

Learning k-Testable tree sets from positive data

Pedro García

DSIC-II/46/93

Learning k -Testable tree sets from positive data*

Pedro García

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

e-mail: pgarcia@dsic.upv.es

Abstract

A k -Testable tree set in the Strict sense (k -TS) is essentially defined by a finite set of patterns of "size" k that are permitted to appear in the trees of the tree language. Given a positive sample S of trees over a ranked alphabet, an algorithm is proposed which obtains the smallest k -TS tree set containing S . The proposed algorithm is polynomial on the size of S and identifies the class of k -TS tree languages in the limit from positive data.

I. INTRODUCTION

Most properties of regular languages and finite automata have been extended to regular tree set theory [7], [3]. This allows for extending many of the results from regular language learning to tree sets [1], [4-6].

Local tree sets have been introduced in [7] as sets of derivation trees of generalized context-free grammars. On the other hand, in [2], an algorithm is proposed that identifies the family of k -Testable in Strict sense languages (k -TS) in the limit. In this paper, we generalize the Thatcher definition and define the families of k -TS sets in such a way that for $k=2$ the families are reduced to that of the local sets mentioned above. Intuitively, a tree set is k -TS if its elements can be characterized by the presence of some "patterns" of "size" k in them.

We also propose an algorithm that identifies the family of k -TS sets in the limit from positive data. The proposed algorithm, when operating with string languages, becomes the algorithm shown in [2].

* Work partially supported by the Spanish CICYT under grant TIC-0448/89

In the Pattern Recognition framework, the representation of complex objects using trees allows us to reflect the structural information in a direct way in the case of scene recognition, or in an abstract way, by expressing hierarchical relations among the primitive elements. The learning of k -TS tree sets is not only interesting from the point of view of Inductive Inference Theory. The potential applications of this family of sets to image recognition and to language modeling for speech recognition are what make this learning very promising.

II.-PRELIMINARIES AND NOTATION

Let \mathbb{N} be the set of natural numbers and \mathbb{N}^* the free monoid generated by \mathbb{N} with $.$ as the operation and λ as the identity. We define $u \leq w$ for $u, w \in \mathbb{N}^*$ iff there exists $v \in \mathbb{N}^*$ such that $w = u.v$ ($u < w$ if $u \leq w$ and $u \neq w$). For $x \in \mathbb{N}^*$, we define the *length* of x denoted by $|x|$ as follows:

$$|\lambda| = 0, |x.n| = |x| + 1 \quad \text{for } n \in \mathbb{N}$$

$D \subseteq \mathbb{N}^*$ is a *tree domain* iff it satisfies: a) $v \in D$ and $u < v$ implies $u \in D$ b) if $u.i \in D$, $i \in \mathbb{N}$, then $u.j \in D$ for $1 \leq j \leq i$.

A *ranked alphabet* V is a finite set associated with a finite relation $r \subseteq (V \times \mathbb{N})$. V_n denotes the subset of V : $\{\sigma \in V \mid (\sigma, n) \in r\}$.

A tree t over a ranked alphabet V is a mapping $t : D \rightarrow V$ with D being a tree domain called domain of t and denoted by $dom(t)$. The set of finite trees over V will be called V^T . The alphabet can be seen as a set of function symbols having different arities in the way that V^T can be considered as the set of terms over V . For example, the tree shown in Figure 1 (left) can be represented as $S(A(a,b), B(c, C(c)))$.

Let $t \in V^T$ and $x \in dom(t)$. The *depth* of x is defined as $depth(x) = |x|$ and the depth of t as $depth(t) = \max\{depth(x) \mid x \in dom(t)\}$. The *subtree* of t rooted at x , denoted as t/x is defined as: $dom(t/x) = \{y \mid x.y \in dom(t)\}$ and $(t/x)(y) = t(x.y) \quad \forall y \in dom(t/x)$. If $t \in V^T$, then $ST(t)$ is the set of subtrees of t , that is, $ST(t) = \{t/x \mid x \in dom(t)\}$ and for the set $T \subseteq V^T$, $ST(T) = \approx_{t \in T} ST(t)$. The *replacement* of the subtree t/x with $s \in V^T$ is defined as:

$$t(x \leftarrow s)(y) = \begin{cases} t(y) & \text{if } |y| \leq |x|; y \neq x; y \in dom(t) \\ s(z) & \text{if } y = x.z; z \in dom(s) \end{cases}$$

