Inference of Reversible Tree Languages

Damián López, José M. Sempere, and Pedro García

***Abstract*—In this paper, we study the notion of k-reversibility and k-testability when regular tree languages are involved. We present an inference algorithm for learning a k-testable tree language that runs in polynomial time with respect to the size of the sample used. We also study the tree language classes in relation to other well known ones, and some properties of these languages are proven.**

***Index Terms*—Grammatical inference, pattern recognition, regular tree languages.**

I. Introduction

In THE FIELD of machine learning, one classical problem is to obtain the rules or patterns that model the structure of a set. When the objects are represented by elements of a formal language, the patterns determine the rules of a grammar which are able to generate the language. The search for these rules is known as *grammatical inference*. In the grammatical inference process, the information about the target language is given as two sets; a positive sample that consists of objects belonging to the language, and a negative sample of objects that do not belong to it.

There exist many factors that limit the learning of these rules [1]. Of these factors, it is important to note that without negative sample it is impossible to infer the class of regular languages. However, there exist several inference methods that use positive samples only. These methods lead to the characterization of important subfamilies of regular languages, for example, the k-testable in the strict sense [2], or the k-piecewise testable languages [3]. In an intuitive way, these languages model the sequences of symbols that can appear in the words of the language (consecutive symbols when k-testable in the strict sense languages are involved, or, nonconsecutive symbols when k-piecewise testable languages are considered).

Another classical approximation to formal string language inference is the use of structured sample during the learning process. In this approach, tree-like descriptions of the words are given to the inference method. This description usually gives relevant information which is related to the training data, for instance, one of the possible ways the data can be generated [4]–[6]. These inference methods can be modified for inferring tree languages. However, such modified methods are not the only way to infer tree languages, and there exist several specific algorithms for these languages in the literature [7]–[13].

The inference of tree languages is related to the inference of context-free string languages using a structural sample, but the development of specific tree language learning algorithms should open new possibilities for the characterization of subclasses of the context-free languages. Sakakibara [6] shows that every context-free language has a grammar in a zero-reversible normal form, and he provides an algorithm to learn the context-free class by using structured sample. Besides, the characterization of tree language classes should also open the possibility to learn context-free languages with some desirable properties that could permit the development of efficient parsers.

Learning results have been widely applied to the field of *pattern recognition* (PR). The syntactic approximation to PR tasks [14], [15] attempts to model the structure, instead of measuring features of the objects to be classified. Several approaches to syntactic representation of patterns have been proposed, but those with the greatest representation power (mainly based on several kinds of graphs), are also the most complex to manipulate. Therefore, in an attempt to reduce the time complexity of the process, linear representations have been used extensively (chains of symbols) due to the existence of efficient algorithms that deal with these representations. Research exists [2], [16]–[19] which propose solutions to PR tasks under a syntactic approach with good results.

Nevertheless, in a syntactical approach to PR, the development of algorithms to perform operations on more complex objects than strings of symbols, permits a more suitable representation of the objects of the domain. Thus, the existence of error-correcting parsers for tree languages [20], [21] permits the use of these languages in recognition tasks [12], [22].

In this paper, two new tree language classes are studied. The first one generalizes the notion of k-reversibility [23] to tree languages. k-reversible languages could be seen as a special case of the distinguishable languages introduced by Fernau [24]. Fernau presents a general scheme that could be used to infer this class. In our paper we propose a more efficient algorithm to learn k-reversible languages. We also study k-reversible languages in relation to other well-known tree language classes. The second class of tree languages characterized in this paper extends the concept of k-testability from string languages [18]. k-testable tree languages are also studied in relation to other classes.

The paper is structured as follows. In Section II, we establish the notation and the basic definitions used throughout the paper. In Section III, we define the class of k-reversible tree languages and we present some characterizations and results. In Section IV the inference algorithm for this class is proposed, one example of run is shown, and the polynomial time complexity is proved. In Section V, this class is studied in relation to other well-known

Manuscript received January 17, 2003; revised July 1, 2003 and February 4, 2004. This work was supported by Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Contract TIC2000-1153. This paper was recommended by Associate Editor J. B. Oommen.

The authors are with the Departamento de Sistemas Informáticos y Computación, Universidad Politécnica of Valencia, 46071 Valencia, Spain (e-mail: dlopez@dsic.upv.es; jsempere@dsic.upv.es; pgarcia@dsic.upv.es).

Digital Object Identifier 10.1109/TSMCB.2004.827190

regular tree languages. The class of k-testable tree languages is also characterized and related to all the previous classes. Finally, the conclusions of the paper are presented.

II. Notation and Definitions

Let a *ranked alphabet* be the association of an alphabet V together with a finite relation r in $V \times \mathbb{N}$. We denote the subset $\{\sigma \in V : (\sigma, n) \in r\}$ with V_n.

The set V^T of trees over V, is defined inductively as follows:

$$a \in V^T \text{ for every } a \in V_0$$
$$\sigma(t_1, \ldots, t_n) \in V^T \text{ whenever } \sigma \in V_n \text{ and } t_1, \ldots, t_n \in V^T \quad (n > 0)$$

and let a *tree language* over V be defined as a subset of V^T.

Let $\mathbb{N}^*$ be the set of finite strings of natural numbers, separated by dots, formed using the catenation as the composition rule and the empty word λ as the identity. Let the prefix relation $\leq$ in $\mathbb{N}^*$ be defined by the condition that $u \leq v$ if and only if $u \cdot w = v$ for some $w \in \mathbb{N}^*$ ($u, v \in \mathbb{N}^*$). A finite subset D of $\mathbb{N}^*$ is called a *tree domain* if

$$u \leq v \text{ where } v \in D \text{ implies } u \in D, \text{ and}$$
$$u \cdot i \in D \text{ whenever } u \cdot j \in D \ (1 \leq i \leq j).$$

Each tree domain D could be seen as an unlabeled tree whose nodes correspond to the elements of D where the hierarchy relation is the prefix order. Thus, each tree t over V can be seen as an application $t : D \to V$. The set D is called the *domain of the tree* t, and denoted by $\mathrm{dom(t)}$. The elements of the tree domain $\mathrm{dom}(t)$ are called *positions* or *nodes* of the tree t. We denote with $t(x)$ the label of a given node x in $\mathrm{dom}(t)$.

