Learning a Subclass of Linear Languages from Positive Structural Information*

José M. Sempere[1] and G. Nagaraja[2]

[1] DSIC, Universidad Politécnica de Valencia, Valencia 46071 (Spain)
email: jsempere@dsic.upv.es
[2] DCSE, Indian Institute of Technology, Powai, Mumbai 400 076 (India)
email: gn@cse.iitb.ernet.in

Abstract. A method to infer a subclass of linear languages from positive structural information (i.e. *skeletons*) is presented. The characterization of the class and the analysis of the time and space complexity of the algorithm is exposed too. The new class, Terminal and Structural Distinguishable Linear Languages (TSDLL), is defined through an algebraic characterization and a pumping lemma. We prove that the proposed algorithm correctly identifies any TSDL language in the limit if structural information is presented. Furthermore, we give a definition of a characteristic structural set for any target grammar. Finally we present the conclusions of the work and some guidelines for future works.

Keywords : Formal languages, grammatical inference, characterizable methods, structural information.

1 Introduction

Linear languages [Ha78] form a language class known to be properly included in the class of context-free languages which properly includes regular languages and even linear languages [AP64]. Within this class, there are some unsolvable problems such as the equivalence problem between linear grammars [Ro72] or the characteristic set problem for grammatical inference [Hi97]. Anyway, some formal language classes which include linear languages or intersect with them have been proposed to be learned from positive strings or positive and negative strings or structural (*skeletons*) information. Specifically, parenthesis linear grammars can be learned by using a reduction to regular languages [Ta88a], even linear languages can be learned by a reduction to regular languages [Ta88b,SG94] and some subclasses of even linear languages can be learned from positive strings [KMT97,Ra87,RN88,Ma96]. Takada has proposed a hierarchy in which some reductions can be performed by using control sets [Ta95] and within this hierarchy, some linear languages and some context-sensitive languages can be learned.

* Part of this work was carried out during a visit of J. Sempere to Prof. G. Nagaraja at IIT, Mumbai. The visit was granted by the Área de Programas Internacionales (API) of the Universidad Politécnica de Valencia

Mäkinen has proposed a method to learn the Szilard language of linear grammars [Ma90].

On the other hand, structural information (*skeletons*) has been used as data source in learning context-free languages. Sakakibara [Sa90,Sa92] has proposed two different methods to learn context-free grammars by using queries and positive information. Mäkinen [Ma92b] has pointed out a different way of taking advantage of structural information in learning context-free grammars by using Szilard languages while Ruiz and García [RG94] have presented some preliminary comparative results in using Sakakibara's algorithm or a method to infer k-testable tree sets proposed by García [Ga93]. Mäkinen [Ma92a] has proposed a grammar class to be learned from structural information (*type invertible grammars*), this work is highly related to Sakakibara's algorithm [Sa90]. Radhakrishnan and Nagaraja [Ra87,RN87] have presented a method to infer a subclass of regular languages, *Terminal Distinguishable Regular Languages* (TDRL), from the structural information induced by positive data. They have proposed a similar method [Ra87,RN88] to infer a subclass of even linear languages, *Terminal Distinguishable Even Linear Languages* (TDELL), from positive strings. Sempere and Fos [SF96] proposed a heuristic technique, based on the previous methods by Radhakrishnan and Nagaraja, to work with positive linear skeletons.

In this paper, we present a characterizable method which infers linear grammars from positive structural information (i.e. linear skeletons). The learned class, *Terminal and Structural Distinguishable Linear Languages* (TSDLL), is characterizable by terminal distinguishability and, which is a necessary condition, by distinguishable subskeletons.

The paper has the following sections : in section 2, we give some basic definitions from formal language theory which help to formalize the results and to understand the proposed inference method. An algebraic definition of TSDL languages together with a pumping lemma is presented in section 3 to define the formal framework in which the method will work. The inference method is proposed and we give some features of the method such as convergence property and complexity analysis. We will present a complete example to understand the method. Finally, we present some conclusions and suggestions for future work related to the results of this paper.

