

*Advances in Pattern Recognition of Lecture Notes in Computer Science LNCS 2013* (Springer, 2001 ) pp. 230–239.

- [19] P. Pudil, F. J. Ferri, J. Novovičová, and J. Kittler, Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions, in *Proceedings of 12th International Conference on Pattern Recognition* (1994) pp. 279–283
- [20] M. Kudo and J. Sklansky, Classifier-Independent Feature Selection for Two-stage Feature Selection in *Advances in Pattern Recognition of Lecture Notes in Computer Science LNCS 1451* (Springer, 1998 ) pp. 548–554
- [21] M. Kudo and M. Shimbo, Optimal Subclasses with Dichotomous Variables for Feature Selection and Discrimination, *IEEE Trans. Systems, Man, and Cybern.*, Vol. 19(1989) pp. 1194–1199
- [22] M. Kudo, S. Yanagi and M. Shimbo, Construction of Class Regions by a Randomized Algorithm: A Randomized Subclass Method, *Pattern Recognition* Vol 29(1996) pp. 581–588.
- [23] J. H. Friedman, A Recursive Partitioning Decision Rule for Nonparametric Classification, *IEEE Transactions on Computers*, Vol 26(1977) pp. 404–408.
- [24] C. Blake and C. Merz, UCI Repository of machine learning databases [<http://www.ics.uci.edu/~mlearn/MLRepository.html>] (1998).

---

## Inference of K-Piecewise Testable Tree Languages

Damián López, José Ruiz and Pedro García  
*Departamento de Sistemas Informáticos y Computación*  
*Universidad Politécnica de Valencia, 46071 Valencia, SPAIN*  
 E-mail: {dlopez, jruiz, pgarcia}@dsic.upv.es

---

## Contents

1	Introduction	341
2	Notation and Definitions	343
3	Piecewise Testable Tree Languages	344
	3.1 Inference Algorithm	347
	3.2 Time Complexity	349
4	Conclusions and Future Work	349
5	Acknowledgement	349

## References

## 1 Introduction

In the field of pattern recognition it is important to represent the objects as accurate as possible. The existence of primitives that can express different features of the objects help to obtain this accuracy.

The syntactic (or structural) representation of objects has been approached in different ways [1, 2]. There are primitives which have a high

representative power (web, plex, or graph grammars), but as the more representative power a primitive has, the more time it needs to operate, strings have been traditionally used to represent objects. Their expressive power is low, but the cost of operations on them, such as classification algorithms, is also low.

The development of efficient algorithms to manipulate primitives with greater representative power is a goal in the field of pattern recognition. In this way, algorithms that perform error correcting analysis for multidimensional languages [12, 9] permit the use of primitives such as trees in the modeling of the different classes that take place in a recognition task.

Due to the limits established by Gold in the field of grammatical inference [7], certain classes of string languages require the use of a structural sample to be inferred [18, 19, 15]. Although those algorithms could easily be adapted as to be considered tree language inference algorithms, this type of algorithms has been specifically considered in literature [5, 6, 13, 3, 11] and several of them have been implemented and used in practice (i.e. [10, 11]).

Another classical approach to string language inference consists in the use of only positive samples in the task of learning. Several important subfamilies of regular languages, like the  $k$ -Testable in the Strict Sense or the  $k$ -Piecewise Testable Languages, have been inferred in this way [4, 17]. Their common characteristic is that those languages can be defined by means of the segments (using consecutive symbols) or the subwords (using non consecutive symbols) that appear in their words. This features have proved to be fruitful in pattern recognition tasks [4, 16].

The aim of this work is to define a new family of tree languages that will be called  $k$ -Piecewise Testable Tree Languages ( $k$ -PTTL), where  $k$  is a strictly positive integer, and to propose an algorithm that, for a given value of  $k$ , infers the family of  $k$ -PTTL from positive data in the limit.

