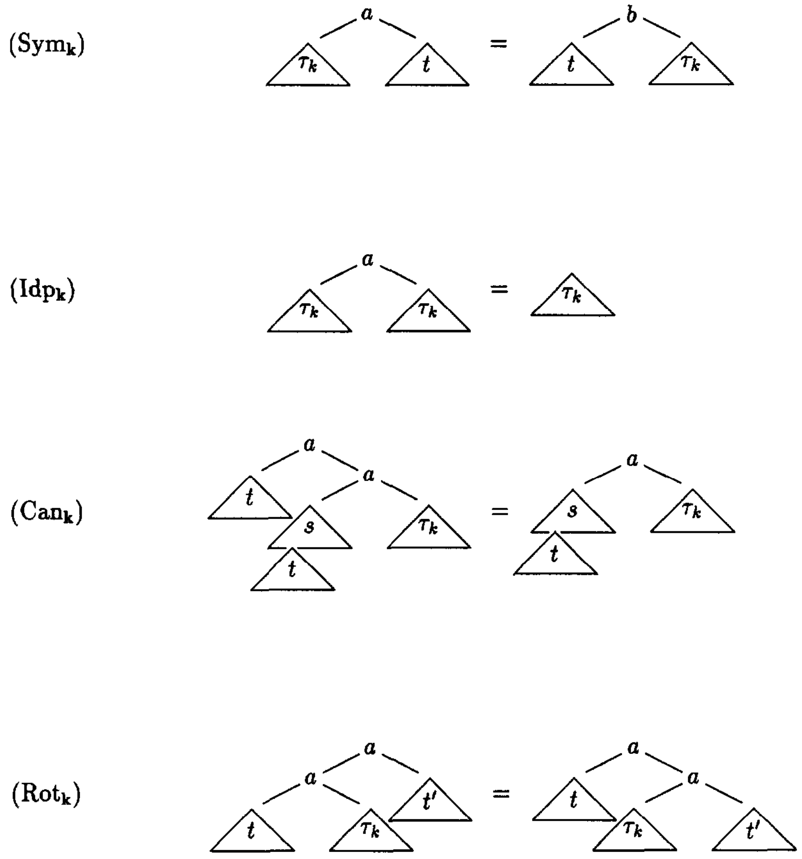

Fig. 2. Graphical illustration of (FT$_k$).

of depth $\geqslant k$ can be turned around its root. Equation (Idp$_k$) allows a duplication of a tree of depth $\geqslant k$, (Can$_k$) allows the elimination of the occurrence of a frontier tree which also occurs in the 'neighbourhood' of a tree of depth $\geqslant k$. Finally (Rot$_k$) expresses that trees can be rotated around a tree of depth $\geqslant k$.

We mention the following result for later use.

Lemma 2. *Every tree algebra in which (Sym$_k$), (Idp$_k$), and (Rot$_k$) hold, satisfies the following equation*:

$$a(t, a(t, \tau_k)) = a(t, \tau_k). \qquad (\mathrm{Can}'_k)$$

Proof. The assertion is proved by the following chain of equations:

$$\begin{aligned} a(t,\tau_k) &= a(a(t,\tau_k), a(t,\tau_k)) && \text{(because of (Idp}_k\text{))} \\ &= a(t, a(\tau_k, a(t,\tau_k))) && \text{(because of (Rot}_k\text{))} \\ &= a(t, a(a(t,\tau_k),\tau_k)) && \text{(because of (Sym}_k\text{))} \end{aligned}$$

$= a(t, a(t, a(\tau_k, \tau_k)))$ (because of (Rot_k))

$= a(t, a(t, \tau_k))$ (because of (Idp_k)) □

The remainder of this section is devoted to the proof of Theorem 1. The implication from (B) to (C) follows immediately from Lemma 1. For the implication from (C) to (B) let L be recognized by a finite tree algebra $\boldsymbol{B}$ satisfying (FT_k), say, $L = h^{-1}(R)$ for a homomorphism $h\colon \mathbf{F} \to \mathbf{B}$ and $R \subseteq B_{\mathrm{T}}$. By Lemma 1 the congruence $\sim^L$ is at least as great as $\ker(h)$. Thus $\mathbf{F}/\sim^L$ satisfies every equation that $\mathbf{F}/\ker(h)$ satisfies. Since $\mathbf{F}/\ker(h)$ is a subalgebra of $\mathbf{B}$ and $\mathbf{B}$ satisfies (FT_k) we obtain that $\mathbf{F}/\sim^L$ satisfies (FT_k).

The more difficult parts are the implication from (A) to (C), i.e., the 'correctness' of (FT_k), and the implication from (C) to (A), i.e., the 'completeness' of (FT_k).

6.1. Correctness

We have to show that for every k-frontier testable tree language L over A there exists a finite tree algebra $\mathbf{B}$ satisfying (FT_k) that recognizes L. We shall obtain the algebra $\mathbf{B}$ as the quotient of $\mathbf{F}$ by an appropriate congruence relation.