Example 2.1: As an example, given $V = \{(a,0), (b,0), (\sigma,2), (\tau,1)\}$, a valid tree over V is $t = \sigma(\sigma(a, \sigma(a,b)), \tau(a))$.

The tree domain of t is $\mathrm{dom}(t) = \{\lambda, 1, 2, 1\cdot 1, 1\cdot 2, 2\cdot 1, 1\cdot 2\cdot 1, 1\cdot 2\cdot 2\}$ and for instance $t(\lambda) = \sigma$ and $t(1\cdot 2\cdot 2) = b$. □

Let the level of $x \in \mathrm{dom}(t)$ be $|x|$. Intuitively, the level of a node measures its distance from the root of the tree. Then, we can define the depth of a tree t as $\mathrm{depth}(t) = \max\{|x| : x \in \mathrm{dom}(t)\}$. In the same way, for any tree t, we define the set of subtrees of t [denoted with $\mathrm{Sub}(t)$] as follows:

$$\mathrm{Sub}(a) = \{a\} \text{ for all } a \in V_0$$
$$\mathrm{Sub(t)} = \{t\} \cup \bigcup_{i=1,\ldots,n} \mathrm{Sub(t_i)} \text{ for } t = \sigma(t_1, \ldots, t_n) \ (n > 0)$$

and the natural extension to operate on a set of trees T as $\mathrm{Sub}(T) = \bigcup_{t \in T} \mathrm{Sub}(t)$. For any $k > 0$, we will denote the set of subtrees of t with depth k by $\mathrm{Sub}_k(t)$.

Let \$ be a new nullary symbol not in V_0, and $V_\$^T$ be the set of trees $(V \cup \{\$\})^T$, where each tree contains \$ only once, we will refer to the elements of this set as *contexts* and to the node with label \$ as the *link point*. For any $s \in V_\$^T$ and $t \in V^T \cup V_\$^T$, the operation $s\#t$ is defined by

$$s\#t(x) = s(x) \text{ whenever } x \in \mathrm{dom}(s) \text{ and } s(x) \neq \$$$
$$s\#t(x) = t(z) \text{ where } x = y \cdot z, \quad s(y) = \$$$
$$y \in \mathrm{dom}(s) \text{ and } z \in \mathrm{dom}(t).$$

Therefore, for any two trees t, $s \in V^T$, the tree quotient $(t^{-1}s)$ is defined by

$$t^{-1}s = \{t\} \text{ for every } t \in V_0$$
$$t^{-1}s = \{r \in V_\$^T : s = r\#t\} \text{ otherwise.}$$

The special treatment of trees in V_0 allows us to reduce the complexity of further definitions (mainly the definition of canonical automaton). This quotient can be extended to consider sets of trees $T \subseteq V^T$ as

$$t^{-1}T = \bigcup_{s \in T} t^{-1}s.$$

A *finite deterministic tree automaton* is defined as a system $A = (Q, V, \delta, F)$: where Q is a finite set of states; V is a ranked alphabet, $Q \cap V = \emptyset$; $F \subseteq Q$ is the set of final states and $\delta = \bigcup_{i:V_i \neq \emptyset} \delta_i$ is a set of transitions defined as follows:

$$\delta_n : (V_n \times (Q \cup V_0)^n) \to Q \quad n = 1, \ldots, m$$
$$\delta_0(a) = a \quad \forall\, a \in V_0.$$

From now on we will refer to finite deterministic tree automata simply as *tree automata*. See [25] and [26] for other definitions of tree automata.

The transition functions δ_n are extended to a function $\delta, V^T \to Q \cup V_0$ on trees as follows:

$$\delta(a) = a \text{ for any } a \in V_0$$
$$\delta(t) = \delta_n(\sigma, t_1, \ldots, t_n) \text{ for } t = \sigma(t_1, \ldots, t_n) \ (n > 0).$$

Please note that the symbol δ denotes both the set of transition functions of the automaton and the extension of these functions to operate on trees. Note that the tree automaton A cannot accept any tree of depth zero.

We say that a tree t is accepted by A if $\delta(t) \in F$. Given $q \in Q$, we define $L(q) = \{t : \delta(t) = q\}$, and, in the same way, the language accepted by the automaton as $L(A) = \bigcup_{q \in F} L(q)$. We also will refer to these languages as regular tree languages or regular tree sets.

For every regular tree language T, let its *canonical automaton* be defined as the automaton $A(T) = (Q, V, \delta, F)$, where

$$Q = \{u^{-1}T : u \in \mathrm{Sub}(t), \ t \in T\}$$
$$F = \{t^{-1}T : t \in T\}$$
$$\delta_0(a) = a \text{ for every } a \in V_0$$
$$\delta_n\left(\sigma, u_1^{-1}T, \ldots, u_n^{-1}T\right) = (\sigma(u_1, \ldots, u_n))^{-1}T \text{ for any } \sigma(u_1, \ldots, u_n) \in \mathrm{Sub}(T)$$

where the set of states is proved to be finite [6]. Note that for any tree t, $\delta(t) = t^{-1}T$ and therefore, $t \in L(t^{-1}T)$.

Given a finite set of trees T, let the *subtree automaton* for T be defined as $AB_T = (Q, V, \delta, F)$, where

$$Q = \mathrm{Sub}(T)$$
$$F = T$$
$$\delta_n(\sigma, u_1, \ldots, u_n) = \sigma(u_1, \ldots, u_n) \quad \sigma(u_1, \ldots, u_n) \in Q$$
$$\delta_0(a) = a \quad a \in V_0.$$

For any $k \geq 0$, let the k–root of a tree t be defined as follows:

$$\mathrm{root}_k(t) = \begin{cases} t, & \text{if } \mathrm{depth}(t) < k \\ t' : t'(x) = t(x) & \\ \quad x \in \mathrm{dom}(t) & \\ \quad |x| \leq k, & \text{otherwise.} \end{cases}$$

Intuitively, the k-root of a tree takes into account only those nodes which are at a level which is lower than or equal to the value of k. The extension to operate over a set of trees T is defined as $\text{root}_k(T) = \bigcup_{t\in T} \text{root}_k(t)$.

For any other definition about formal languages, we follow the notation of Hopcroft and Ullman [27].