The definition of  $k$ -PTTL uses the concept of *tree subword*. Informally, a  $k$ -tree subword of a tree  $t$  is a tree of depth  $k$ , which is formed by *forks* of height one that appear in  $t$  and maintain the same node hierarchy. A language is  $k$ -PTTL if it is saturated by a tree congruence in which two trees are equivalent if they have the same set of  $k$ -tree subwords.

The work is structured as follows, first we introduce some notation and the definitions that will be used along the work. Then, we define the new family of tree languages and establish some useful properties. After wards, we propose the inference algorithm and prove that it infers the desired family, the time complexity is also obtained. The conclusions and some lines of future work end this paper.

## 2 Notation and Definitions

Let  $\mathcal{N}$  be the set of positive integers and let  $(\mathcal{N}^*, \cdot)$  be the free monoid with the concatenation as the binary operation and  $\lambda$  as the identity. For  $u, w \in \mathcal{N}^*$ , we write  $u \leq w$  if and only if there exists  $v \in \mathcal{N}^*$  such that  $w = u \cdot v$ ;  $u < w$ , if  $u \leq w$  and  $u \neq w$ . The length of a word  $x$  is denoted by  $|x|$ .

A tree domain is a set  $D \subseteq \mathcal{N}^*$  that fulfills the following conditions:

- $v \in D$  and  $u < v$  implies that  $u \in D$ .
- If  $u \cdot i \in D, i \in \mathcal{N}$  then  $u \cdot j \in D \quad \forall j: 1 \leq j \leq i$ .

Given an alphabet  $V$ , a *ranked alphabet* is a pair  $(V, r)$ , where  $r$  is a finite relation in  $(V \times \mathcal{N})$ . Let  $V_n = \{\sigma \in V \mid (\sigma, n) \in r\}$  and let  $V^T$  be the set of finite trees whose nodes are labelled with symbols in  $V$ . A tree can inductively be defined as follows:

$$V_0 \subseteq V^T \\ \sigma(t_1, \dots, t_n) \in V^T \quad \forall t_1, \dots, t_n \in V^T, \quad \sigma \in V_n$$

We define the *extended ranked alphabet* of  $V$ , denoted by  $V_e$ , as the pair  $(V, r_e)$ , where  $r_e = r \cup \{(a, 0) \mid \exists n: a \in V_n\}$ .

Note that a tree  $t \in V^T$  can be seen as a mapping  $t: D \rightarrow V$ , where  $D$  is the tree domain of  $t$ .

The size of a tree  $t$ , denoted by  $\|t\|$ , is inductively defined as follows:

$$\|a\| = 0 \quad \forall a \in V_0 \\ \|\sigma(t_1, \dots, t_n)\| = 1 + \sum_{i=1}^n \|t_i\| \quad \forall t_1, \dots, t_n \in V^T, \quad \sigma \in V_n$$

The depth of a tree  $t$ , denoted by  $depth(t)$ , is defined as  $Max(|x|), x \in dom(t)$ .

We denote by  $V^{Tk}$  the set of trees  $V^{Tk} = \{t \mid t \in V^T \wedge depth(t) = k\}$ .

The set of subtrees of  $t$ ,  $Sub(t)$ , is defined as follows:

$$Sub(a) = \emptyset \quad \forall a \in V_0 \\ Sub(\sigma(t_1, \dots, t_n)) = \{t_1, \dots, t_n\} \cup \bigcup_{i=1}^n Sub(t_i), \\ \forall t_1, \dots, t_n \in V^T, \quad \sigma \in V_n$$

The subtree of  $t \in V^T$  rooted at  $x \in dom(t)$  is defined as:

$$t/x = \begin{cases} dom(t/x) = \{y \mid x \cdot y \in dom(t)\} \\ (t/x)(y) = t(x \cdot y) \end{cases}$$

The substitution of the subtree  $t/x$  with  $s \in V^T$  is defined as follows:

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

An equivalence relation  $\equiv$  over  $V^T$  is *subtree invariant* whenever  $t_1 \equiv t_2$  implies  $t(x \leftarrow t_1) \equiv t(x \leftarrow t_2)$ ,  $\forall t \in V^T, x \in \text{dom}(t)$ . A tree set is regular if and only if is the union of some classes from a subtree invariant, finite index, equivalence relation.