A *skeletal alphabet* is a ranked alphabet with exactly one symbol whose arities are greater or equal to one. If Sk denotes such an alphabet and V_0 is an alphabet of symbols whose arity is zero, a tree over $Sk \cup V_0$ is called a *skeleton* (all its inner nodes are labeled by σ and the leaves are symbols from V_0). Given (V, r) and (V', r') , a *transcription* is a mapping $\pi : V \rightarrow V'$ with $\pi(V_n) \subseteq V'_n$. The function π can be extended to $\pi : V^T \rightarrow V'^T$ making $\pi(\sigma(t_1, \dots, t_n)) = \pi(\sigma)(\pi(t_1), \dots, \pi(t_n))$. Let $V' = (Sk \cup V_0)$ and for any $a \in V_0$, $\pi(a) = a$ and $\pi(A) = \sigma$ for $A \in V_n$, $n \geq 1$; in this case if $t \in V^T$, $\pi(t)$ will give the skeleton of t which will be denoted as $sk(t)$. For the set $T \subseteq V^T$, the set of skeletons associated to the trees in T is $sk(T) = \cup_{t \in T} sk(t)$.

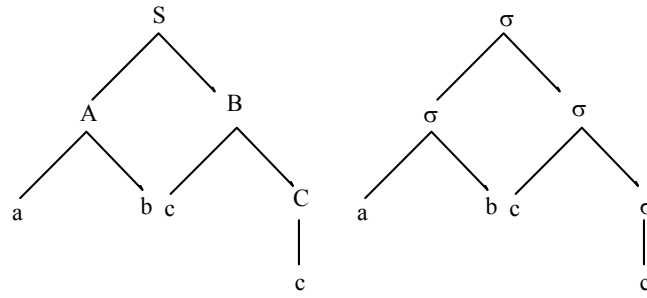


Fig.1. Example of a derivation tree and its associated skeleton

Let V be a ranked alphabet and m the greatest arity of the symbols in V . A *deterministic tree automaton (DTA)* is defined as the four-tuple $A = (Q, V, \delta, F)$ where Q is a finite set (states) containing V_0 , $F \subseteq Q$ is the set of final states and $\delta = (\delta_0, \delta_1, \dots, \delta_m)$ the set of state transition functions defined by:

$$\delta_n: (V_n \times Q^n) \rightarrow Q, \quad n=1,2,\dots,m$$

$$\delta_0(a) = a, \quad \forall a \in V_0$$

δ can be extended to operate on trees as follows:

$$\delta(\sigma(t_1, \dots, t_n)) = \begin{cases} \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) & \text{if } n > 0, \\ \delta_0(\sigma) & \text{if } n = 0 \end{cases}$$

A tree $t \in V^T$ is accepted by A if $\delta(t) \in F$. The set of trees accepted by A is defined as $T(A) = \{t \in V^T \mid \delta(t) \in F\}$

An equivalence relation \equiv defined over V^T is *subtree invariant* if $t_1 \equiv t_2$ implies $t(x \leftarrow t_1) \equiv t(x \leftarrow t_2)$ for each $t \in V^T$ and $x \in \text{dom}(t)$. A tree set is regular iff it is the union of some of the equivalence classes of a subtree-invariant equivalence relation of finite index.

For a context-free grammar $G = (N, \Sigma, P, S)$ and for each symbol $X \in N \cup \Sigma$ we define the set of trees from G rooted at X as:

$$D_X(G) = \begin{cases} \{a\} & \text{if } X = a \in \Sigma \\ \{X(t_1, \dots, t_n) \mid X \rightarrow X_1 \dots X_n \in P; t_i \in D_{X_i}(G); 1 \leq i \leq n\} & \text{if } X \in N \end{cases}$$

$D_S(G)$ denotes the set of *derivation trees* of G . If $Sk = \{\sigma\}$, the set of trees over $Sk \cup \Sigma$ which are skeletons associated to trees in $D_S(G)$ is denoted by $sk(D(G))$ [5]. Both $D_S(G)$ and $sk(D(G))$ are regular tree sets.