III. k-Reversible Tree Language

The first attempt to learn reversible languages is due to Angluin [23] who proposes algorithms to learn the classes of zero reversible and k-reversible string languages.

Sakakibara [6] extends the idea of zero reversibility to tree languages, and proposes an inference algorithm to learn this class. He also proves that every context-free string language has a grammar in normal form such that the derivation trees of that grammar are elements of a zero reversible tree language. From this, by considering structural information (i.e., the skeletons of the derivation trees of a context-free grammar in normal form), Sakakibara proves that it is possible to infer the whole context-free class.

Following [6], we call a tree automaton $A = (Q, V, \delta, F)$ *reset free* if there are no two distinct states $p, q \in Q$ such that

$$\delta_n(\sigma, q_1, \ldots, q_{i-1}, p, q_{i+1}, \ldots, q_n) = \delta_n(\sigma, q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_n)$$

for some $n > 0$, $\sigma \in V_n$, $1 \leq i \leq n$ and $q_1, \ldots, q_{i-1}$, $q_{i+1}, \ldots, q_n \in Q$. A tree automaton is *zero reversible* if it is reset-free and has at most, one final state.

Lemma 3.1 (Sakakibara [6]): Let $A = (Q, V, \delta, F)$ be a zero reversible tree automaton, t be a context and $u_1, u_2 \in V^T$. If $t\#u_1 \in L(A)$ and $t\#u_2 \in L(A)$, then $\delta(u_1) = \delta(u_2)$.

Let a tree automaton be defined as *order k reset free*, if, given $p, q \in Q$, where $\text{root}_k(L(p)) \cap \text{root}_k(L(q)) \neq \emptyset$, in the automaton there do not exist two transitions $\delta_n(\sigma, q_1, \ldots, q_{i-1}, p, \ldots, q_n)$ and $\delta_n(\sigma, q_1, \ldots, q_{i-1}, q, \ldots, q_n)$ leading to the same state.

Thus, we say a tree automaton is *k-reversible* if it is order k reset free and for every pair of final states f_1 and f_2 the condition $\text{root}_k(L(f_1)) \cap \text{root}_k(L(f_2)) = \emptyset$ is fulfilled. A tree language T is defined as k-reversible if there exist a k-reversible tree automaton A such that $T = L(A)$. Note that the definition of zero and k-reversible languages are equivalent when $k = 0$.

Lemma 3.2: Let $A = (Q, V, \delta, F)$ be a k-reversible tree automaton, t be a context and $u_1, u_2 \in V^T$. If $t\#u_1 \in L(A)$ and $t\#u_2 \in L(A)$ with $\text{root}_k(u_1) = \text{root}_k(u_2)$, then $\delta(u_1) = \delta(u_2)$.

Proof: Let's suppose that $\delta(u_1) \neq \delta(u_2)$, and let $x \in \text{dom}(t)$ be such that $t(x) = \$$ (the link point of $t\#u_1$ or $t\#u_2$).

In reference to the link point of the trees, there are two possibilities.

- $|x| < k$: This leads to a contradiction, Clearly $\text{root}_k(t\#u_1) = \text{root}_k(t\#u_2)$, and therefore, there would exist two final states f_1, f_2 with $\text{root}_k(L(f_1)) \cap \text{root}_k(L(f_2)) \neq \emptyset$.
- $|x| \geq k$: This also leads to a contradiction. Let t', t'' two contexts such that $t = t'\#t''$. It is easy to see that $\text{root}_k(t''\#u_1) = \text{root}_k(t''\#u_2)$. The automaton fulfills the order k reset free condition because it is k-reversible. Again, two possibilities arise. We can apply the first reasoning of this proof or repeat this reduction. Thus, we conclude that $\delta(t''\#u_1) = \delta(t''\#u_2)$. □

Now, we give a characterization of the k-reversible tree languages which is based on the canonical automaton.

Theorem 3.3: A tree language T is k-reversible if, and only if, its canonical automaton is k-reversible.

Proof: If $A(T) = (Q, V, \delta, F)$ is k-reversible, then the proof is direct. Conversely, let's suppose that there exist a k-reversible automaton $A = (Q^A, V^A, \delta^A, F^A)$ such that $L(A) = T$. Let $t_1, t_2 \in V^T$ be such that, $\text{root}_k(t_1) = \text{root}_k(t_2)$. Two possibilities arise.

- When $t_1, t_2 \in T$, then it is clear that $\delta^A(t_1) = \delta^A(t_2)$, because A is k-reversible.
- When t_1 or t_2 do not belong to T, and
  - there exist $v_1, \ldots, v_n \in V^T$ such that $\delta^A(\sigma(v_1, \ldots, v_{i-1}, t_1, \ldots, v_n)) = p$ and $\delta^A(\sigma(v_1, \ldots, v_{i-1}, t_2, \ldots, v_n)) = p'$;
  - there also exist a context t such that $t\#\sigma(v_1, \ldots, v_{i-1}, t_1, \ldots, v_n)$ and $t\#\sigma(v_1, \ldots, v_{i-1}, t_2, \ldots, v_n)$ both belong to T,

  then by the Lemma 3.2 $\delta^A(t_1) = \delta^A(t_2)$.

The canonical automaton would be k-reversible whenever, for any two transitions such that

$$\delta(\sigma, v_1^{-1}T, \ldots, v_{i-1}^{-1}T, t_1^{-1}T, \ldots, v_n^{-1}T) = \delta(\sigma, v_1^{-1}T, \ldots, v_{i-1}^{-1}T, t_2^{-1}T, \ldots, v_n^{-1}T)$$

and $\text{root}_k(t_1) = \text{root}_k(t_2)$, this imply that $t_1^{-1}T = t_2^{-1}T$. We conclude so, because in both situations, $\delta^A(t_1) = \delta^A(t_2)$, and this implies $t_1^{-1}T = t_2^{-1}T$. □

A characterization of the k-reversible tree languages in terms of regular tree sets is the following:

Theorem 3.4: Let $T \subseteq V^T$ be a regular tree language. T is k-reversible if and only if for any context t and $t_1, t_2 \in V^T$ such that $\text{root}_k(t_1) = \text{root}_k(t_2)$, whenever $t\#t_1$ and $t\#t_2$ belong to T, then $t_1^{-1}T = t_2^{-1}T$.