Let $\approx^k$ be the binary 3-sorted relation on F defined as follows:

$$\approx^k_{\mathrm{A}} = \{(a, a') \in A \times A \mid a = a'\}$$
$$\approx^k_{\mathrm{S}} = \{(s, s') \in S \times S \mid \mathrm{front_k}(s) = \mathrm{front_k}(s')\}$$
$$\approx^k_{\mathrm{T}} = \{(t, t') \in T \times T \mid \mathrm{front_k}(t) = \mathrm{front_k}(t')\}$$

The properties of this relation are stated in the following lemma.

Lemma 3. (1) *The relation $\approx^k$ is a congruence on* $\mathbf{F}$ *of finite index.*

(2) *A tree language over A is k-frontier testable iff it is a union of $\approx^k$-classes.*

(3) *The quotient algebra $\mathbf{F}/\approx^k$ satisfies (FT_k).*

Proof. Clearly $\approx^k$ is an equivalence relation of finite index and saturates every k-testable tree language over A, i.e., every k-testable tree language over A is a union of $\approx^k$-classes. In order to prove that $\approx^k$ is even a congruence relation we have to show that $\approx^k$ is compatible with the operations ι^A, ρ^A, λ^A, κ^A, σ^A, and η^A.

Compatibility with ι: This is immediate.

Compatibility with ρ: Suppose $a \approx^k a'$ and $t \approx^k t'$ for $a, a' \in A$ and $t, t' \in T$. Then $a = a'$ and $\mathrm{front_k}(t) = \mathrm{front_k}(t')$, and we have to show $f_a(x,t) \approx^k f_{a'}(x,t')$. We proceed by case distinction on the depth of t.

Case 1: $\mathrm{depth}(t) < k$. Then $t = t'$ by Remark 4, thus $f_a(x,t) = f_{a'}(x,t')$, hence $f_a(x,t) \approx^k f_{a'}(x,t')$.

Case 2: $\mathrm{depth}(t) \geqslant k$. Then also $\mathrm{depth}(t') \geqslant k$ by Remark 4. Therefore, we may write $\mathrm{front_k}(f_a(x,t)) = \mathrm{front_k}(t) \cup \{x\}$ and also $\mathrm{front_k}(a'(x,t')) = \mathrm{front_k}(t') \cup \{x\}$, thus $f_a(x,t) \approx^k f_{a'}(x,t')$.

Compatibility with λ: Since the definitions of ρ and λ are symmetric to each other and since the definition of $\approx^k$ is symmetric in the same sense, the compatibility of $\approx^k$ with λ follows from its compatibility with ρ.

Compatibility with σ and η: This follows from the fact that for all $s \in S$ and $t \in S \cup T$ the set $\mathrm{front_k}(st)$ only depends on $\mathrm{front_k}(s)$ and $\mathrm{front_k}(t)$: $t' \in \mathrm{front_k}(st)$ iff

- $t' \in \mathrm{front_k}(t)$, or
- t' is an ordinary tree and $t' \in \mathrm{front_k}(s)$, or
- there is a unique tree t'' of maximal depth in $\mathrm{front_k}(t)$ and $\mathrm{depth}(t'') < k$, $t' = s't''$ for some $s' \in \mathrm{front_k}(s) \cap S$, and $\mathrm{depth}(t') \leqslant k$ (see Remark 4).

To prove (3) we have to show: if $\phi = \psi$ is an equation of (FT_k) and if h: $\hat{\mathbf{F}} \to \mathbf{F}/{\approx^k}$ is a homomorphism then $h(\phi) = h(\psi)$ or, equivalently, if h: $\hat{\mathbf{F}} \to \mathbf{F}$ is a homomorphism then $h(\phi) \approx^k h(\psi)$. But the latter follows immediately from Remark 5 and the fact that $\mathrm{front_k}(f_a(t,t')) = \mathrm{front_k}(t) \cup \mathrm{front_k}(t')$ for $t,t' \in T$ with $\mathrm{depth}(t) \geqslant k$ or $\mathrm{depth}(t') \geqslant k$. $\square$

So every k-frontier testable tree language over A is recognized by $\mathbf{F}/{\approx^k}$, and $\mathbf{F}/{\approx^k}$ satisfies (FT_k). This is the correctness of (FT_k).

6.2. Completeness

Let us assume that $h: \mathbf{F} \to \mathbf{B}$ is a homomorphism into a finite tree algebra satisfying (FT_k). We have to show that $\ker(h)$ is at least as large as $\approx^k$. So we assume, furthermore, that t_0 and t_1 are trees over A with $t_0 \approx^k t_1$, and we need to prove $h(t_0) = h(t_1)$.

If $\mathrm{depth}(t_0) < k$ or $\mathrm{depth}(t_1) < k$, then $t_0 = t_1$ by Remark 4, in particular, $h(t_0) = h(t_1)$. If $\mathrm{depth}(t_0) \geqslant k$ and $\mathrm{depth}(t_1) \geqslant k$, the situation is more involved.

We will introduce the notion of a tree in normal form. Each such tree will have the shape of a comb, and the collection of all its teeth will coincide with its set of frontier trees of depth $\leqslant k$. Furthermore, the notion of a tree in normal form will be chosen in such a way that we will be able to prove the following.

(1) For every tree t of depth $\geq k$ there exists a tree t' in normal form such that $t \approx^k t'$ and $h(t) = h(t')$, see Lemma 6.