Let $A = (Q, Sk \cup \Sigma, \delta, F)$ be a DTA for a set of skeletons. There exists a context-free grammar $G = (N, \Sigma, P, S)$ such that $sk(D(G)) = T(A)$ which can be obtained as follows:

$$N = (Q - \Sigma) \cup \{S\}; P = \{\delta_n(\sigma, q_1, \dots, q_n) \rightarrow q_1 \dots q_n \mid n \geq 1, \sigma \in Sk_n, q_1, \dots, q_n \in Q\} \cup \{S \rightarrow q_1 \dots q_n \mid n \geq 1, \delta_n(\sigma, q_1, \dots, q_n) \in F\}$$

III. K-TESTABLE TREE SETS IN THE STRICT SENSE

Let (V, r) be a ranked alphabet, $k \geq 2$ and let V^T be the set of finite trees over V . For every $t \in V^T$, the k -test vector of t is defined as: $Test_k(t) = (\mathbf{r}_{k-1}(t), \mathbf{l}_{k-1}(t), \mathbf{p}_k(t))$ where:

$$\mathbf{r}_{k-1}(t) = \begin{cases} t & \text{if } depth(t) \leq k-2 \\ t' \mid depth(t') = k-2; dom(t') = dom_{k-2}(t); t'(y) = t(y) \forall y' \in dom(t) & \text{if } depth(t) > k-2 \end{cases}$$

where $dom_k(t) = \{x \in dom(t) \mid |x| \leq k\}$

$$\mathbf{p}_k(t) = \begin{cases} \emptyset & \text{if } depth(t) < k-1 \\ \{r_k(t') \mid t' \in ST(t); depth(t') \geq k-1\} & \text{if } depth(t) \geq k-1 \end{cases}$$

$$\mathbf{l}_{k-1}(t) = \{t' \in ST(t) \mid depth(t') \leq k-2\}$$

Example 1. Let $t = \sigma(\sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c)))$. Then:

$$\mathbf{r}_2(t) = \sigma(\sigma, \sigma); \mathbf{p}_3(t) = \{\sigma(\sigma(a, \sigma(b), b), \sigma(c, \sigma)), \sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c))\};$$

$$\mathbf{l}_2(t) = \{a, b, c, \sigma(a, b), \sigma(c)\};$$

$$\mathbf{r}_4(t) = \sigma(\sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c))); \mathbf{p}_5(t) = \emptyset;$$

$$\mathbf{l}_4(t) = \{a, b, c, \sigma(a, b), \sigma(c), \sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c)), \sigma(\sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c)))\} \quad \square$$

Let \equiv_k be the equivalence relation in V^T defined as:

$$\forall t, s \in V^T \quad t \equiv_k s \quad \text{iff} \quad Test_k(t) = Test_k(s).$$

Obviously \equiv_k is a subtree invariant relation of finite index. A set $T \subseteq V^T$ is k -Testable (k - T) iff it is the union of some of the equivalence classes of \equiv_k (so k - T are regular). A set is *Locally Testable* (LT) if it is k - T for some k . A tree language is k -Testable in a *Strict* sense (k - TS) if there exist three finite sets \mathbf{R}, \mathbf{L} and \mathbf{P} such that for every $t \in T$:

$$\mathbf{r}_{k-1}(t) \in \mathbf{R}, \mathbf{l}_{k-1}(t) \subseteq \mathbf{L}, \mathbf{p}_k(t) \subseteq \mathbf{P}$$

The family of the k - T languages is the boolean closure of the k - TS family.

IV. SMALLEST k-TS TREE LANGUAGE CONTAINING A POSITIVE SAMPLE

We can associate to the finite subset $S \subset V^T$, the four-tuple $Z_k(S) = (V(S), \mathbf{r}_{k-1}(S), \mathbf{l}_{k-1}(S), \mathbf{p}_k(S))$ being:

$V(S)$ the ranked alphabet from S and $\mathbf{r}_{k-1}(S), \mathbf{l}_{k-1}(S), \mathbf{p}_k(S)$ the $\mathbf{r}_{k-1}, \mathbf{l}_{k-1}$ and \mathbf{p}_k function extensions to sets, that is:

$$\mathbf{r}_{k-1}(S) = \{\mathbf{r}_{k-1}(t) \mid t \in S\}; \mathbf{l}_{k-1}(S) = \bigcup_{t \in S} \mathbf{l}_{k-1}(t); \mathbf{p}_k(S) = \bigcup_{t \in S} \mathbf{p}_k(t)$$