A *deterministic tree automaton* is a four-tuple  $A = (Q, V, \delta, F)$  where  $Q$  is a finite set of states;  $V$  is a ranked alphabet with  $Q \cap V = \emptyset$ ;  $F \subseteq Q$  is the set of final states and  $\delta = (\delta_0, \dots, \delta_m)$  is a finite set of transitions defined as follows:

$$\begin{aligned} \delta_n : (V_n \times (Q \cup V_0)^n) &\rightarrow Q & n = 1, \dots, m \\ \delta_0(a) = a & & \forall a \in V_0 \end{aligned}$$

$\delta$  can be extended to operate on trees in the following way:

$$\begin{aligned} \delta : V^T &\rightarrow Q \cup V_0 \\ \delta(\sigma(t_1, \dots, t_n)) &= \delta_n(\sigma, \delta(t_1), \dots, \delta(t_n)) \text{ if } n > 0 \\ \delta(a) &= a \end{aligned}$$

The language accepted by a tree automaton  $A$  is  $L(A) = \{t \in V^T \mid \delta(t) \in F\}$

Given a tree automaton  $A = (Q, V, \delta, F)$  and a state  $q \in Q$ , we say that  $q$  is at level  $i$  if and only if  $i = \min\{\text{depth}(t) \mid t \in V^T, \delta(t) = q\}$

### 3 Piecewise Testable Tree Languages

**Definition 3.1** Given  $t, s \in V^T$ , let  $P^t(s)$  be defined as follows:

$$\begin{aligned} P^t(a) &= \{x \in \text{dom}(t) \mid t(x) = a\} & \forall a \in V \\ P^t(\sigma(t_1, \dots, t_n)) &= \{x \in \text{dom}(t) \mid \exists z_1, \dots, z_n \\ & : x = i \cdot z_i \in P^t(t_i)\} & \forall \sigma \in V_n, \\ & & \forall t_1, \dots, t_n \in V^T \end{aligned}$$

**Example 3.2** Consider the tree  $t$  of the Figure 1, then:

$$\begin{aligned} P^t(b) &= \{13, 1212, 1222\} \\ P^t(\sigma(a, b)) &= \{121, 122\} \\ P^t(\sigma(a, \sigma(a, b), b)) &= \{1\} \\ P^t(\sigma(\sigma(a, \sigma(a, b), b)), \sigma(c)) &= \{\lambda\} \end{aligned}$$

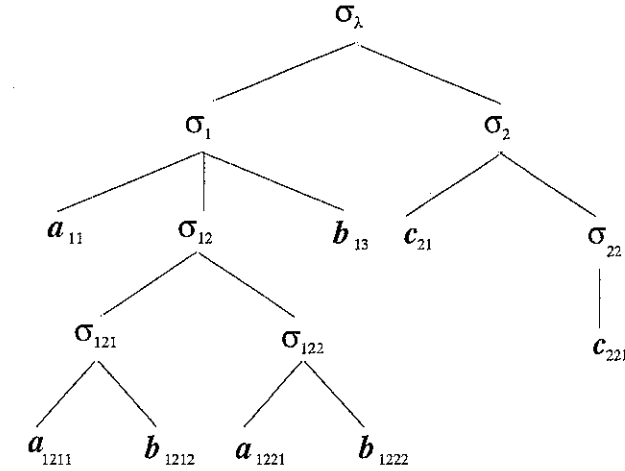


Figure 1: Tree example where the subscripts denote the domain of the node.