2 Basic definitions and notation

We introduce some basic concepts about formal language theory and formal grammars. Most of them can be found in any introductory book on the subject such as [Ha78].

In what follows, Σ denotes an alphabet and Σ^* the universal language over Σ, that is the set of all possible strings over Σ. Given an alphabet Σ and a string $x \in \Sigma^*$, we denote by $\mid x \mid$ the length of x. λ denotes the empty string with $\mid \lambda \mid = 0$.

$G = (N, \Sigma, P, S)$ is a *context-free* grammar, where N is an alphabet of auxiliary symbols, Σ is an alphabet of terminal symbols with $\Sigma \cap N = \emptyset$, P is

a set of productions where every production is a pair (A, β) with $A \in N$ and $\beta \in \{N \cup \Sigma\}^*$, it can be denoted as $A \to \beta$, and $S \in N$ is the axiom of the grammar.

Given a grammar $G = (N, \Sigma, P, S)$, and two strings $x, y \in \{N \cup \Sigma\}^*$, we can say that x *derives to* y, denoted by $x \overset{*}{\underset{G}{\Rightarrow}} y$, if y can be obtained by applying to x a finite set of productions of P. The language generated by the grammar G is denoted as $L(G)$ and is defined as the set $L(G) = \{x \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} x\}$.

Definition 1. *Let $G = (N, \Sigma, P, S)$ be a grammar. $\forall A \in N$, the set $L(G, A)$ is defined as $\{x \in \Sigma^* \mid A \overset{*}{\underset{G}{\Rightarrow}} x\}$. Obviously, $L(G, S) = L(G)$, and we denote $L(G, A)$ as $L(A)$ whenever G is understood.*

Definition 2. *Let $G = (N, \Sigma, P, S)$ be a context-free grammar. A derivation tree in G is a tree which can be defined by the following rules:*

1. *Every node of the tree can be labeled with a (terminal or auxiliary) symbol of the grammar or with the empty string λ.*
2. *The root of the tree is labeled with S.*
3. *The internal nodes of the tree are always labeled with auxiliary symbols.*
4. *If an internal node is labeled with A and its sons are labeled with symbols $X_1, X_2, \cdots X_n$ then $A \to X_1 X_2 \cdots X_n \in P$.*
5. *If a node is labeled with λ then it is the only son of its father.*

The *result* of a derivation tree is the string obtained by looking over all its leaves from left to right. It is obvious that every string of $L(G)$ admits at least one derivation tree.

A grammar G is said to be *ambiguous* if there exists a string $x \in L(G)$ such that there exist more than one derivation tree with result x. A derivation A-tree is a derivation tree with root label A.

Definition 3. *Let $G = (N, \Sigma, P, S)$ be a context-free grammar. A skeleton in G is a derivation tree in which the internal nodes are not labeled.*

In Figure 1, a grammar together with a derivation tree and the corresponding skeleton are given for a string $x = abbd$.

Definition 4. *Let $G = (N, \Sigma, P, S)$ be a grammar. We will say that G is linear if every production in P takes one of the following forms*

- $A \to uBv$, *where* $A, B \in N$ *and* $u, v \in \Sigma^*$.
- $A \to u$, *where* $A \in N$ *and* $u \in \Sigma^*$.

Every linear grammar $G = (N, \Sigma, P, S)$ admits the following normal form in its productions $A \to aB \mid Ba \mid a \mid \lambda$ where $A, B \in N$ and $a \in \Sigma$

From now on, we will deal with linear grammars in this normal form. Furthermore, we can omit the production $A \to \lambda$, given that it is only necessary in such languages where $\lambda \in L(G)$. In such case we can easily extend the results and formalization that we are going to carry out.