(2) If t and t' are trees in normal with $t \approx^k t'$, then $h(t) = h(t')$, see Lemma 9.

From this, $h(t_0) = h(t_1)$ follows: by (1), there exist trees t_0' and t_1' such that $t_0 \approx^k t_0'$, $h(t_0) = h(t_0')$, $t_1 \approx^k t_1'$, and $h(t_1) = h(t_1')$; we then have $h(t_1') = h(t_2')$ by (2), thus $h(t_0) = h(t_0') = h(t_1') = h(t_1)$. It remains to give a formal definition of the notion of a tree in normal form and to prove (1) and (2), i.e., Lemmas 6 and 9.

For notational convenience we denote $\ker(h)$ by $\approx^h$ (i.e., $h(t) = h(t')$ iff $t \approx^h t'$) and $\approx^h \cap \approx^k$ by $\approx$.

Let $a \in A$ be a fixed letter. Given trees $u_0, \ldots, u_n$ we denote by $\gamma(u_0, \ldots, u_n)$ the tree

$$f_a(u_0, f_a(u_1, \ldots, f_a(u_{n-1}, u_n) \ldots)). \tag{27}$$

By convention, when $n = 0$, then $\gamma(u_0)$ is set to u_0.

Definition 4. A tree t is in *normal form* if $t = \gamma(u_0, \ldots, u_n)$ for some $n > 0$ and some trees $u_0, \ldots, u_n$ such that $\operatorname{depth}(u_n) = k$, $\operatorname{depth}(u_i) \leqslant k$ for i with $i \leqslant n$, and $\operatorname{front}_k(t) = \{u_0, \ldots, u_n\}$.

The last condition is equivalent to $\operatorname{front}_k(u_i) \subseteq \{u_0, \ldots, u_n\}$ for every i with $i \leqslant n$.

If we say that '$t = \gamma(u_0, \ldots, u_n)$ is a tree in normal form' we assume that the u_i satisfy the conditions of Definition 4. (Observe that, in general, there is no unique way to write a tree as in (27).)

In what follows the reader has to keep in mind that, by Lemma 3, every transformation according to an equation of (FT_k) does not only preserve $\approx^h$-equivalence but also ensures $\approx^k$-equivalence, hence $\approx$-equivalence.

We start with two technical lemmas.

Lemma 4. *Let t be a tree with depth $\geqslant k$ and let $u_0, \ldots, u_{n-1}$ be an enumeration of the elements of* $\operatorname{front}_k(t)$. *Then $\gamma(u_0, \ldots, u_{n-1}, t) \approx t$.*

Proof. For $i = 0, \ldots, n$ we prove $\gamma(u_0, \ldots, u_{i-1}, t) \approx t$ by induction. The induction base $(i = 0)$ is trivial. For the induction step let $i < n$ and suppose $\gamma(u_0, \ldots, u_{i-1}, t) \approx t$. Since $u_i \in \operatorname{front}_k(t)$ we can write $t = su_i$ for some special tree s. Then $t \approx \gamma(u_0, \ldots, u_i, t)$ is proven by the following chain of transformations:

$$\begin{aligned}
t &\approx \gamma(u_0, \ldots, u_{i-1}, t) && \text{(induction hypothesis)}\\
&\approx \gamma(u_0, \ldots, u_{i-1}, f_a(su_i, t)) && \text{(because of } (\mathrm{Idp}_k))\\
&\approx \gamma(u_0, \ldots, u_{i-1}, f_a(u_i, f_a(su_i, t))) && \text{(because of } (\mathrm{Can}_k) \text{ or } (\mathrm{Can}'_k))\\
&\approx \gamma(u_0, \ldots, u_{i-1}, u_i, f_a(su_i, t)) && \text{(definition of } \gamma)\\
&\approx \gamma(u_0, \ldots, u_i, t) && \text{(because of } (\mathrm{Idp}_k)) \quad \square
\end{aligned}$$

Lemma 5. *Let $t = \gamma(u_0, \ldots, u_m)$ be a tree with* $\operatorname{depth}(u_m) \geqslant k$ *and (t') an arbitrary tree. Then $f_a(t, t') \approx \gamma(u_0, \ldots, u_m, t')$.*

Proof. We prove this by induction on m. The case $m = 0$ is trivial. If $m > 0$, observe the following:

$$\begin{aligned}
f_a(t, t') &= f_a(\gamma(u_0, \ldots, u_m), t') && \text{(assumption about } t)\\
&= f_a(f_a(u_0, \gamma(u_1, \ldots, u_m)), t') && \text{(definition of } \gamma)\\
&\approx f_a(u_0, f_a(\gamma(u_1, \ldots, u_m), t')) && \text{(because of } (\mathrm{Rot}_k))\\
&\approx f_a(u_0, \gamma(u_1, \ldots, u_m, t')) && \text{(induction hypothesis)}\\
&= \gamma(u_0, u_1, \ldots, u_m, t') && \text{(definition of } \gamma) \quad \square
\end{aligned}$$

We can now prove that every tree of depth $\geqslant k$ is $\approx$-equivalent to a tree in normal form.