Proof: If T is k-reversible, then, by applying Theorem 3.3 $A(T)$ is k-reversible. If $t\#t_1$ and $t\#t_2$ belong to T with $\text{root}_k(t_1) = \text{root}_k(t_2)$, in $A(T)$, $\delta(t_1) = \delta(t_2)$, then $t_1^{-1}T = t_2^{-1}T$.

Conversely, it is always possible to construct a canonical automaton for any regular tree language T. If $t\#t_1$ and $t\#t_2$ belong to T, then there exist transitions in $A(T)$ fulfilling the following:

$$\delta(\sigma, u_1^{-1}T, \ldots, u_{i-1}^{-1}T, t_1^{-1}T, \ldots, u_m^{-1}T) = \delta(\sigma, u_1^{-1}T, \ldots, u_{i-1}^{-1}T, t_2^{-1}T, \ldots, u_m^{-1}T)$$

by hypothesis $\text{root}_k(t_1) = \text{root}_k(t_2)$. Thus, if $t_1^{-1}T = t_2^{-1}T$, then $A(T)$ is k-reversible. □

Lemma 3.5: The class of k-reversible tree languages is properly included in the class of $(k+1)$-reversible languages.

Proof: Easy from the k-reversibility definition. Note that given a set of trees, whenever two trees $t_1, t_2 \in T$ fulfill that $\text{root}_{k+1}(t_1) = \text{root}_{k+1}(t_2)$ then they also fulfill that $\text{root}_k(t_1) = \text{root}_k(t_2)$. As an example of the strictness of the

Input:
A set S of trees.
An integer value k.
Output:
A k-reversible tree automaton $A = (Q, V, \delta, F)$ such that $S \subseteq L(A)$.
Method:
/* Initialization . */
Let the initial automaton be $A = AB_S = (Q, V, \delta, F)$
for all $p \in Q$
$C_p = \emptyset$
for all $q \in Q : \ q \neq p$
if $root_k(L(p)) = root_k(L(q))$ **then**
$C_p = C_p \cup \{I(q)\}$ **fi**
end for
end for

Let π_0 be the trivial partition over Q
$i = 0$
for all $q, q' \in F$
if $I(q) \in C_{q'}$ **then**
q'' = Fusion of q, q' in π_{i+1}
$C_{q''} = C_q \cup C_{q'}$
fi
end for

do
$i = i + 1$
if there exist two transitions
such that $\delta(u_1, \ldots, u_{i-1}, q, u_{i+1}, \ldots, u_n) = \delta(u_1, \ldots, u_{i-1}, q', u_{i+1}, \ldots, u_n)$ **and**
$I(q) \in C_{q'}$ **then**
q'' = Fusion of q, q' in π_{i+1}
$C_{q''} = C_q \cup C_{q'}$
fi
if there exist two transitions such that
$\delta(u_1, \ldots, u_n) = q$ **and** $\delta(u_1, \ldots, u_n) = q'$ **then**
q'' = Fusion of q, q' in π_{i+1}
$C_{q''} = C_q \cup C_{q'}$
fi
while $\pi_i \neq \pi_{i+1}$
EndMethod.

Fig. 1. k-reversible tree language inference algorithm. $I(q)$ denotes a state identifier function (see text for details).

inclusion, consider the language accepted by the following automaton:

$$\begin{gathered}\delta(\sigma, a, b) = q_1 \\ \delta(\sigma, q_1, q_1) = q_1 \\ \delta(\sigma, a) = q_2 \\ \delta(\sigma, a, q_3, b) = q_2 \\ \delta(\sigma, q_3, b) = q_0 \in F \\ \delta(\sigma, q_1, b) = q_0 \in F\end{gathered}$$

This language is 1-reversible but it is not 0-reversible. □

IV. Inference of k-Reversible Tree Languages

We propose the algorithm in Fig. 1 for obtaining a k-reversible tree automaton A such that $S \subseteq L(A)$. The proposed algorithm constructs the subtree automaton that accepts the training set, merging those states that do not fulfill the k-reversibility conditions. To do this in an efficient way, for every state of the automaton, the algorithm maintains a table that contains the state identifiers with a common k-root. In the algorithm, $I(q)$ denotes a function that returns the identifier of the state q, and C_i denotes the set of states which share a k-root with the state whose identifier is i.

First, the algorithm merges the final states which share a k-root. Once this step is performed, the algorithm traverses the transitions of the automaton, in order to fulfill the k-reversibility conditions.

Example 4.1: As an example of run, consider $k = 1$ and let the training set be the following:

$$\begin{aligned}S = \Big\{&\sigma(\phi(a, \phi(a, \sigma(a, b), b), b), \sigma(a, \sigma(a, \sigma(a, \omega(a))))), \\ &\sigma(\phi(a, \phi(a, \phi(a, \sigma(a, b), b), b), b), \sigma(a, \sigma(a, \omega(a)))), \\ &\sigma(\phi(a, \phi(a, \sigma(a, b), b), b), \sigma(a, \sigma(a, \omega(a)))), \\ &\sigma(\sigma(a, b), \sigma(a, \omega(a)))\Big\}.\end{aligned}$$

The transitions of the subtree automaton together with the entries of the common k-root table are shown as follows.

$\delta(\sigma, a, b) = q_1$,	$C_1 = \emptyset$
$\delta(\phi, a, q_1, b) = q_2$,	$C_2 = \emptyset$
$\delta(\phi, a, q_2, b) = q_3$,	$C_3 = \{9\}$
$\delta(\omega, a) = q_4$,	$C_4 = \emptyset$
$\delta(\sigma, a, q_4) = q_5$,	$C_5 = \emptyset$
$\delta(\sigma, a, q_5) = q_6$,	$C_6 = \{7\}$
$\delta(\sigma, a, q_6) = q_7$,	$C_7 = \{6\}$
$\delta(\sigma, q_3, q_7) = q_8 \in F$,	$C_8 = \{10, 11, 12\}$
$\delta(\phi, a, q_3, b) = q_9$,	$C_9 = \{3\}$
$\delta(\sigma, q_9, q_6) = q_{10} \in F$,	$C_{10} = \{8, 11, 12\}$
$\delta(\sigma, q_3, q_6) = q_{11} \in F$,	$C_{11} = \{8, 10, 12\}$
$\delta(\sigma, q_1, q_5) = q_{12} \in F$,	$C_{12} = \{8, 10, 11\}$

First, the final states that share a k-root are merged. Therefore, the states q_8, q_{10}, q_{11} and q_{12} are considered to be the same. The modified transitions are marked with an asterisk.