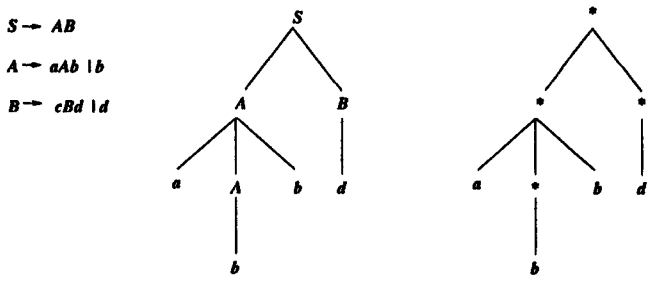


Fig. 1. A context-free grammar together with a derivation tree and the associated skeleton for a string $x = abbd$.

Definition 5. *Let G be a linear grammar in normal form, $x \in L(G)$ and sk a skeleton with result x according to grammar G. Then, for every internal node n of sk the frontier of n can be defined as the result of the subskeleton rooted at n, the head of n can be defined as the left result of the skeleton up to n and the tail of n can be defined as the right result of the skeleton up to n. Given an internal node n, the tuple $(head(n), frontier(n), tail(n))$ denotes its structural information.*

In Figure 2, the *frontier*, *head* and *tail* of a skeleton internal node is shown. It is obvious that, given a skeleton sk and one of its internal nodes n, then the result of the skeleton, denoted by $result(sk)$, is equal to $head(n) \cdot frontier(n) \cdot tail(n)$.

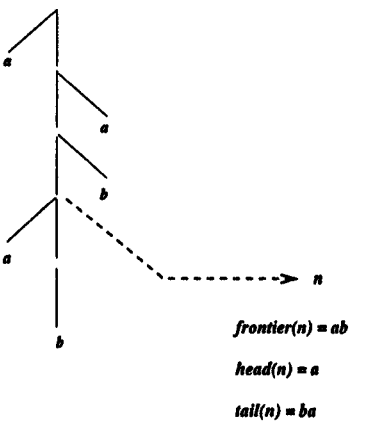


Fig. 2. A skeleton and *head*, *tail* and *frontier* of an internal node.

Definition 6. *Given a linear grammar G in normal form, a skeleton sk according to G and an internal node n of sk, we define the structural head of n,*

*denoted by headstr(n) and the structural tail of n denoted by tailstr(n) as the strings obtained by including in head(n) and tail(n) the symbol * which denotes the absence of left or right son in the ancestral nodes.*

In Figure 2, given the internal node n of the skeleton, then $headstr(n) = a**$ and $tailstr(n) = ba*$.

Definition 7. *Let $w \in \Sigma^*$. We denote by $ter(w)$ the set of symbols of Σ which appear in w. That is, $ter(w) = \{a \in \Sigma \mid \exists w_1, w_2 \in \Sigma^* : w = w_1 \cdot a \cdot w_2\}$. Obviously, $ter(\lambda) = \emptyset$.*

Given a grammar $G = (N, \Sigma, P, S)$ and $A \in N$, we denote by $ter(A) = \bigcup_{w \in L(A)} ter(w)$.

3 Algebraic and structural characterization of TSDL languages

Radhakrishnan and Nagaraja [Ra87,RN87,RN88] have proposed different regular and even linear language classes which can be inferred from positive strings. Furthermore, they have proposed a structural definition for the new classes. In what follows, we propose an extension from their methods to enlarge the class, and we focus on linear languages. One of the most important problems to deal with linear languages is the *ambiguity* problem, that is different skeletons with the same result. In order to obtain an efficient method to recognize similar structures we need to impose a condition which we have named *strong backward determinism* and we define it as follows.

Definition 8. *Let $G = (N, \Sigma, P, S)$ be a linear grammar in normal form. G is said to be a strongly backward deterministic grammar if $\forall w \in \Sigma^*$, $A \overset{*}{\underset{G}{\Rightarrow}} w$ and $B \overset{*}{\underset{G}{\Rightarrow}} w$ implies that $A = B$. Moreover, the A-tree with result w is unique.*

Obviously, every *strongly backward deterministic* grammar is an unambiguous grammar.