Lemma 6. *For every tree t of depth $\geqslant k$, there exists a tree t' in normal form such that $t \approx t'$.*

Proof. We prove this by induction on depth(t).

Induction base: depth(t) $= k$. If $\text{front}_k(t) = \{u_0, \ldots, u_{n-1}\}$ then, by Lemma 4, $t \approx \gamma(u_0, \ldots, u_{n-1}, t)$ and $\gamma(u_0, \ldots, u_{n-1}, t)$ is in normal form.

Induction step: depth(t) $> k$. Let $t = f_b(t_0, t_1)$. We proceed by case distinction on the depth of t_1.

Case 1: depth$(t_1) \geqslant k$. By induction hypothesis there exists $t'' = \gamma(v_0, \ldots, v_m)$ in normal form with $t'' \approx t_1$. A case distinction on the depth of t_0 is helpful.

Case 1a: depth$(t_0) \leqslant k$. Assume $\text{front}_k(f_b(t_0, t'')) = \{u_0, \ldots, u_{n-1}\}$. We have

$$\begin{aligned} t &\approx f_b(t_0, t'') \\ &\approx \gamma(u_0, \ldots, u_{n-1}, f_b(t_0, t'')) && \text{(Lemma 4)} \\ &\approx \gamma(u_0, \ldots, u_{n-1}, f_a(t_0, t'')) && \text{(because of (Sym))} \\ &\approx \gamma(u_0, \ldots, u_{n-1}, t_0, t'') && \text{(definition of } \gamma) \\ &\approx \gamma(u_0, \ldots, u_{n-1}, t_0, v_0, \ldots, v_m) && \text{(definition of } \gamma) \end{aligned}$$

and the last tree is in normal form.

Case 1b: depth$(t_0) > k$. By induction hypothesis there exists a tree $t' = \gamma(u_0, \ldots, u_m)$ in normal form with $t' \approx t_0$. By Lemma 5 we know $f_b(t', t'') \approx \gamma(u_0, \ldots, u_m, v_0, \ldots, v_n)$. Since the tree on the right-hand side is in normal form, this is sufficient.

Case 2: depth$(t_1) < k$. Then depth$(t_0) \geqslant k$, and an application of (Sym$_k$) yields $t \approx f_b(t_1, t_0)$. Case 1 applies to the tree of the right-hand side, so we find t' in normal form with $t' \approx f_b(t_1, t_0)$, hence $t' \approx t$. □

We now show that adjacent teeth commute.

Lemma 7. *Let $t = \gamma(u_0, \ldots, u_n)$ be a tree in normal form. If*

(1) $i < n - 1$, *or*

(2) $i = n - 1$ *and* depth$(u_{n-1}) = k$,

then $t \approx \gamma(u_0, \ldots, u_{i-1}, u_{i+1}, u_i, u_{i+2}, \ldots, u_n)$, and the tree on the right-hand side is also in normal form.

Proof. The first case is an immediate consequence of (Sym$_k$). The second case is proven as follows.

$$\begin{aligned} t &= \gamma(u_1, \ldots, u_{i-1}, f_a(u_i, f_a(u_{i+1}, \gamma(u_{i+2}, \ldots, u_n)))) && \text{(definition of } \gamma) \\ &\approx \gamma(u_1, \ldots, u_{i-1}, f_a(f_a(u_{i+1}, \gamma(u_{i+2}, \ldots, u_n)), u_i)) && \text{(because of (Sym}_k)) \\ &\approx \gamma(u_1, \ldots, u_{i-1}, f_a(u_{i+1}, f_a(\gamma(u_{i+2}, \ldots, u_n), u_i))) && \text{(because of (Rot}_k)) \\ &\approx \gamma(u_1, \ldots, u_{i-1}, f_a(u_{i+1}, f_a(u_i, \gamma(u_{i+2}, \ldots, u_n)))) && \text{(because of (Sym}_k)) \\ &= s\gamma(u_1, \ldots, u_{i-1}, u_{i+1}, u_i, u_{i+2}, \ldots, u_n) && \text{(definition of } \gamma) \quad \square \end{aligned}$$

Next, we observe that multiple occurrences of a teeth can be eliminated.

Lemma 8. *If $t = \gamma(u_0, \ldots, u_n)$ is a tree in normal form and if $0 \leqslant i < n$ and $u_i = u_{i+1}$, then $t \approx \gamma(u_0, \ldots, u_i, u_{i+2}, \ldots, u_n)$, and the tree on the right-hand side is also in normal form.*

Proof. For $i = n - 1$, the claim follows immediately from (Idp_k). The case $i < n - 1$ is proven by the following transformations:

$$\begin{aligned} t &= \gamma(u_1, \ldots, u_{i-1}, f_a(u_i, f_a(u_i, \gamma(u_{i+2}, \ldots, u_n)))) && \text{(definition of } \gamma,\ u_i = u_{i+1}) \\ &\approx \gamma(u_1, \ldots, u_{i-1}, f_a(u_i, \gamma(u_{i+2}, \ldots, u_n))) && \text{(because of } (\mathrm{Can}'_k)) \\ &= \gamma(u_1, \ldots, u_i, u_{i+2}, \ldots, u_n) && \text{(definition of } \gamma) \quad \square \end{aligned}$$

Combining the foregoing two lemmas we can conclude the following.