**Definition 3.3** Given a tree  $t \in V^T$ , the set  $PSub(t)$  is defined as:

$$PSub(t) = \{s \in V_e^T \mid P^t(s) \neq \emptyset\}$$

and let also  $kPSub(t)$  be defined as follows:

$$kPSub(t) = \begin{cases} t & \text{if } \text{depth}(t) < k \\ s \in V_e^T \mid s \in PSub(t) & \text{depth}(s) = k \\ \text{other wise} & \text{other wise} \end{cases}$$

**Example 3.4** Given the tree of the Figure 1, the set  $2PSub(t)$  and some members of  $3PSub(t)$  are shown below:

$$\begin{aligned} 2PSub(t) &= \{\sigma(\sigma(a, b), \sigma), \sigma(\sigma, \sigma(a, b)), \sigma(\sigma(a, b), \sigma(a, b)), \sigma(a, \sigma(\sigma, \sigma), b) \\ & \sigma(a, \sigma(a, b), b), \sigma(\sigma(a, \sigma, b), \sigma(c, \sigma)), \sigma(\sigma(a, \sigma, b), \sigma(c)) \\ & \sigma(\sigma(a, \sigma, b), \sigma), \sigma(\sigma(\sigma, \sigma), \sigma(c, \sigma)), \sigma(\sigma(a, b), \sigma(c, \sigma)) \\ & \sigma(\sigma(\sigma, \sigma), \sigma(c)), \sigma(\sigma(a, b), \sigma(c)), \sigma(\sigma(\sigma, \sigma), \sigma), \sigma(\sigma(a, b), \sigma) \\ & \sigma(\sigma, \sigma(c, \sigma)), \sigma(\sigma, \sigma(c)), \sigma(c, \sigma(c))\} \end{aligned}$$

$$\begin{aligned} 3PSub(t) &= \{\sigma(\sigma, \sigma(c, \sigma(c))), \sigma(\sigma(a, \sigma, b), \sigma(c, \sigma(c))), \sigma(\sigma(\sigma, \sigma), \sigma(c, \sigma(c))) \\ & \sigma(\sigma(a, \sigma(a, b), b), \sigma(c, \sigma(c))), \sigma(\sigma(\sigma(a, b), \sigma(a, b)), \sigma(c, \sigma(c))) \\ & \sigma(\sigma(\sigma(a, b), \sigma(a, b)), \sigma(c, \sigma)), \sigma(\sigma(\sigma(a, b), \sigma), \sigma(c, \sigma)) \\ & \sigma(\sigma(a, \sigma(\sigma, \sigma), b), \sigma), \sigma(\sigma(a, \sigma(a, b), b), \sigma) \\ & \sigma(a, \sigma(\sigma, \sigma(a, b)), b), \dots\} \end{aligned}$$

As  $\text{depth}(t) = 7$ , the set  $PSub(t)$  is the union of the sets  $kPSub(t)$ ,  $k : 0..7$

**Definition 3.5** Given  $t, s \in V^T$ , the equivalence relation  $\equiv_k$  is:

$$t \equiv_k s \iff kPSub(t) = kPSub(s)$$

The equivalence class of the tree  $t$  will be denoted by  $[t]$ . Clearly, the above equivalence relation is subtree invariant, and has the following properties:

- 1.-  $\equiv_{k+1}$  refines  $\equiv_k$ .
- 2.- Given  $t, s \in V^T$ ,  $x \in dom(t)$ , if  $t(x \leftarrow s) \in [s]$ , then  $t(x \leftarrow (t(x \leftarrow \dots t(x \leftarrow s) \dots))) \in [s]$ .
- 3.- Any equivalence class  $[t]$  is either a singleton or has infinite members.

**Definition 3.6** A tree language is said to be  $k$ -Piecewise Testable ( $k$ -PTTL) if and only if it is the union of some equivalence classes of  $\equiv_k$ . A language  $L$  is said to be Piecewise Testable if there exist a value of  $k$  for which  $L$  is  $k$ -Piecewise Testable.

Given two trees  $t, s \in V^T$ , we say that  $t \leq_k s$  if and only if  $jPSub(t) \subseteq jPSub(s)$ ,  $\forall j \leq k$ . This relationship establishes a partial order in  $V^T$ , and it is easy to see that  $t \in PSub(s)$  implies  $t \leq_k s$ .