Definition 9. *Let $G = (N, \Sigma, P, S)$ be a linear grammar in normal form. G is said to be a Terminal and Structural Distinguishable Linear (TSDL) grammar if the following conditions are fulfilled:*

1. *G is strongly backward deterministic.*
2. *$\forall A, B, C \in N$ such that*

$$(A \to aB \ \wedge \ A \to aC \ \in P) \vee (A \to Ba \ \wedge \ A \to Ca \ \in P)$$

then $ter(B) \neq ter(C)$.

A language L is *Terminal and Structural Distinguishable Linear* (TSDL) if there exists a TSDL grammar G such that $L = L(G)$.

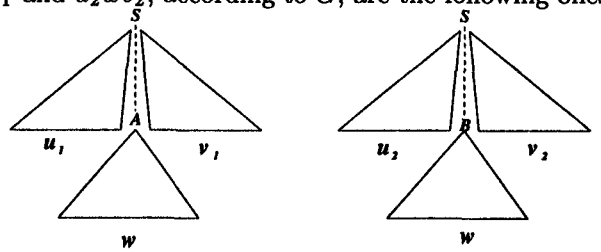
We can give the following pumping lemma, which characterizes TSDL languages, by using some substring properties from the definition that we have given before.

Lemma 1. *Let L be a TSDL language over Σ^*. Then $\forall u_1, v_1, u_2, v_2, w, z \in \Sigma^*$ the following result holds*

$$u_1 w v_1,\ u_2 w v_2 \in L \Rightarrow (u_1 z v_1 \in L \Leftrightarrow u_2 z v_2 \in L).$$

Proof.

Let us suppose that $L = L(G)$ with G a TSDL grammar. The derivation trees for $u_1 w v_1$ and $u_2 w v_2$, according to G, are the following ones



Since G is a TSDL grammar it is strongly backward deterministic, and $A = B$ in the previous derivation trees. Hence, $u_1 z v_1$ and $u_2 z v_2$ are or are not in L depending on the existence of the derivation $A \overset{*}{\underset{G}{\Rightarrow}} z$.

□

Example 1. The following languages are not TSDL languages. Observe that we propose examples of finite, regular, even linear and linear languages which are not TSDL languages and we prove it by applying the previous lemma.

- $\{aab, aaabb, aabb\}$. In this case it is enough to take $u_1 = a$, $v_1 = \lambda$, $u_2 = aa$, $v_2 = b$, $w = ab$ and $z = abb$
- $\{aab\} \cup aaab^*$. By taking $u_1 = a$, $v_1 = \lambda$, $u_2 = aa$, $v_2 = b$, $w = ab$ and $z = abb$, a contradiction appears against the lemma.
- $\{a^{2n+1}b^{2n}\} \cup \{a^{2n}b^{2n+1}\}$. Here we take $u_1 = aa$, $v_1 = bb$, $u_2 = a$, $v_2 = bbb$, $w = aaabb$ and $z = aabbb$.
- $\{a^n b^{2n+1}\} \cup \{a^{n+1}b^{2n}\}$. In this case $u_1 = aa$, $v_1 = bbb$, $u_2 = aaa$, $v_2 = bb$, $w = abbbb$ and $z = aabbb$.

TSDL languages form a class which intersect with finite languages, regular languages, even linear languages and linear languages, but do not properly contain any of them. We have given some examples which prove the non proper inclusion property. In Figure 3 we show the relationships between *TSDL* languages and other formal language classes.

4 Inference method

Once we have defined the *TSDL* language class, we propose an inference method. In the first place, we propose an equivalence relationship between the internal nodes of the skeletons. Later, the proposed algorithm calculates the equivalence classes according to the previous relationship and then induces a *TSDL* grammar from the equivalence classes.