Lemma 9. *Two $\approx^k$-equivalent trees in normal form are also $\approx^h$-equivalent.*

Proof. Let $t = \gamma(u_0, \ldots, u_m)$ and $t' = \gamma(v_0, \ldots, v_n)$ be $\approx^k$-equivalent trees in normal form. Then $\{u_0, \ldots, u_m\} = \{v_0, \ldots, v_n\}$ by definition. By Lemma 7, the u_i and the v_j can be rearranged such that their order in both trees is the same. In addition, Lemma 8 allows to reduce the multiplicities of every u_i or v_j, respectively, to one. Thus both trees can be transformed into the same tree, hence $t \approx t'$. □

In view of what was said at the beginning of this subsection, the proof of the last lemma also completes the proof of the completeness of (FT_k), which, in turn, completes the proof of Theorem 1.

7. An efficient algorithm that decides frontier testability

On the basis of Theorem 1 we will develop an efficient decision procedure that decides whether a regular tree language represented by a finite tree automaton is frontier testable or not.

Corresponding to Remark 5 we have the following.

Remark 6. Let $\mathbf{D}$ be a finite tree algebra generated by D_A and assume that the sets E_i with $i \geqslant 1$ are defined as follows:

$$E_1 = D_\mathrm{T}$$

$$E_{i+1} = \{\kappa^{\mathbf{D}}(a, q, q') \mid a \in D_\mathrm{A} \wedge (q, q') \in (D_\mathrm{T} \times E_i) \cup (E_i \times D_\mathrm{T})\}$$

Then $q \in E_i$ if and only if there exists a homomorphism $h: \hat{\mathbf{F}} \to \mathbf{D}$ such that $h(\tau_i) = q$.

The sets E_i therefore build a descending chain. Since none of the E_i is empty (provided $D_{\mathrm{A}} = \emptyset$), the chain is stationary from the $|D_{\mathrm{T}}|$-th entry onwards:

Remark 7. Let $\mathbf{D}$ and the sets E_i be as in the previous remark. Then $E_1 \supseteq E_2 \supseteq \cdots \supseteq E_l = E_{l+1} = \cdots$, where $l = |D_{\mathrm{T}}|$.

This remark is helpful to prove the following lemma, in the proof of which, for every $q \in D_{\mathrm{T}}$, the set E^q is defined to be the supremum of the chain $E_0^q \subseteq E_1^q \subseteq \cdots$ with E_i^q inductively defined as follows:

$$E_0^q = \{q\}$$
$$E_{i+1}^q = E_i^q \cup \{\kappa^{\mathbf{D}}(a,q,q') \mid a \in D_{\mathrm{A}} \wedge (q,q') \in (D_{\mathrm{T}} \times E_i^q) \cup (E_i^q \times D_{\mathrm{T}})\}$$

Lemma 10. *Let* $\mathbf{D}$ *and* l *be as in the previous lemma. If* $\mathbf{D}$ *satisfies* (FT_k) *for some* k*, then* $\mathbf{D}$ *satisfies* (FT_l).

Proof. By Remark 6, $\mathbf{D}$ satisfies (FT_k) if and only if it satisfies (28)–(31) below.

$$a(d,q) = b(q,d) \quad \text{for } a,b \in D_{\mathrm{A}},\ q \in D_{\mathrm{T}},\ d \in E_k \tag{28}$$
$$a(d,d) = d \quad \text{for } a \in D_{\mathrm{A}},\ d \in E_k \tag{29}$$
$$a(q,a(d',d)) = a(d',d) \quad \text{for } a \in D_{\mathrm{A}},\ q \in D_{\mathrm{T}},\ d \in E_k,\ d' \in E^q \tag{30}$$
$$a(a(q,d),q') = a(q,a(d,q')) \quad \text{for } a \in D_{\mathrm{A}},\ q,q' \in D_{\mathrm{T}},\ d \in E_k \tag{31}$$

Therefore, $\mathbf{D}$ satisfies (FT_l) if it satisfies (FT_k) for some k, since, by Remark 7, the set E_l is the smallest among all E_i. □

We obtain the following interesting consequence of Theorem 1, which corresponds to a result in [8].

Corollary 1. *Let* L *be a regular tree language over the alphabet* A*. Then the following conditions are equivalent*:

(A) L *is frontier testable.*

(B) L *is* l*-frontier testable, where* $l = |(F/{\sim^L})_{\mathrm{T}}|$.

From Lemma 10 and its proof we see that, in order to check whether a regular tree language L over A is frontier testable, it is sufficient to check whether an isomorphic copy $\mathbf{D}$ of $\mathbf{F}/{\sim^L}$ satisfies (28)–(31) with $k = |D_{\mathrm{T}}|$.

In general, computing an isomorphic copy $\mathbf{D}$ of the syntactic tree algebra of a regular tree language L starting from a tree automaton is expensive: the cardinality of D_{S} has an exponential lower bound in terms of the states of a minimal automaton for L. However, in the definition of the sets E_i and E^q and in (28)–(31) we do not need D_{S}; we only need D_{T} and $\kappa^{\mathbf{D}}$. These two objects can be computed efficiently, as we will see soon.