When  $\leq_k$  is extended to classes of equivalence, the quotient set  $(V^T / \equiv_k, \leq_k)$  is a finite lattice with an absolute maximum (the class of trees  $t \equiv_k [V^{T^k}]$ ), and an absolute minimum (the class of the empty tree). Considering the tree  $t$  and the order relation defined, each subtree  $s = \sigma(s_1, s_2, \dots, s_n)$  in a postorder traversal of the tree  $t$ , fulfills that  $s_i \leq_k s$ ,  $i: 1 \dots n$ .

From the above properties is easy to prove that:

- 1.- If  $L$  is a  $k$ -PTTL, then  $L$  is  $j$ -PTTL  $\forall j \leq k$ .
- 2.- If  $L$  is Piecewise Testable, then  $L$  is regular.
- 3.- Given a tree automaton  $A$  that accepts a  $k$ -PTTL,  $A$  has no transitions  $\delta(\sigma, u_1, \dots, q, \dots, u_n) = p$  where  $p$  is at level  $i$ ,  $q$  is at level  $j$ , and  $i < j$ .
- 4.- Given  $V$  a ranked alphabet, the class of  $k$ -PTTL has a finite number of languages, bounded by  $|\mathcal{P}(\mathcal{P}(V_e^{T^k}))| \leq 2^{2^{|V_e|^{k \cdot bf}}}$  where  $bf$  is the maximum branching factor and where  $\mathcal{P}$  denotes the power set.

### 3.1 Inference Algorithm

Considering the equivalence relation defined above, given a set of trees  $\mathcal{T} = \{t_1, t_2, \dots, t_k\}$ , it is possible to obtain a  $k$ -PTTL  $L_k(\mathcal{T})$  as it is shown below:

$$\forall t \in V^T \quad t \in L_k(\mathcal{T}) \iff \exists s \in \mathcal{T} \quad t \equiv_k s$$

This definition has the following properties:

- 1.-  $S \subseteq L_k(\mathcal{T})$ .
- 2.-  $L_k(\mathcal{T})$  is the smallest  $k$ -PTTL that contains  $S$ .

*Proof* Let suppose that exist a  $k$ -PTTL  $L$  such that,  $S \subseteq L \subset L_k(\mathcal{T})$ . By definition 3.6,  $L$  must contain all the equivalence classes in  $S$ . Let  $t \in V^T$ ,  $t \in L_k(\mathcal{T}) - L$ , then there exists no  $s \in S$  such that  $[s] = [t]$ , which is a contradiction.  $\square$

- 3.-  $L_k(\mathcal{T}') \subseteq L_k(\mathcal{T})$  whenever  $\mathcal{T}' \subseteq \mathcal{T}$ .
- 4.-  $L_{k+1}(\mathcal{T}') \subseteq L_k(\mathcal{T})$ .
- 5.- If  $k > \text{Max}_{t \in \mathcal{T}}(\text{depth}(t))$  then  $L_k(\mathcal{T}') = \mathcal{T}'$ .

We propose the algorithm depicted in Figure 2, which for a given positive sample  $\mathcal{T}$  of the target language obtains a  $k$ -PTTL consistent with the sample that recognizes a subset of  $L_k(\mathcal{T})$ .

**Example 3.7** As an example of run, let  $k = 2$  and consider the following set of trees  $\mathcal{T}$  as input:

$$\mathcal{T} = \left\{ \begin{array}{l} t_1 = \sigma(\sigma(a, \sigma(\sigma(\sigma(a), \sigma(a)), \sigma(a)), b), \sigma(c, \sigma(c, \sigma(c))))), \\ t_2 = \sigma(\sigma(\sigma(\sigma(\sigma(a), \sigma(a)), \sigma(a)), \sigma(a)), \sigma(c, \sigma(c, \sigma(c, \sigma(c))))), \\ t_3 = \sigma(\sigma(a, \sigma(\sigma(a), \sigma(a)), b), \sigma(c, \sigma(c, \sigma(c)))) \end{array} \right\}$$