	$\delta(\sigma, a, b) = q_1$,	$C_1 = \emptyset$
	$\delta(\phi, a, q_1, b) = q_2$,	$C_2 = \emptyset$
	$\delta(\phi, a, q_2, b) = q_3$,	$C_3 = \{9\}$
	$\delta(\omega, a) = q_4$,	$C_4 = \emptyset$
	$\delta(\sigma, a, q_4) = q_5$,	$C_5 = \emptyset$
	$\delta(\sigma, a, q_5) = q_6$,	$C_6 = \{7\}$
	$\delta(\sigma, a, q_6) = q_7$,	$C_7 = \{6\}$
$(*)$	$\delta(\sigma, q_3, q_7) = q_8 \in F$,	$C_8 = \{8, 10, 11, 12\}$
	$\delta(\phi, a, q_3, b) = q_9$,	$C_9 = \{3\}$
$(*)$	$\delta(\sigma, q_9, q_6) = q_8 \in F$,	
$(*)$	$\delta(\sigma, q_3, q_6) = q_8 \in F$,	
$(*)$	$\delta(\sigma, q_1, q_5) = q_8 \in F$,	

The algorithm searches for transitions that do not fulfill the k-reversibility conditions; for instance, $\delta(\sigma, q_3, q_7) = q_8$ and

$\delta(\sigma, q_3, q_6) = q_8$, because the states q_6 and q_7 have a common k-root. The modifications of these transitions are also marked.

	$\delta(\sigma, a, b) = q_1$	$C_1 = \emptyset$
	$\delta(\phi, a, q_1, b) = q_2$	$C_2 = \emptyset$
	$\delta(\phi, a, q_2, b) = q_3$	$C_3 = \{9\}$
	$\delta(\omega, a) = q_4$	$C_4 = \emptyset$
	$\delta(\sigma, a, q_4) = q_5$	$C_5 = \emptyset$
$(*)$	$\delta(\sigma, a, q_5) = q_6$	$C_6 = \{6, 7\}$
$(*)$	$\delta(\sigma, a, q_6) = q_6$	
$(*)$	$\delta(\sigma, q_3, q_6) = q_8 \in F$	$C_8 = \{8, 10, 11, 12\}$
	$\delta(\phi, a, q_3, b) = q_9$	$C_9 = \{3\}$
$(*)$	$\delta(\sigma, q_9, q_6) = q_8 \in F$	
	$\delta(\sigma, q_1, q_5) = q_8 \in F$	

The algorithm proceeds in the same way with the states q_3 and q_9

	$\delta(\sigma, a, b) = q_1$,	$C_1 = \emptyset$
	$\delta(\phi, a, q_1, b) = q_2$,	$C_2 = \emptyset$
$(*)$	$\delta(\phi, a, q_2, b) = q_3$,	$C_3 = \{3, 9\}$
	$\delta(\omega, a) = q_4$,	$C_4 = \emptyset$
	$\delta(\sigma, a, q_4) = q_5$,	$C_5 = \{6, 7\}$
	$\delta(\sigma, a, q_5) = q_6$,	$C_6 = \{5, 6, 7\}$
	$\delta(\sigma, a, q_6) = q_6$,	
$(*)$	$\delta(\sigma, q_3, q_6) = q_8 \in F$,	$C_8 = \{8, 10, 11, 12\}$
$(*)$	$\delta(\phi, a, q_3, b) = q_3$	
	$\delta(\sigma, q_1, q_5) = q_8 \in F$,	

Note that the transitions $\delta(\sigma, a, q_5) = q_6$ and $\delta(\sigma, a, q_5) = q_6$ do not lead to the merging of q_5 and q_6, because they do not have a common k-root. As there are no more states to merge, the automaton obtained is shown in the last table. □

Lemma 4.2 proves the polynomial complexity of the algorithm with respect to the training sample size, where the size is considered as the number of subtrees of every tree contained in the set.

Lemma 4.2: The algorithm in Fig. 1 learns the k-reversible tree language class with complexity $\mathcal{O}(n^3)$, where n denotes the size of the training set.

Proof: The proposed algorithm carries out the inference process in three steps. First, the algorithm obtains table C_i by traversing every transition in the subtree automaton. This can be done in $\mathcal{O}(n^3)$ steps.

Once the table is obtained, the algorithm merges the states of the automaton when necessary in order to fulfill the k-reversibility conditions. The traversal of the automaton transitions can be performed in $\mathcal{O}(n^2)$ steps. Therefore, taking into account that it is possible to merge all the states of the subtree automaton [bounded by $\mathcal{O}(n)$], the whole process can be performed in $\mathcal{O}(n^3)$. □

The general scheme to learn any function distinguishable tree language proposed by Fernau [11] is suitable also to learn k-reversible tree languages. Nevertheless, when that scheme is used to learn this class of tree languages, the parameter k has an exponential role in the time complexity of the algorithm.

V. k-Reversible and k-Definite Tree Languages

When dealing with tree languages, it is possible to extend the results from the classical theory of formal languages. For any $k \geq 0$, the relation $\approx_{kd}$ on V^T is defined by the condition that $t_1 \approx_{kd} t_2$ if and only if $\text{root}_k(t_1) = \text{root}_k(t_2)$. Heuter [28] calls a tree language over V *k-definite* if it is the union of some $\approx_{kd}$-classes. A tree language is *definite* if it is k-definite for some k. The relation $\approx_{krd}$ on V^T is similarly defined by the condition that $t_1 \approx_{krd} t_2$ if and only if $\text{Sub}_k(t_1) = \text{Sub}_k(t_2)$. A tree language over V is called *k-reverse definite* if it is the union of some $\approx_{krd}$-classes. In an analogous way, the relation $\approx_{kgd}$ on V^T is defined by the condition that $t_1 \approx_{kgd} t_2$ if and only if $t_1 \approx_{kd} t_2$ and $t_1 \approx_{krd} t_2$. A tree language over V is *k-generalized definite* if it is the union of some $\approx_{kgd}$-classes [29].