Let $(\mathbf{B},R)$ be a tree automaton over A and L the language recognized by it. Consider the finite tree algebra $\mathbf{C}$ (with $C=(A,B^B,B)$) as constructed between Remarks 2 and 3, which recognizes L via the homomorphism $h:\mathbf{F}\to\mathbf{C}$ with $h(a)=a$ for every $a\in A$. Let $\mathbf{C}'$ be the substructure of $\mathbf{C}$ generated by the 3-sorted set $(A,\emptyset,\emptyset)$. (Note that $C'_\mathrm{A}=A$ and $C'_\mathrm{T}=B$, since $\mathbf{B}$ is assumed to be generated by the empty set.) According to (24)–(26) we define a binary 3-sorted relation $\sim$ on C' as follows:

$$\begin{aligned}\sim_\mathrm{A} = \{(a,a')\in A\times A \mid (&c_a^\mathbf{B}\in R\leftrightarrow c_{a'}^\mathbf{B}\in R)\\ &\wedge\forall s\in C'_\mathrm{S}\,\forall q,q'\in B\,(sf_a^\mathbf{B}(q,q')\in R\leftrightarrow sf_{a'}^\mathbf{B}(q,q')\in R)\\ &\wedge\forall q,q'\in B\,(f_a^\mathbf{B}(q,q')\in R\leftrightarrow f_{a'}^\mathbf{B}(q,q')\in R)\},\end{aligned}\tag{32}$$

$$\begin{aligned}\sim_\mathrm{S} = \{(f,f')\in C'_\mathrm{S}\times C'_\mathrm{S} \mid &\forall q\in B\,(f(q)\in R\leftrightarrow f'(q))\\ &\wedge\forall q\in B\,\forall g\in C'_\mathrm{S}\,(g(f(q))\in R\leftrightarrow g(f'(q))\in R)\},\end{aligned}\tag{33}$$

$$\begin{aligned}\sim_\mathrm{T} = \{(q,q')\in B\times B \mid &(q\in R\leftrightarrow q'\in R)\\ &\wedge\forall f\in C'_\mathrm{S}\,(f(q)\in R\leftrightarrow f'(q)\in R)\}.\end{aligned}\tag{34}$$

It is clear that $\sim$ is a congruence on $\mathbf{C}'$ and that $\mathbf{D}$ defined by $\mathbf{D}=\mathbf{C}'/{\sim}$ is an isomorphic copy of $\mathbf{F}/{\sim^L}$. We have

$$\kappa^\mathbf{D}(a/{\sim},q/{\sim},q'/{\sim}) = f_a(q,q')/{\sim}\tag{35}$$

for $a\in A$ and $q,q'\in B$.

Another way to define $\sim_\mathrm{T}$ is to say that it is the coarsest equivalence relation on B that is compatible with the partition $\{R,B\setminus R\}$ and the functions $g_a^q:B\to B$ and $h_a^q:B\to B$ defined by $g_a^q(q')=f_a^\mathbf{B}(q,q')$ and $h_a^q(q')=f_a^\mathbf{B}(q',q)$ for $a\in A$ and $q\in B$. By a result of [1], $\sim_\mathrm{T}$ can therefore be computed in time $\mathrm{O}(mn\log n)$ where $m=|A|$ and $n=|B|$.

Taking all together we can now state:

Theorem 2. *There exists an algorithm that, given a tree automaton with n states and over an alphabet of cardinality m, decides in time* $\mathrm{O}(mn^3+m^2n^2)$ *whether the tree language recognized by the automaton is frontier testable.*

Proof. A sketch of an algorithm follows. Its correctness is clear from the proof of Lemma 10 and Theorem 1.

1. Compute the binary relation $\sim_\mathrm{T}$ on $C'_\mathrm{T}=B$ using the algorithm presented in [1]. This takes time $\mathrm{O}(mn\log n)$.

2. Let $D_\mathrm{T}=C'_\mathrm{T}/{\sim}$ and $l=|D_\mathrm{T}|$. (Note: $l\leqslant n$.)

3. Compute $E_1,\ldots,E_l$ as defined in Remark 6, using (35). This can be done in a straightforward way according to the definitions in time $\mathrm{O}(mn^2)$.

4. For every $q\in D_\mathrm{T}$, compute the set E^q. This can be done in a straightforward way in time $\mathrm{O}(mn^3)$.

5. Check (28)–(31) with $k = l$, again using (35). This takes time $O(m^2n^2 + mn + mn^3 + mn^3)$.

6. Report whether the equations are satisfied or not.

The overall running time is $O(mn^3 + m^2n^2)$. □

8. An equational characterization of frontier testability

The aim of this section is to present a finite set of equations characterizing frontier testable tree languages (where the parameter k is not fixed). This is not achievable, if we stick to the present signature Σ. The following proposition even states that no set (no matter whether this set is finite or infinite) of equations in Σ characterizes the class of the syntactic tree algebras of all frontier testable tree languages. However, as we will see below, extending Σ helps.