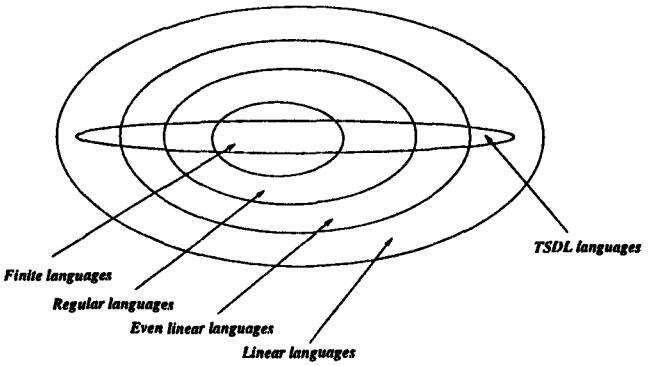


Fig. 3. TSDL languages in relationship with other formal language families.

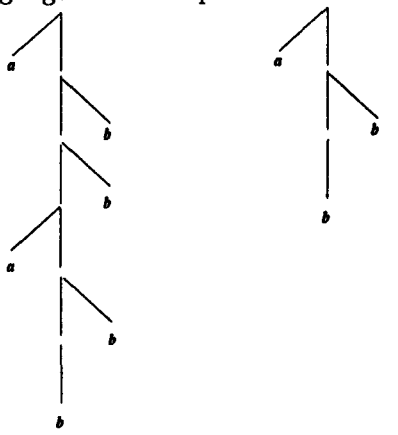
Definition 10. *Let G be a TSDL grammar and SK a nonempty set of skeletons according to G. We can define an equivalence relationship between the internal nodes of the skeletons as follows:*

$n \equiv m$ if and only if one of the following conditions is fulfilled:

1. $frontier(n) = frontier(m)$.
2. $headstr(n) = headstr(m) \wedge tailstr(n) = tailstr(m) \wedge ter(frontier(n)) = ter(frontier(m))$.

In Figure 4 we propose the inference method and we give the following example to apply it

Example 2. Let us take the language $\{a^i b^{2i} : i \geq 1\}$. We provide the following skeletons to the learning algorithm as input data:



In the first step, the algorithm enumerates the internal nodes as follows:

Input : A set of skeletons $SK = \{sk_1, \cdots, sk_n\}$ where every skeleton sk_i is linear in normal form and none contains the label λ.

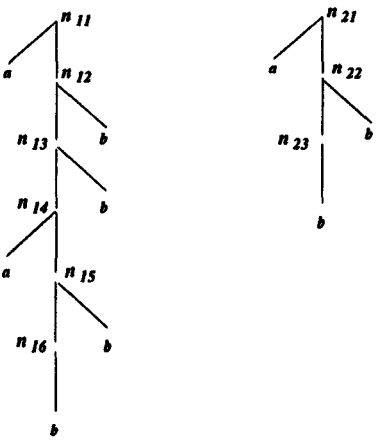
Output : A TSDL grammar G in normal form, where $\forall sk_i \in SK$, $result(sk_i) \in L(G)$.

Method :

(1) Enumerate the internal nodes of every skeleton sk_i. So, internal node n_{ij}. is the jth internal node of ith skeleton from root to leaves.
(2) Calculate the *structural information* of every internal node $(headstr(n_{ij}), frontier(n_{ij}), tailstr(n_{ij}))$.
(3) Calculate the relationship $\equiv$ between internal nodes.
(4) Enumerate every equivalence class $SK_1, SK_2 \cdots SK_l$ with $l \leq n$. Every equivalence class which contains a skeleton root (i.e. sk_{i1}) is labelled with S.
(5) Substitute every internal node by its equivalence class (every skeleton is transformed into a derivation tree).
(6) Obtain a grammar G from the derivation trees.

endMethod

Fig. 4. TSDL languages inference method.