In the string language domain, the classes of the k-definite, k-reverse definite and the class of the k-generalized definite languages are defined taking into account the set of prefixes and suffixes of the strings. All this classes are properly included in the class of the k-reversible languages. This relation also remains when tree languages are considered, as we demonstrate in the following theorems.

Theorem 5.1: For any $k \geq 0$, every k-definite tree language is k-reversible.

Proof: Let $t\#t_1, t\#t_2 \in T$ were $t_1, t_2 \in V^T$, t is a context and $\text{root}_k(t_1) = \text{root}_k(t_2)$. It is clear that the membership of $t\#t_1$, $t\#t_2$ to T depend on the k-roots of $t\#t_1$ and $t\#t_2$. Thus whenever $t\#t_1 \in T$ it follows that $t\#t_2 \in T$, and therefore we can conclude that $t_1^{-1}T = t_2^{-1}T$, and by Theorem 3.4, T is k-reversible. □

Theorem 5.2: Any k-reverse definite tree language is k-reversible.

Proof: Let $t\#t_1, t\#t_2 \in T$, where t is a context, $t_1, t_2 \in V^T$ and $\text{root}_k(t_1) = \text{root}_k(t_2)$. To show that $t_1^{-1}T = t_2^{-1}T$ we distinguish two cases:

- if $depth(t_1) = depth(t_2) < k$ then $t_1 = t_2$ and the claim holds trivially;
- if $depth(t_1), depth(t_2) \geq k$ then let t' be a context $t' \neq t$. The claim follows from the fact that, whenever $t'\#t_1 \in T$ it follows that $t'\#t_2 \in T$ because $\text{Sub}_k(t'\#t_2) \subseteq \text{Sub}_k(t'\#t_1) \cup \text{Sub}_k(t\#t_2)$. □

Theorem 5.3: Any k-generalized definite tree language is k-reversible.

Proof: For any k-generalized definite tree language T is both k-definite and k-reverse definite, and therefore, by theorems 5.1 and 5.2, we can conclude that T is k-reversible. □

VI. k-Reversible and k-Testable in the Strict Sense Tree Languages

The *k-testable in the strict sense* (k-TSS) languages are intuitively defined by a set of structures which are allowed to appear in the elements of the language. The first algorithm proposed to learn the class of k-TSS tree languages is due to Knuutila [7]. García [8] improves the algorithm, reducing its time complexity from $\mathcal{O}(n \cdot logn)$ to $\mathcal{O}(n)$, where in both cases n denotes the size of the training set.

For any t over V, we define the k-test vector of t as $v_k(t) = (r_{k-1}(t), l_{k-1}(t), p_k(t))$ where

$$r_{k-1}(t) = \text{root}_{k-1}(t)$$
$$l_{k-1}(t) = \bigcup_{j \leq k-1} \text{Sub}_j$$
$$p_k(t) = \begin{cases} \emptyset, & \text{if depth}(t) \leq k-1 \\ \bigcup_{t' \in \text{Sub}(t)} r_k(t'), & \text{otherwise.} \end{cases}$$

In order to obtain the k-test vector of a tree set T, we define $r_k(T) = \cup_{t \in T} r_k(t)$, $l_k(T) = \cup_{t \in T} l_k(t)$ and $p_k(T) = \cup_{t \in T} p_k(t)$. A tree language T is k-TSS if there exist three tree sets $R \subseteq r_{k-1}(V^T)$, $L \subseteq l_{k-1}(V^T)$ and $P \subseteq p_k(V^T)$, such that for any t over V, $t \in T$ if and only if $r_{k-1}(t) \in R$, $l_{k-1}(t) \subseteq L$ and $p_k(t) \subseteq P$. Another characterization of these languages is the following:

Theorem 6.1: Let $T \subseteq V^T$. T is a k-TSS if and only if, for any trees $t_1, t_2 \in V^T$ such that $\text{root}_{k-1}(t_1) = \text{root}_{k-1}(t_2)$, whenever $t_1^{-1}T \neq \emptyset$ and $t_2^{-1}T \neq \emptyset$, then $t_1^{-1}T = t_2^{-1}T$.

Proof: Let's suppose T to be k-TSS characterized by the sets R, L and P. Let t_1 and t_2 be in V^T with $\text{root}_{k-1}(t_1) = \text{root}_{k-1}(t_2)$ such that $t_1^{-1}T \neq \emptyset$ and $t_2^{-1}T \neq \emptyset$. Looking for a contradiction, let $t_1^{-1}T \neq t_2^{-1}T$ and let t be a context such that $t\#t_1, \in T$ and $t\#t_2 \notin T$. It is easy to see that $r_{k-1}(t\#t_2) = r_{k-1}(t\#t_1) \in R$. Furthermore, $l_{k-1}(t\#t_2) \subseteq L$ due to the fact that $t\#t_1 \in T$ and $t_2^{-1}T \neq \emptyset$. Finally $t\#t_1 \in T$, $t_2^{-1}T \neq \emptyset$ and $\text{root}_{k-1}(t_1) = \text{root}_{k-1}(t_2)$ imply that $p_k(t\#t_2) \subseteq P$. All these facts together lead to a contradiction.

In order to demonstrate the other direction, it is possible to prove that, given a tree language T, when its canonical automaton fulfill the k-reversibility conditions, then it is isomorphic to the automaton obtained by using the k-TSS tree language inference algorithm proposed in [8] (see Appendix). □

The inclusion of the k-TSS string languages into the class of the $(k-1)$-reversible string languages is demonstrated in [2]. Now we prove this result to take into account tree languages.

Theorem 6.2: Let $T \subseteq V^T$. If T is k-TSS, then T is $(k-1)$-reversible.

Proof: Let t_1, $t2$ be trees over V where $\text{root}_k(t_1) = \text{root}_k(t_2)$ and t be a context such that both $t\#t_1$ and $t\#t_2$ belong to T.

Clearly $t_1^{-1}T \neq \emptyset$ and $t_2^{-1}T \neq \emptyset$. If T is a k-TSS tree language, then by Theorem 6.1, $t_1^{-1}T = t_2^{-1}T$, and, therefore, by the Theorem 3.4, T is $(k-1)$-reversible. □

VII. k-Reversible and k-Testable Tree Languages

The k-test vector v_k, defined in Section VI, allows us to define the equivalence relation $\equiv_k$ such that, given two trees, t and u, $t \equiv_k u$ if and only if $v_k(t) = v_k(u)$. From this equivalence relation, we define the *k-testable* tree languages as those that result from the union of a finite number of equivalence classes defined by $\equiv_k$. This extends to tree languages a well-known string languages family (i.e., [18] and [30]).