Proposition 4. *There is no set E of equations in the signature Σ such that the following holds for every regular tree language L.*

(+) L is frontier testable if and only if the syntactic tree algebra of L satisfies E.

Proof. For contradiction let us assume that E is a set of equations satisfying (+). We distinguish two cases.

Case 1: Every equation of E is true in $\mathbf{F}$. Then every quotient of $\mathbf{F}$ satisfies E, in particular, the syntactic tree algebra of a regular non-frontier testable tree language.

Case 2: There is an equation $\phi = \psi$ in E such that there exists a homomorphism $h\colon \hat{\mathbf{F}} \to \mathbf{F}$ with $h(\phi) \neq h(\psi)$. W.l.o.g. we assume that ϕ and ψ are of sort T. (If ϕ and ψ are of sort A, consider $\eta(\phi) = \eta(\psi)$. If ϕ and ψ are of sort S, consider $\eta(\phi, t) = \eta(\psi, t)$ for a new variable t of sort T.) Let $t_0 = h(\phi)$, $t_1 = h(\psi)$, and $k = \max\{\mathrm{depth}(t_0), \mathrm{depth}(t_1)\}$. W.l.o.g. assume $t_1 \notin \mathrm{front}(t_0)$ and set $L = \{t \in T \mid t_0 \in \mathrm{front}_k(t) \wedge t_1 \notin \mathrm{front}_k(t)\}$. Then L is a k-frontier testable language, but its syntactic tree algebra does not satisfy the equation $\phi = \psi$ (since $t_0 \in L$, but $t_1 \notin L$), hence it does not satisfy E – a contradiction. □

To be able to characterize frontier testability by equations, we could adapt the notion of 'ultimately defined by an infinite sequence of equations' (e.g., see [11]) known from finite semigroup theory. But introducing implicit operations (see [14]) is an even better remedy, for in our case the set of equations characterizing frontier testability will turn out to be finite. We do not need the entire machinery of implicit operations and implicit equations (as elaborated in [2]), just the implicit ω-operation, which is known from finite semigroup theory.

We recall some facts and notation from finite semigroup theory (see, e.g., [3]). An element s of a finite semigroup is called idempotent if $s^2 = s$. In every finite semigroup there exists, for every element s, a unique idempotent element in the subsemigroup generated by s, i.e., in the set $\{s,s^2,s^3,\ldots\}$. This element is denoted by s^ω.

As pointed out in Remark 1, if $\mathbf{B}$ is a tree algebra, then B_S together with $\sigma^\mathbf{B}$ forms a semigroup. In view of what has just been said about finite semigroups, this motivates the following definition in which Σ' denotes the 3-sorted signature $\Sigma \cup \{{}^\omega{}_{(S,S)}\}$.

Definition 5. If $\mathbf{B}$ is a finite tree algebra, then $\overline{\mathbf{B}}$ is the 3-sorted Σ'-expansion of $\mathbf{B}$ in which $s^{\omega^{\overline{\mathbf{B}}}}$ is the unique idempotent element in $\{s,s^2,s^3,\ldots\}$, where B_S together with $\sigma^\mathbf{B}$ is viewed as a semigroup.

We use the following lemma known from finite semigroup theory (see, e.g., [11]).

Lemma 11. *If S is a finite semigroup of cardinality n and if $m > n$, then the following assertions are equivalent for an element $s \in S$:*

(A) *There exist $s_0,\ldots,s_{m-1} \in S$ such that $s = s_0 s_1 \cdots s_{m-1}$.*

(B) *There exist $s_0,s_1,s_2 \in S$ such that $s = s_0 s_1^\omega s_2$.*

In our situation this extends to the following remark, where $\hat{\mathbf{F}}'$ denotes the free 3-sorted Σ'-term algebra in the variables s_0, s_1, and t_0.

Remark 8. Let $\mathbf{D}$, the sets E_i, and l be as in Remarks 6 and 7. Then $q \in E_l$ iff there exists a homomorphism h: $\hat{\mathbf{F}}' \to \overline{\mathbf{D}}$ such that $h(s_0 s_1^\omega t_0) = q$.

From this, in view of Theorem 1 and the proof of Lemma 10, we get the desired result:

Theorem 3. *Let L be a regular tree language over A. Then the following conditions are equivalent.*

(A) *L is frontier testable.*

(B) *The 3-sorted Σ'-algebra $\overline{\mathbf{F}/{\sim^L}}$ satisfies the following equations, which are denoted by (FT).*

$$a(s_0 s_1^\omega t_0, t) = b(t, s_0 s_1^\omega t_0) \qquad \text{(Sym)}$$

$$a(s_0 s_1^\omega t_0, s_0 s_1^\omega t_0) = s_0 s_1^\omega t_0 \qquad \text{(Idp)}$$

$$a(t, a(st, s_0 s_1^\omega t_0)) = a(st, s_0 s_1^\omega t_0) \qquad \text{(Can)}$$

$$a(a(t, s_0 s_1^\omega t_0), t') = a(t, a(s_0 s_1^\omega t_0, t')) \qquad \text{(Rot)}$$

(C) *L is recognized by a finite tree algebra $\mathbf{B}$ such that $\overline{\mathbf{B}}$ satisfies (FT).*