The language defined by $Z_k(S)$ will be denoted as $T_k(S)$

Theorem IV.1. Let S, S' be two finite tree sets, $k \geq 2$ and $T_k(S)$ and $T_k(S')$. Then:

- a) $S \subseteq T_k(S)$
- b) $T_k(S)$ is the smallest k -TS tree set containing S
- c) $S' \subset S \Rightarrow T_k(S') \subseteq T_k(S)$
- d) $T_{k+1}(S) \subseteq T_k(S)$
- e) if $k > 1 + \max_{t \in S} \text{depth}(t)$, then $T_k(S) = S$

Proof.

a) If $t \in S$, then $\mathbf{r}_{k-1}(t) \in \mathbf{r}_{k-1}(S)$, $\mathbf{l}_{k-1}(t) \subseteq \mathbf{l}_{k-1}(S)$ and $\mathbf{p}_k(t) \subseteq \mathbf{p}_k(S)$. Therefore $t \in T_k(S)$.

b) Let's see that for every language k -TS T , if $S \subseteq T$, then $T \subseteq T_k(S)$.

Let's suppose that $T \not\subseteq T_k(S)$. In such a case there exists an $t \in (T_k(S) - T)$ and therefore $\mathbf{r}_{k-1}(t) \notin \mathbf{r}_{k-1}(S)$, $\mathbf{l}_{k-1}(t) \not\subseteq \mathbf{l}_{k-1}(S)$ and $\mathbf{p}_k(t) \not\subseteq \mathbf{p}_k(S)$. If the tree set T is defined by $\mathbf{R}, \mathbf{L}, \mathbf{P}$, then $\mathbf{r}_{k-1}(t) \notin \mathbf{R}$ or $\mathbf{l}_{k-1}(t) \not\subseteq \mathbf{L}$ or $\mathbf{p}_k(t) \not\subseteq \mathbf{P}$. Therefore there exists an $s \in S$ such that $s \notin T$ and $S \not\subseteq T$.

c) $S' \subset S$ implies $\mathbf{r}_{k-1}(S') \subseteq \mathbf{r}_{k-1}(S)$, $\mathbf{l}_{k-1}(S') \subseteq \mathbf{l}_{k-1}(S)$ and $\mathbf{p}_k(S') \subseteq \mathbf{p}_k(S)$.

Therefore $T_k(S') \subseteq T_k(S)$

d) If $t \in T_{k+1}(S)$, then $\mathbf{r}_k(t) \in \mathbf{r}_k(S)$, $\mathbf{l}_k(t) \subseteq \mathbf{l}_k(S)$ and $\mathbf{p}_{k+1}(t) \subseteq \mathbf{p}_{k+1}(S)$. Then

$$\mathbf{r}_{k-1}(t) = \mathbf{r}_{k-1}(\mathbf{r}_k(t)) \in \mathbf{r}_{k-1}(\mathbf{r}_k(S)) = \mathbf{r}_{k-1}(S)$$

$$\mathbf{l}_{k-1}(t) = \mathbf{l}_{k-1}(\mathbf{l}_k(t)) \subseteq \mathbf{l}_{k-1}(\mathbf{l}_k(S)) = \mathbf{l}_{k-1}(S)$$

Finally, in order to prove that $\mathbf{p}_k(t) \subseteq \mathbf{p}_k(S)$ we will distinguish three cases:

If $\text{depth}(t) < k-1$, then $\mathbf{p}_{k+1}(t) = \mathbf{p}_k(t) = \emptyset$ and therefore $\mathbf{p}_k(t) \subseteq \mathbf{p}_k(S)$

If $\text{depth}(t) = k-1$, then $\mathbf{p}_{k+1}(t) = \emptyset$, $\mathbf{p}_k(t) = t \in S$ and therefore $\mathbf{p}_k(t) \subseteq \mathbf{p}_k(S)$