Example 7.1: Let A be the following automaton:

$$\delta(\sigma, a, b) = q_1$$
$$\delta(\sigma, a, q_1, b) = q_2 \in F$$
$$\delta(\sigma, a, q_2, b) = q_2 \in F$$

$L(A)$ is 1-testable. A accepts all the trees equivalent to the following:

$$\sigma(a, \sigma(a, b), b)$$

in other words, every tree t whose vector $v_1(t)$ is the following:

$$r_0(t) = \{\sigma\}$$
$$l_0(t) = \{a, b\}$$
$$p_1(t) = \{\sigma(a, b), \sigma(a, \sigma, b)\}.$$

If A were 1-TSS then the tree $\sigma(a, b)$ would belong to the language, however, it does not. □

It is easy to see that the class of k-definite tree languages (as well as the k-reverse definite and the k-generalized definite tree languages) is included in the class of k-testable tree languages. However, the classes of k-testable and k-reversible tree languages are not comparable.

Theorem 7.2: The classes of k-testable and k-reversible tree languages are not comparable.

Proof: The 2-testable language obtained with the set of trees equivalent to $\sigma(a, \sigma(a, \sigma(a, b), b), b)$ is accepted by the canonical automaton with transitions.

$$\delta(\sigma, a, b) = q_1$$
$$\delta(\sigma, a, q_1, b) = q_2$$
$$\delta(\sigma, a, q_2, b) = q_3 \in F$$
$$\delta(\sigma, a, q_3, b) = q_3 \in F$$

This language is a counterexample of a k-testable language that is not k-reversible. If it were, the states q_2 and q_3 should be merged, as well as the states q_1 and q_2 and, in this case, the automaton will accept, for instance, the tree $\sigma(a, b)$ that does not belong to the k-testable language.

Conversely, the automaton with transitions:

$$\delta(\sigma, q_1, q_2) = q_0 \in F, \quad \delta(\sigma, q_2, q_2) = q_2$$
$$\delta(\sigma, a, b) = q_1, \quad \delta(\sigma, q_1, q_1) = q_1$$
$$\delta(\sigma, a) = q_2,$$

is k-reversible, but it is not k-testable. For instance, the tree

$$\sigma(\sigma(\sigma(\sigma(a), \sigma(a)), \sigma(\sigma(a), \sigma(a))), \sigma(\sigma(\sigma(a, b)$$
$$\sigma(a, b)), \sigma(\sigma(a, b), \sigma(a, b))))$$

that does not belong to the language should belong to it, as it has the same characteristic tuple as the tree

$$\sigma(\sigma(\sigma(\sigma(a, b), \sigma(a, b)), \sigma(\sigma(a, b), \sigma(a, b))))$$
$$\sigma(\sigma(\sigma(a), \sigma(a)), \sigma(\sigma(a), \sigma(a))))$$

that belongs to the language. □

VIII. Conclusions and Future Work

In this paper, two classes of tree languages are characterized, some properties concerning these classes are proven, and they are also studied in relation to other well-known tree language classes.

The first class of tree languages is obtained by extension of the notion of k-reversibility from string languages to tree languages. We prove that this class contains several classes of tree languages and we propose an algorithm which learns the class in polynomial time complexity with respect to the size of the training sample set. The class of k-reversible tree languages could be seen as a function distinguishable language, and therefore it is possible to use the scheme proposed by Fernau [11], but in this case, with an exponential time complexity. We also define the class of k-testable tree languages as the extension to tree-like structures of the well-known notion in string languages. This class is also studied in relation to other classes.

For future lines of work, the characterization of new tree languages will offer a way to learn new subclasses of context-free string languages. The development of tree language inference algorithms will also allow, in pattern recognition tasks, to use powerful representation primitives to model the different classes of classification problems.

Appendix
Proof of Theorem 6.1

Given a tree set $T \subseteq V^T$ and $u_1, u_2 \in V^T$, in order to demonstrate that whenever

$$\begin{aligned} \text{root}_{k-1}(u_1) &= \text{root}_{k-1}(u_2) \\ u_1^{-1}T \neq \emptyset &\text{ and } u_2^{-1}T \neq \emptyset \end{aligned}$$

implies that $u_1^{-1}T = u_2^{-1}T$ then T is k-TTS, we will take into account the algorithm proposed by Garcia [8], that constructs a k-TTS tree automaton $A = (Q, V, \delta, F)$ where

$$\begin{aligned} Q &= r_{k-1}(T) \cup l_{k-1}(T) \cup p_k(T) \\ F &= r_{k-1}(T) \end{aligned}$$

for all $t \in L$ add to δ the transition $\delta(t) = t$ for all $\sigma(t_1, \ldots, t_n) \in P$ add to δ the transition $\delta(\sigma, t_1, \ldots, t_n) = r_{k-1}(\sigma(t_1, \ldots, t_n))$ and we will prove that the canonical automaton $A(T)$ that fulfills the conditions is isomorphic to A.

Let $A(T) = (Q^A, V, \delta^A, F^A)$ be the canonical automaton of T where

$$\begin{aligned} Q^A &= \{u^{-1}T : u \in \text{Sub}(t),\ t \in T\} \\ F^A &= \{t^{-1}T : t \in T\} \\ \delta_n^A\left(\sigma, u_1^{-1}T, \ldots, u_n^{-1}T\right) &= (\sigma(u_1, \ldots, u_n))^{-1}T \text{ where} \\ \sigma(u_1, \ldots, u_n) &\in \text{Sub}(t) \text{ and } t \in T \\ \delta_0^A(a) &= a \quad a \in V_0. \end{aligned}$$

If the automaton $A(T)$ fulfills the k-TSS conditions, then we can identify the sets $u^{-1}T$ with $r_{k-1}(u^{-1}T)$, that is, we can identify Q^A with Q. In the same way, we can see that F^A is equivalent to F and that finally the set of transitions δ^A is also equivalent to δ.