9. Concluding remarks

In [10, 12] Nivat, Péladeau, and Podelski use semigroups to classify regular tree languages; they characterize classes of regular tree languages by the corresponding classes of syntactic semigroups. This does not work for frontier testability. If A is a two letter alphabet and a denotes one of its elements, the syntactic semigroup of the tree language $L = \{t \in T \mid a \in \text{front}_1(t)\}$ is an extension of the syntactic semigroup of the language

$$L' = L \cup \{t \in T \mid \exists\, s \in S' \,\exists\, t_0, t_1 \in T\, (t = sf_a(t_0, t_1))\},$$

and L is 1-frontier testable but L' is not. So neither the class of all frontier testable nor the class of k-frontier testable tree languages (for any $k > 0$) can be characterized by a set of identities or pseudoidentities for semigroups (since validity of equations is preserved by passing to substructures).

In [16] Steinby develops an algebraic framework for the classification of regular tree languages. From (FT$_k$) one can easily derive a finite set of equations characterizing k-frontier testability in that framework. However, it is not clear to me whether the class of all frontier testable tree languages can be characterized in Steinby's framework by a *finite* set of equations (even when implicit operations are allowed). From (FT) I can derive only an infinite set of equations using an infinite number of implicit operations.

It is not hard to extend the equational characterization of frontier testability of Section 8 to the more general case of generalized definite tree languages [9, 15]; one only needs to combine our results with the results on definite tree languages presented in [10]. It is, however, unlikely that the property of being generalized definite (not even of being definite) can be decided efficiently.

Presumably, a transformation of known results [2] from universal algebra could provide an abstract framework for the classification of regular tree languages using tree algebras, i.e., a correspondence between classes of regular tree languages defined by certain closure properties and 'pseudovarieties of A-generated (see Remark 6) tree algebras'. Steinby's framework mentioned above provides a similar correspondence for so-called "varieties of Σ-tree languages" and "varieties of finite Σ-algebras" for arbitrary one-sorted signatures Σ. However, an abstract classification result for tree algebras would have to prove to be useful in the sense that it allows succinct characterizations of interesting classes of regular tree languages. Therefore, a first aim should be a characterization of other, more complicated classes of regular tree languages by using tree algebras. Perhaps, a characterization of locally testable tree languages [8, 16], a natural class of regular tree languages, can be obtained along these lines.

Acknowledgements

I thank Magnus Steinby, Andreas Podelski, and the anonymous referee for valuable comments on and corrections of this paper and also its conference version [18].

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[2] J. Almeida, On pseudovarieties, varieties of languages, filters of congruences, pseudoidentities and related topics, *Algebra Universalis* **27** (1990) 333–350.

[3] J. Almeida, *Finite Semigroups and Universal Algebra* (World Scientific, Singapore, 1995).

[4] S. Burris and H.P. Sankappanavar, *A Course in Universal Algebra* (Springer, New York, 1981).

[5] B. Courcelle, D. Niwinski and A. Podelski, A geometrical view of the determinization and minimization of finite-state automata, *Math. Systems Theory* **24** (1991) 117–146.

[6] S. Eilenberg, *Automata, Languages and Machines*, Vol. B (Academic Press, New York, 1976).

[7] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification 1* (Springer, Berlin, 1985).

[8] U. Heuter, Zur Klassifizierung regulärer Baumsprachen, Dissertation, RWTH Aachen, 1989.

[9] U. Heuter, Generalized definite tree languages, in: *Proc. MFCS '89*, Lecture Notes in Computer Science, Vol. 379 (Springer, Berlin, 1985) 270–280.

[10] M. Nivat and A. Podelski, Definite tree languages, *Bull. European Assoc. Theoret. Comput. Sci.* **38** (1989) 186–190.

[11] J.-E. Pin, *Varieties of Formal Languages* (North Oxford Academic Press, London, 1986).

[12] P. Péladeau and A. Podelski, On reverse and general definite tree languages, in: *Proc. ICALP '92*, Lecture Notes in Computer Science, Vol. 623 (Springer, Berlin, 1992) 150–161.

[13] A. Podelski, Monoïdes d'arbres et automates d'arbres, Thèse, Université Paris 7, 1989.

[14] J. Reiterman, The Birkhoff Theorem for finite algebras, *Algebra Universalis* **14** (1982) 1–10.

[15] T. Scholz, Charakterisierung definiter Baumsprachen durch Gleichungen in der Termalgebra, Diplomarbeit, Inst. f. Inform. u. Prakt. Math., Universität Kiel, 1992.

[16] M. Steinby, A theory of tree language varieties, in: Maurice Nivat and Andreas Podelski, ed., *Tree Automata and Languages* (Elsevier, Amsterdam, 1992) 57–81.

[17] W. Thomas, Logical aspects in the study of tree languages, in: *Proc. CAAP'84* (Cambridge Univ. Press, Cambridge, 1984) 31–51.

[18] Th. Wilke, Algebras for classifying regular tree languages and an application to frontier testability, in: *Proc. ICALP'93*, Lecture Notes in Computer Science, Vol. 700 (Springer, Berlin, 1993) 347–358.