If $\text{depth}(t) > k-1$, then $\mathbf{p}_k(t) = \mathbf{p}_k(\mathbf{p}_{k+1}(t)) \subseteq \mathbf{p}_k(\mathbf{p}_{k+1}(S)) \subseteq \mathbf{p}_k(S)$ and therefore $\mathbf{p}_k(t) \subseteq \mathbf{p}_k(S)$

e) Trivial from the fact that $\mathbf{p}_k(S) = \emptyset$

V. INFERENCE ALGORITHM

Based on the above definitions we propose the following algorithm (Fig. 2) for inference on k - TS tree sets:

INPUT $k \geq 2, S$ finite set of finite trees

OUTPUT $DTA A_k = (Q, V, \delta, F)$

$(V, \mathbf{R}, \mathbf{L}, \mathbf{P}) := (V(S), \mathbf{r}_{k-1}(S), \mathbf{l}_{k-1}(S), \mathbf{p}_k(S))$

$Q := \mathbf{R} \cup \mathbf{L} \cup \mathbf{p}_{k-1}(\mathbf{P})$

$F := \mathbf{R}$

$\forall t \in \mathbf{L} \delta(t) := t$

$\forall \sigma(t_1, \dots, t_n) \in \mathbf{P} \delta_n(\sigma, t_1, \dots, t_n) := \mathbf{r}_{k-1}(\sigma(t_1, \dots, t_n))$

$A_k := (Q, V, \delta, F)$

Fig. 2. Inference algorithm

Examples 2 and 3 illustrate the proposed algorithm. Example 4 shows how the algorithm can be used for the inference of context free grammars from positive structural information.

Example 2. Let $k=2$ and $S = \{\sigma(\sigma(a,b), \sigma(c)), \sigma(\sigma(a, \sigma(a, \sigma(a,b), b), b), \sigma(c, \sigma(c, \sigma(c))))\}$.

$\mathbf{R} = \{\sigma\}$; $\mathbf{L} = \{a, b, c\}$; $\mathbf{P} = \{\sigma(\sigma, \sigma), \sigma(a, b), \sigma(c), \sigma(a, \sigma, b), \sigma(c, \sigma)\}$ and the inferred automaton $A_2 = (Q, V, \delta, F)$ with $Q = \{\sigma, a, b, c\}$, $F = \{\sigma\}$ and the transition functions:

$\delta_0(a) = a$; $\delta_0(b) = b$; $\delta_0(c) = c$; $\delta_1(\sigma, c) = \sigma$; $\delta_2(\sigma, \sigma, \sigma) = \sigma$; $\delta_2(\sigma, a, b) = \sigma$; $\delta_2(\sigma, c, \sigma) = \sigma$;
 $\delta_3(\sigma, a, \sigma, b) = \sigma$.

□

Example 3. Let $k=3$ and S be the same as Example 2

$\mathbf{R} = \{\sigma(\sigma, \sigma)\}$; $\mathbf{L} = \{a, b, c, \sigma(a, b), \sigma(c)\}$; $\mathbf{P} = \{\sigma(\sigma(a, b), \sigma(c)), \sigma(\sigma(a, \sigma, b), \sigma(c, \sigma)), \sigma(a, \sigma(a, \sigma, b), b), \sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c, \sigma)), \sigma(c, \sigma(c))\}$. The inferred automaton $A_3 = (Q, V, \delta, F)$ with

$Q = \{\sigma(\sigma, \sigma), a, b, c, \sigma(a, b), \sigma(c), \sigma(a, \sigma, b), \sigma(c, \sigma)\}$; $F = \{\sigma(\sigma, \sigma)\}$ and

$\delta_0(a) = a$; $\delta_0(b) = b$; $\delta_0(c) = c$; $\delta_1(\sigma, c) = \sigma(c)$; $\delta_2(\sigma, a, b) = \sigma(a, b)$;

$\delta_2(\sigma, \sigma(a, b), \sigma(c)) = \sigma(\sigma, \sigma)$; $\delta_2(\sigma, \sigma(a, \sigma, b), \sigma(c, \sigma)) = \sigma(\sigma, \sigma)$;

$\delta_3(\sigma, a, \sigma(a, \sigma, b), b) = \sigma(a, \sigma, b)$; $\delta_3(\sigma, a, \sigma(a, b), b) = \sigma(a, \sigma, b)$;

$\delta_2(\sigma, c, \sigma(c, \sigma)) = \sigma(c, \sigma)$; $\delta_2(\sigma, c, \sigma(c)) = \sigma(c, \sigma)$.