Then, the structural information of every internal node is showed in the next table.

	headstr	*tailstr*	*frontier*	*ter(frontier)*
n_{11}	λ	λ	$aabbbb$	$\{a,b\}$
n_{12}	a	$*$	$abbbb$	$\{a,b\}$
n_{13}	$a*$	$b*$	$abbb$	$\{a,b\}$
n_{14}	$a**$	$bb*$	abb	$\{a,b\}$
n_{15}	$a**a$	$*bb*$	bb	$\{b\}$
n_{16}	$a**a*$	$b*bb*$	b	$\{b\}$
n_{21}	λ	λ	abb	$\{a,b\}$
n_{22}	a	$*$	bb	$\{b\}$
n_{23}	$a*$	$b*$	b	$\{b\}$

The equivalence classes induced by the relationship are

$$SK_1 = \{n_{11}, n_{21}, n_{14}\}$$
$$SK_2 = \{n_{16}, n_{23}\}$$
$$SK_3 = \{n_{22}, n_{15}\}$$
$$SK_4 = \{n_{12}\}$$
$$SK_5 = \{n_{13}\}$$

So, we can make the following replacement in order to label the auxiliary symbols $SK_1 = S, SK_4 = A, SK_5 = B, SK_3 = C$ and $SK_2 = D$. The output grammar is the following one:

$$S \rightarrow aA \mid aC$$
$$A \rightarrow Bb$$
$$B \rightarrow Sb$$
$$C \rightarrow Db$$
$$D \rightarrow b.$$

4.1 Convergence and identification in the limit

Once we have presented the inference algorithm, we are going to prove that it converges to the target grammar if enough information is presented. Here, the information presentation is structural. Under this protocol, we prove that the proposed algorithm identifies any $TSDL$ grammar in the limit [Go64]. The proof is based on the existence of a structural characteristic set for any $TSDL$ grammar in normal form. We give the construction algorithm for such a set by the following lemma

Lemma 2. *Let G be a $TSDL$ grammar and $L = L(G)$. Then there exists a finite set of skeletons according to G such that if it is given as input to the proposed inference algorithm the output is a grammar G' which is isomorphic to G.*

Proof.

Let $G = (N, \Sigma, P, S)$. For every auxiliary symbol $A \in N$, its minimal structural information can be calculated as follows:

- Take the minimal derivation path in which A appears. That is $S \overset{*}{\underset{G}{\Rightarrow}} \alpha A \beta \overset{*}{\underset{G}{\Rightarrow}} w$ with $w \in \Sigma^*$ and it is the shortest derivation path.
- Construct the derivation tree according to the previous derivation.
- Calculate $headstr(A), tailstr(A)$ and $frontier(A)$ as in skeletons.

Since G is TSDL, so it is strongly backward deterministic, then the set of skeletons induced by the derivation trees have the property that if internal nodes n_i and n_j are labeled with different auxiliary symbols, then

1. $frontier(n_i) \neq frontier(n_j)$, and
2. $headstr(n_i) = headstr(n_j) \ \wedge \ tailstr(n_i) = tailstr(n_j) \Rightarrow$
   $\Rightarrow ter(frontier(n_i)) \neq ter(frontier(n_j))$.

If

$$S \underset{G}{\Rightarrow} \alpha_1 A_1 \beta_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_n A_n \beta_n \underset{G}{\Rightarrow} \alpha A \beta \overset{*}{\underset{G}{\Rightarrow}} \alpha w \beta$$

and

$$S \underset{G}{\Rightarrow} \alpha_1 A_1 \beta_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_n A_n \beta_n \underset{G}{\Rightarrow} \alpha B \beta \overset{*}{\underset{G}{\Rightarrow}} \alpha w' \beta$$

are derivations with $A \neq B$, then $ter(w) \neq ter(w')$ and A and B are not related under $\equiv$.

So, the algorithm correctly distinguishes two different internal nodes associated to two different auxiliary symbols.

On the other hand, let us suppose that, in the skeleton set, two different internal nodes, n_i and n_j, are labeled by the same auxiliary symbol. Then one of the following conditions is fulfiled:

1. $frontier(n_i) = frontier(n_j)$. This condition can be true given that we have selected the shortest derivations for every symbol.
2. $\exists n_k : n_k \equiv n_i \ \wedge \ n_k \equiv n_j$. So $n_i \equiv n_j$.