The first iteration of the algorithm, using  $t_1$  as input, outputs the following automaton:

$$\begin{array}{ll} \delta(\sigma, a) = q_1 & \delta(\sigma, c) = q_5 \\ \delta(\sigma, q_1, q_1) = q_2 & \delta(\sigma, c, q_5) = q_6 \\ \delta(\sigma, q_2, q_1) = q_3 & \delta(\sigma, c, q_6) = q_7 \\ \delta(\sigma, a, q_3, b) = q_4 & \delta(\sigma, q_4, q_7) = q_8 \in F \end{array}$$

<i>Input :</i>	A tree set $\mathcal{T} = \{t_1, t_2, \dots, t_k\}$ . A value $k$ .
<i>Output :</i>	A tree automaton $A = (Q, V, \delta, F)$ : $L(A) \subseteq L_k(\mathcal{T})$ .
<i>Method :</i>	$Q = kPSub(Sub(\mathcal{T}))$ $F = kPsub(\mathcal{T})$ $\forall t \in \mathcal{T}$ $\forall s = \sigma(u_1, u_2, \dots, u_p) \in Sub(t) \cup \{t\}$ $\delta \supseteq \delta(\sigma, kPSub(u_1),$ $\quad kPSub(u_2), \dots, kPSub(u_p)) = kPSub(s)$ $\text{end}\forall$ $\text{end}\forall$
	<i>EndMethod.</i>

Figure 2: k-Piecewise Testable Tree Language Inference algorithm

When  $t_2$  is processed, the following productions are added to the output automaton:

$$\begin{aligned}\delta(\sigma, q_3, q_1) &= q_3 \\ \delta(\sigma, c, q_7) &= q_7 \\ \delta(\sigma, a, q_3, q_7) &= q_9 \in F\end{aligned}$$

Finally, after  $t_3$  is processed, the following productions are added:

$$\begin{aligned}\delta(\sigma, a, q_2, b) &= q_{10} \\ \delta(\sigma, q_{10}, q_7) &= q_{11} \in F\end{aligned}$$

**Theorem 3.8** *The Algorithm of figure 2 identifies the class of k-PTTL in the limit.*

*Proof.* The algorithm outputs an automaton that recognizes a superset of the input whenever a new structure  $s = \sigma(s_1, \dots, s_n)$ , such that  $\exists i : 1..n : s_i \equiv_k s$ , is processed.

The finiteness of the lattice  $(V^T / \equiv_k, \leq_k)$ , and the ascending path the algorithm follows through the lattice assure the convergence.  $\square$

### 3.2 Time Complexity

In the process of inferring a  $k$ -PTTL the algorithm associates to every state of the automaton a set of piecewise subtrees. The number of possibly associated sets is bounded above by  $|V_e|^{k \cdot bf}$ , where  $bf$ , the maximum branching factor of the tree, is considered as a constant.

If we call  $t$  to the tree in  $\mathcal{T}$  with the greatest amount of nodes,  $kPSub(t) \leq \|t\|^{k \cdot bf}$ . Since the processing of each  $t \in \mathcal{T}$  implies the creation of  $\|t\|$  transitions at most, if we consider  $k$  as a constant, as the number of different sets of piecewise subtrees is bounded, the time complexity of the algorithm could be expressed as  $\mathcal{O}(p(\|t\|) \cdot n)$ , where  $p$  is a polynomial and  $n$  denotes the number of samples in  $\mathcal{T}$ .

## 4 Conclusions and Future Work

The availability of powerful structural primitives to represent the objects is a key factor in pattern recognition. Multidimensional (tree) representations allow to model important features in disciplines like speech recognition and natural language processing. *Tree adjoining grammars* [8] have been used in this field, for instance to capture semantic characteristics of verbs inside sentences [14].