□

Example 4. Let $k=3$ and S be considered as a set of skeletons of a context-free grammar the same as Example 2.

From the automaton A_3 obtained in Example 3, we obtain a context-free grammar $G = (N, \Sigma, P, S)$ such that $sk(D(G)) = T(A_3)$.

By renaming the states in $Q\text{-}\Sigma$ in order to simplify the notation: $\sigma(\sigma, \sigma) = A$, $\sigma(a, b) = A_2$, $\sigma(c) = B_2$, $\sigma(a, \sigma, b) = A_1$, $\sigma(c, \sigma) = B_1$, we obtain the following grammar:

$N = \{S, A, A_1, B_1, A_2, B_2\}$; $\Sigma = \{a, b, c\}$ and productions

$$\begin{array}{ll}
 S \rightarrow A_1 B_1 \mid A_2 B_2 & \text{or equivalently } S \rightarrow A_1 B_1 \mid A_2 B_2 \\
 A \rightarrow A_1 B_1 \mid A_2 B_2 & A_1 \rightarrow a A_1 b \mid a A_2 b \\
 A_1 \rightarrow a A_1 b \mid a A_2 b & B_1 \rightarrow c B_1 \mid c B_2 \\
 B_1 \rightarrow c B_1 \mid c B_2 & A_2 \rightarrow ab \\
 A_2 \rightarrow ab & B_2 \rightarrow c \\
 B_2 \rightarrow c &
 \end{array}$$

□

The correctness of the algorithm is established by the following theorem:

Theorem V.1. Let S be a finite set of finite trees and A_k ($k \geq 2$) the DTA obtained from the proposed algorithm with input S and k . Then $T_k(S) = T(A_k)$.

Proof.

a) $T_k(S) \subseteq T(A_k)$

By structural induction over the depth of t we will show that if $t \in T_k(S)$, then $\delta(t) = \mathbf{r}_{k-1}(t)$.

If $\text{depth}(t) < k$, $t \in \mathbf{R} \cap \mathbf{L}$ and $\delta(t) = t = \mathbf{r}_{k-1}(t)$

Let $t = \sigma(t_1, \dots, t_n)$ with $\delta(t_i) = \mathbf{r}_{k-1}(t_i)$, $1 \leq i \leq n$ and $\text{depth}(t_i) \geq k$. Then $\sigma(\mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n)) \in \mathbf{P}$ and $\delta(t) = \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) = \delta_n(\sigma, \mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n)) = \mathbf{r}_{k-1}(\sigma(\mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n))) = \mathbf{r}_{k-1}(\sigma(t_1, \dots, t_n)) = \mathbf{r}_{k-1}(t)$

Moreover, as $\mathbf{r}_{k-1}(t) \in \mathbf{R} = \mathbf{F}$, we can conclude $t \in T(A_k)$.

b) $T(A_k) \subseteq T_k(S)$

We will show that $\delta(t) = s$ implies $\mathbf{l}_{k-1}(t) \subseteq \mathbf{L}$, $\mathbf{p}_k(t) \subseteq \mathbf{P}$ and $s = \mathbf{r}_{k-1}(t)$

If $\text{depth}(t) < k$, then $t \in \mathbf{L}$ and $\mathbf{l}_{k-1}(t) \subseteq \mathbf{L}$, $\mathbf{p}_k(t) = \emptyset \subseteq \mathbf{P}$, $s = t = \mathbf{r}_{k-1}(t)$

If $\text{depth}(t) \geq k$ let $t = \sigma(t_1, \dots, t_n)$ by induction hypothesis $1 \leq i \leq n$ $\mathbf{l}_{k-1}(t_i) \subseteq \mathbf{L}$, $\mathbf{p}_k(t_i) \subseteq \mathbf{P}$ and

$\delta(t_i) = \mathbf{r}_{k-1}(t_i)$. But then $\delta(t) = \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) = \delta_n(\sigma, \mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n)) = \mathbf{r}_{k-1}(\sigma(t_1, \dots, t_n))$

But $\sigma(\mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n)) \in \mathbf{P}$ and $s = \mathbf{r}_{k-1}(t)$.