From the previous conditions we can affirm that the algorithm correctly relates those internal nodes induced by one auxiliary symbol.

We can conclude that if the skeleton set we have defined is given as input data then every auxiliary symbol in the grammar is represented and the algorithm correctly distinguishes each of them.

□

From Lemma 2, we can conclude that the algorithm identifies any $TSDL$ grammar in the limit. Here, the proof is trivial given that if every skeleton string is presented then sooner or later the algorithm will work with a characteristic set as input data and, from Lemma 2, it will output the target grammar. Hence, we have the following theorem

Theorem 1. *The proposed algorithm identifies any $TSDL$ language in the limit if structural information is presented.*

4.2 Complexity analysis

Let us suppose that the language to be identified is defined over the alphabet Σ, with $| \Sigma | = n$. The size of an input sample SK is defined by the number of internal nodes that it contains. In linear languages the number of internal nodes of a skeleton is equal to the length of its result. So, we can establish

$$| SK | = \sum_{sk_i \in SK} | frontier(sk_i) | = m.$$

Let k be the size of the maximum length string in the input data.
We can establish the time complexity of the algorithm step by step as follows:

1. **Step 1.** The enumeration of every internal node takes as much time as reading all the skeletons. So the time complexity is $\mathcal{O}(m)$.
2. **Step 2.** As in the previous step, here the algorithm only needs reading all the skeletons once. Observe that we can store the *headstr* and *tailstr* of every internal node by reading the skeleton. Later, the calculation of every *frontier* and its terminals can be established by applying the definition we have given in section 2. In this case the time complexity is $\mathcal{O}(m)$.
3. **Step 3.** The relationship $\equiv$ is established by comparing the structural information of every internal node. The algorithm makes at most $m \times m$ comparisons. For every comparison, the strings *tailstr*, *headstr* and *frontier* have k as maximum length and *ter*(*frontier*) has n as maximum size, so the time complexity is $\mathcal{O}(m^2(k+n))$.
4. **Step 4.** As in step 1, this takes time complexity $\mathcal{O}(m)$ given that the number of equivalence classes is less than or equal to m.
5. **Step 5.** Again, as in step 4, it has time complexity $\mathcal{O}(m)$.
6. **Step 6.** As in steps 1, 4 and 5, the time complexity is $\mathcal{O}(m)$.

The space complexity takes into account the necessary space to storage the structural information of every internal node. Let l be the number of skeletons to be processed, then the necessary space of a skeleton sk is

$$\sum_{i=1}^{|frontier(sk)|} i.$$

This space is $\mathcal{O}(k^2)$. So, the total space is $\mathcal{O}(k^2 l)$.

Theorem 2. *Let SK be a set of skeletons given to the proposed algorithm as input and defined over an alphabet with n symbols. Let m be the size of SK, let k be the size of the maximum length string in SK and let l be the number of skeletons in SK. The proposed algorithm has time complexity $\mathcal{O}(m^2(k+n))$ and space complexity $\mathcal{O}(k^2 l)$.*

5 Conclusions and future work

We have presented a new language class which can be inferred from positive structural information. We think that the proposed method can be applied to other classes and our future work will focus on this. The main advantage of the method is that it doesn't need negative data, this is important when working with non regular languages and structural information, given that the meaning of negative examples can be taken as good strings with bad skeletons or simply bad strings.

The complexity of the method and the method itself makes its implementation relatively easy. It is important for real tasks as picture description languages [RN88] or transduction tasks in which left and right linear productions can be taken as input/output strings.

Aknowledgements

The initial ideas in this paper were formulated during the visit of G. Nagaraja to UPV in July 1997 on the invitation of Prof. Enrique Vidal under a grant from Iberdrola which is gratefully acknowledged. G. Nagaraja is also grateful to IIT Bombay for permission to visit UPV.

J. Sempere is grateful to Dr. Pedro García for helpful discussions about this work.

The authors gratefully acknowledge the suggestions and corrections of the anonymous referee.