Acknowledgment

The authors wish to acknowledge the anonymous referee for all the remarks that helped to improve this paper.

References

[1] E. M. Gold, "Language identification in the limit," *Inform. Contr.*, vol. 10, pp. 447–474, 1967.
[2] P. García, E. Vidal, and J. Oncina, "Learning locally testable languages in the strict sense," in *Proc. Workshop Algorithmic Learning Theory*, 1990, pp. 325–328.
[3] J. Ruiz and P. García, "Learning k-piecewise testable languages from positive data," in *Proc. 3rd Int. Colloq.*, vol. 1147, 1996, pp. 203–210.
[4] V. Radhakrisnan and G. Nagaraja, "Inference of regular grammars via skeletons," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, pp. 982–992, 1987.
[5] Y. Sakakibara, "Learning context-free grammars from structural data in polynomial time," *Theor. Comput. Sci.*, vol. 76, pp. 223–242, 1990.
[6] ——, "Efficient learning of context-free grammars from positive structural examples," *Inform. Computat.*, vol. 97, pp. 23–60, 1992.
[7] T. Knuutila, "Chapter inference of k-testable tree languages," in *Proc. Advances Structural Syntactic Pattern Recognition*, 1992, pp. 109–120.
[8] P. García. (1993) Learning k-Testable Tree Sets From Positive Data, Tech. Rep. DSIC/II/46/1993. Dept. Syst. Inform. Comput., Univ. Politécnica Valencia, Valencia, Spain. [Online]. Available: http://www.dsic.upv.es/users/tlcc/tlcc.html
[9] P. García and J. Oncina, "Inference of Recognizable Tree Sets," Dept. Syst. Inform. Comput., Univ. Politécnica Valencia, Valencia, Spain, Tech. Rep. DSIC/II/47/1993, 1993.
[10] E. Mäkinen, "On inferring linear single-tree languages," *Inform. Proc. Lett.*, vol. 73, pp. 1–3, 2000.
[11] H. Fernau, "Learning tree languages from text," in *Proc. 15th Annu. Conf. Computational Learning Theory*, vol. 2375, 2002, pp. 153–168.
[12] D. López and S. España, "Error correcting tree language inference," *Pattern Recognit. Lett.*, vol. 23, no. 1–3, pp. 1–12, 2002.
[13] D. López, J. Ruiz, and P. García, "Chapter Inference of k-piecewise testable tree languages," in *Pattern Recognition and String Matching*. Norwell, MA: Kluwer, 2003.
[14] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[15] R. González and M. Thomason, *Syntactic Pattern Recognition. An Introduction*. Reading, MA: Addison-Wesley, 1978.
[16] E. Vidal, H. Rulot, J. M. Valiente, and G. Andreu, "Application of the error-correcting grammatical inference algorithm (ECGI) to planar shape," in *Grammatical Inference: Theory, Applications and Alternatives*. Essex, U.K.: Institut. Elec. Eng., 1993.
[17] N. Prieto, E. Sanchis, and L. Palmero, "Continuous speech understanding based on automatic learning of acoustic and semantic models," in *Proc. Int. Conf. Spoken Language Processing*, 1994, pp. 2175–2178.
[18] J. Ruiz, "Familia de lenguajes explorables: Inferencia inductiva y caracterización algebraica," Ph.D. dissertation, Dept. Syst. Inform. Comput., Univ. Politécnica Valencia, Valencia, Spain, 1998.
[19] I. Torres and A. Varona, "k-TSS language models in speech recognition systems," *Comput. Speech Lang.*, vol. 15, pp. 127–149, 2001.
[20] S. Y. Lu and K. S. Fu, "Error-correcting tree automata for syntactic pattern recognition," *IEEE Trans. Comput.*, vol. C-27, pp. 1040–1053, 1978.
[21] D. López, J. M. Sempere, and P. García, "Error correcting analysis for tree languages," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 14, no. 3, pp. 357–368, 2000.
[22] D. López and I. Piñaga, "Syntactic pattern recognition by error correcting analysis on tree automata," in *Proc. Joint IAPR Int. Workshops SSPR SPR(S +*, vol. 1876, 2000, pp. 133–142.
[23] D. Angluin, "Inference of reversible languages," *J. ACM*, vol. 29, no. 3, pp. 741–765, 1982.
[24] H. Fernau, "Identification of function distinguishable languages," *Theor. Comput. Sci.*, vol. 290, pp. 1679–1711, 2003.
[25] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. (1997) Tree Automata Techniques and Applications. [Online]. Available: http://www.grappa.univ-lille3.fr/tata
[26] F. Gécseg and M. Steinby, "Chapter tree languages," in *Handbook Formal Languages*. New York: Springer-Verlag, 1997, vol. 3, pp. 1–69.
[27] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[28] U. Heuter, "Definite tree languages," *Bull. EATCS*, vol. 35, pp. 137–142, 1988.
[29] ——, "Generalized definite tree languages," in *Proc. Mathematical Foundations Computer Science*, vol. 379, 1989, pp. 270–280.
[30] J. A. Brzozowski, "Hierarchies of aperiodic languages," *Inform. Theor.*, vol. 10, no. 8, pp. 33–49, 1976.

Damián López received the B.S. and Ph.D. degrees in computer science from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1995 and 2003, respectively.

He joined the Departamento de Sistemas Informáticos y Computación, UPV, in 1996 and currently teaches at the School of Industrial Engineering, UPV. His current fields of interest include multidimensional formal languages and their application in syntactic pattern recognition tasks.

Dr. López is a member of the European Association for Theoretical Computer Science (EATCS).

José M. Sempere received the B.S. and Ph.D. degrees in computer science from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1992 and 2002, respectively.

He joined the Departamento de Sistemas Informáticos y Computación, UPV, in 1989 and currently teaches at the School of Industrial Engineering, UPV. His current fields of interest include computational learning theory, formal languages, and pattern recognition.

Dr. Sempere is a member of the European Association for Theoretical Computer Science (EATCS).

Pedro García received the B.S. degree in physics from the Universidad de Valencia, Valencia, Spain, and the Ph.D. degree in computer science from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1976 and 1988, respectively.

He joined the Departamento de Sistemas Informáticos y Computación, UPV, in 1987 and currently teaches at the School of Industrial Engineering, UPV. His current fields of interest include automata theory and formal languages.