In this work, the generalization of the notion of subword to tree structures allow us to characterize the class of  $k$ -Piecewise Testable Tree Languages. Together with the characterization, and the extension of several string languages properties to tree languages, a polynomial time inference algorithm is proposed which learns the class in the limit. The time complexity of the algorithm is also obtained.

This new family of formal languages allows the modelling of relations between components of the sentences that could be useful, not only in linguistics, but also in other pattern recognition tasks.

## 5 Acknowledgement

This work has been partially supported by the Spanish CICYT under contract TIC2000-1153

## References

- [1] K. S. Fu *Syntactic pattern recognition and applications*, (Prentice-Hall, 1982)
- [2] R. González and M. Thomason, *Syntactic Pattern Recognition An Introduction*, (Addison-Wesley Publishing Company, 1978).
- [3] H. Fernau, Learning tree languages from text, *Technical Report, Wilhelm-Schickard-Institut für Informatik*, Universität Tübingen, 2001
- [4] P. García and E. Vidal, Inference of k-testable languages in the strict sense and application to syntactic pattern recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 12 No 9 (1990) pp. 920-925
- [5] P. García, Learning k-testable tree sets from positive data, *Technical Report DSIC II/46/1993*, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 1993.
- [6] P. García and J. Oncina, Inference of recognizable tree sets, *Technical Report DSIC II/47/1993*, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 1993.
- [7] E. M. Gold, Language identification in the limit, *Information and Control* Vol 10 (1967) pp. 447-474.
- [8] A. K. Joshi and Y. Schabes, Tree-Adjoining Grammars, in G. Rozenberg and A. Salomaa (eds) *Handbook of formal Languages Vol 3*, (Springer, 1997) pp. 69-123.
- [9] D. López, J. M. Sempere and P. García, Error correcting analysis for tree languages, *International Journal on Pattern Recognition and Artificial Intelligence*, Vol 14, No. 3 (2000) pp. 357-368.
- [10] D. López and I. Piñaga, Syntactic pattern recognition by error correcting analysis on tree automata, in F. Ferri, J. Iñesta, A. Amin and P. Pudil (eds) *Advances in Pattern Recognition*, (Springer-Verlag, 2000) pp. 133-142.
- [11] D. López and S. España, Error correcting tree language inference, *Pattern Recognition Letters*, Vol 23 No (1-3) (2002) pp. 1-12
- [12] S. Y. Lu and K. S. Fu, Error-correcting tree automata for syntactic pattern recognition, *IEEE Transactions on Computers*, C-27 No 11 (1978) pp. 1040-1053
- [13] E. Mäkinen, On inferring linear single-tree languages, *Information Processing Letters* Vol 73 No (1-2) (2000) pp. 1-3
- [14] M. Palmer, J. Rosenzweig and W. Schuler, Capturing motion verb generalizations in synchronous tree adjoining grammars, in P. Saint-Dizier (ed) *Predicative forms in natural language and in lexical knowledge bases*, (Kluwer Academic Publishers, 1999) pp. 229-256
- [15] V. Radhakrishnan and G. Nagaraja, Inference of regular grammars via skeletons, *IEEE Trans. System, Man and Cybernetics* SCM-17 (1987) pp. 982-992.
- [16] J. Ruiz, Familias de lenguajes explorables: Inferencia inductiva y caracterizaciones algebraicas, *PhD. Thesis* (in spanish), Universidad Politécnica de Valencia, 1998.
- [17] J. Ruiz and P. García, Learning k-Piecewise Testable Languages from Positive Data, in Miclet and de la Higuera (eds) *Grammatical Inference: Learning Syntax from Sentences*, LCNS 1147 (1996) pp. 203-210.
- [18] Y. Sakakibara, Learning Context-Free Grammars from Structural Data in Polynomial Time, *Theoretical Computer Science* Vol 76 (1990) pp. 223-242.
- [19] Y. Sakakibara, Efficient Learning of Context-Free Grammars from Positive Structural Examples, *Information and Computation* Vol 97 (1992) pp. 23-60.