References

[AP64] V. AMAR, G. PUTZOLU *On a Family of Linear Grammars.* Information and Control **7** (1964) 283–291.

[Ga93] P. GARCÍA *Learning k-Testable Tree Sets from positive data.* Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. Technical report DSIC-II/46/93. 1993.

[Go64] E. MARK GOLD *Language Identification in the Limit.* Information and Control, **10** (1969) 447–474.

[Ha78] M. HARRISON *Introduction to Formal Language Theory.* Addison-Wesley Publishing Company. 1978.

[Hi97] C. DE LA HIGUERA *Characteristic Sets for Polynomial Grammatical Inference.* Machine Learning **27** (1997) 125–138.

[KMT97] T. KOSHIBA, E. MÄKINEN, Y. TAKADA *Learning deterministic even linear languages from positive examples.* Theoretical Computer Science **185** (1997) 63–97.

[Ma90] E. MÄKINEN *The grammatical inference problem for the Szilard languages of Linear Grammars.* Information Processing Letters **36** (1990) 203–206.

[Ma92a] E. MÄKINEN *On the structural grammatical inference problem for some classes of context-free grammars.* Information Processing Letters **42** (1992) 1–5.

[Ma92b] E. MÄKINEN *Remarks on the structural grammatical inference problem for context-free grammars.* Information Processing Letters **44** (1992) 125–127.

[Ma96] E. MÄKINEN *A note on the grammatical inference problem for even linear languages.* Fundamenta Informaticae **25**, No. 2 (1996) 175–181.

[Ra87] V. RADHAKRISHNAN *Grammatical Inference from Positive Data : An Effective Integrated Approach.* Ph.D. Thesis. Department of Computer Science and Engineering. IIT Bombay. 1987.

[RN87] V. RADHAKRISHNAN AND G. NAGARAJA *Inference of Regular Grammars via Skeletons.* IEEE Trans. on Systems, Man and Cybernetics, **17**, No. 6 (1987) 982–992.

[RN88] V. RADHAKRISHNAN AND G. NAGARAJA *Inference of Even Linear Languages and Its Application to Picture Description Languages.* Pattern Recognition, **21**, No. 1 (1988) 55–62.

[Ro72] G. ROZENBERG *Direct Proofs of the Undecidability of the Equivalence Problem for Sentential Forms of Linear Context-Free Grammars and the Equivalence Problem for 0L Systems.* Information Processing Letters **1** (1972) 233–235.

[RG94] J. RUIZ AND P. GARCÍA *The algorithms RT and k-TTI : A first comparison.* Proceedings of the Second International Colloquium, ICGI-94. LNAI Vol. 862, pp 180-188. 1994.

[Sa90] Y. SAKAKIBARA *Learning context-free grammars from structural data in polynomial time.* Theoretical Computer Science **76** (1990) 223–242.

[Sa92] Y. SAKAKIBARA *Efficient Learning of Context-Free Grammars from Positive Structural Examples.* Information and Computation **97** (1992) 23–60.

[SG94] J.M. SEMPERE AND P. GARCÍA *A Characterization of Even Linear Languages and its Application to the Learning Problem.* Proceedings of the Second International Colloquium, ICGI-94. LNAI Vol. 862, pp 38-44. Springer-Verlag. 1994.

[SF96] JOSÉ M. SEMPERE AND ANTONIO FOS *Learning linear grammars from Structural Information.* Proceedings of the Third International Colloquium, ICGI-96. LNAI Vol. 1147, pp 126-133. Springer-Verlag. 1996.

[Ta88a] Y. TAKADA *Inferring Parenthesis Linear Grammars Based on Control Sets.* Journal of Information Processing, **12**, No. 1 (1988) 27–33.

[Ta88b] Y. TAKADA *Grammatical Inference for Even Linear Languages based on Control Sets.* Information Processing Letters. **28**, No. 4 (1988) 193–199.

[Ta95] Y. TAKADA *A hierarchy of languages families learnable by regular language learning.* Information and Computation, **123(1)** (1995) 138–145.