Moreover, $\mathbf{l}_{k-1}(t) = \cup_{i \in \{1, \dots, n\}} \mathbf{l}_{k-1}(t_i) \subseteq \mathbf{L}$,

$\mathbf{p}_k(t) = ((\cup_{i \in \{1, \dots, n\}} \mathbf{p}_k(t_i)) \cup \{\sigma(\mathbf{r}_{k-1}(t_1), \dots, \mathbf{r}_{k-1}(t_n))\}) \subseteq \mathbf{P}$.

Finally, if $t \in T(A_k)$, then $\delta(t) \in \mathbf{F} = \mathbf{R}$ and, therefore

$\mathbf{r}_{k-1}(t) \in \mathbf{R}$ also. Consequently $t \in T_k(S)$. □

The fact that, given the value of k and a ranked alphabet, the resulting class of the k -TS tree languages is finite, together with the above theorem we can obtain the following result.

Theorem V.2. The proposed algorithm identifies the family of k -TS tree languages in the limit from positive samples.

Theorem V.3. Let S be a finite set of trees and let $Z_k(S) = (V(S), \mathbf{r}_{k-1}(S), \mathbf{l}_{k-1}(S), \mathbf{p}_k(S))$ the four-tuple which defines the smallest k -TS tree set containing S . The proposed algorithm can be implemented to run in $O(m^{k-1} n \log|\mathbf{p}_k(S)|)$ time, where n is the total number of nodes of the trees in S and m is the greatest arity of the symbols in $V(S)$.

This bound comes from the fact that, by using the appropriate data structures to represent the different sets involved in the construction of $Z_k(S)$ and A_k , the required find-insert operations can be carried out in at most $O(m^{k-1} \log|\mathbf{p}_k(S)|)$. From the fact that $|\mathbf{p}_k(S)| \leq n$ we obtain an inference time $O(m^{k-1} n \log n)$.

VI. CONCLUSION

Sakakibara [6] has recently proposed a tree language inference algorithm as a generalization of 0-reversible string language inference algorithm [8]. This algorithm identifies the 0-reversible tree language family in the limit.

The family of k -Testable in the Strict sense tree languages, which is the subject of this paper, constitutes a broad subclass of recognizable tree languages which is incomparable with the 0-reversible language class. However, Sakakibara's, algorithm in relation to the learning Context-Free languages is of interest in that any Context-Free language can be generated by a reversible Context-Free grammar (such that the set of skeletons is 0-reversible). This allows for the identification in the limit of the class of Context-Free languages from positive structural information, as long as such information corresponds to skeletons from a reversible Context-Free grammar.

One open question in relation to the algorithm we have presented is the possibility of characterizing those Context-Free language which could be generated by grammars whose set of skeletons is k -Testable in the Strict sense.

VII. REFERENCES

- [1] J.M. Brayer and K.S. Fu. "A note on the k -tail method of tree grammar inference". *IEEE Trans. System Man and Cybernetics*, Vol. SMC-7, pp.293-300, 1977.
- [2] P. García and E. Vidal. "Inference of k -Testable Languages in the Strict Sense and application to Syntactic Pattern Recognition.". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-12, No 9, pp. 920-925, 1990.
- [3] F. Gécsec and M. Steinby. "Tree Automata" *Akadémiai Kiadó*, Budapest, 1984.
- [4] B. Levine. "Derivatives of tree sets with applications to Grammatical Inference. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No 3, 1981.
- [5] Y. Sakakibara. "Learning context-free grammars from structural data in polynomial time". *Theoretical Computer Science*, 76, pp. 223-242, 1990.

[6] Y. Sakakibara. "Efficient Learning of Context-Free Grammars from Positive Structural Examples". *Information and Computation*, 97, pp. 23-60, 1992.

[7] J.W. Thatcher. "Generalized Sequential Machine Maps". *Journal of Computer and System Sciences*, 4, pp. 339-367, 1970.

[8] D. Angluin. "Inference of reversible languages" *Journal of the Association for Computing Machinery.*, 29, pp. 741-765, 1982