Learning Linear Grammars from Structural Information *

Jose M. Sempere and Antonio Fos

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Valencia, SPAIN.
email:jsempere@dsic.upv.es

Abstract. Linear language class is a subclass of context-free language class. In this paper, we propose an algorithm to learn linear languages from structural information of their strings. We compare our algorithm with other adapted algorithm from Radhakrishnan and Nagaraja [RN1]. The proposed method and the adapted algorithm are heuristic techniques for the learning tasks, and they are useful when only positive structural data is available.

1 Introduction

In this paper we present a method to infer linear grammars from positive structural examples (*grammar skeletons*). The method that we propose is inspired in a previous work by Radhakrishnan and Nagaraja [RN1]. In their work, Radhakrishnan and Nagaraja proposed an algorithm to infer even linear grammars [AP1] from grammar skeletons under the grammatical inference paradigm [An1]. Learning of even linear grammars has been carried out by other methods from positive and negative strings as in [SG1, Ta1], while learning of context-free grammars has been carried out from skeletons and tree automaton [Ga1, RG1, Sa1]. Learning of linear grammars has not been carried out from string because of the linear grammar ambiguity problem. Anyway, we can apply the methods proposed in [Ga1, Sa1] to learn directly linear languages as context-free grammars. What we propose in this paper is to learn linear languages as linear grammars. So, we can obtain linear time parsers to carry out the test phase in opposite to those obtained in [Ga1, Sa1].

2 Basic definitions and notation

In the first place, we are going to provide several definitions which help us to understand the inference methods. The definitions of formal language theory have been obtained from [HU1, Sa2].

Definition 1. Given a grammar G=(E_A,E_T,P,S), we will say that it is a linear grammar if every production in P follows one of the forms

* Work partially supported by the Spanish CICYT under grant TIC-1026/92-CO2

- $A \to vBw$, where $A, B \in E_A$ and $v, w \in E_T^*$
- $A \to x$, where $A \in E_A$ and $x \in E_T^*$

It is clear that, for every linear grammar, we can obtain an equivalent grammar with its productions in the following forms

- $A \to aB$, where $A, B \in E_A$ and $a \in E_T$
- $A \to Ba$, where $A, B \in E_A$ and $a \in E_T$
- $A \to a$, where $A \in E_A$ and $a \in E_T \cup \{\lambda\}$

From now on, we will deal with linear grammars in the latter form.

Definition 2. Given a grammar G and a string $w \in \mathrm{L}(G)$, we define a *skeleton* for the string w in the grammar G as a derivation tree for the string, where the internal nodes of the tree appear without labels.

In Figure 1 you can see a skeleton for the string *aaabb* of the following grammar

- $S \to aA$
- $A \to Sb \mid aA \mid b$

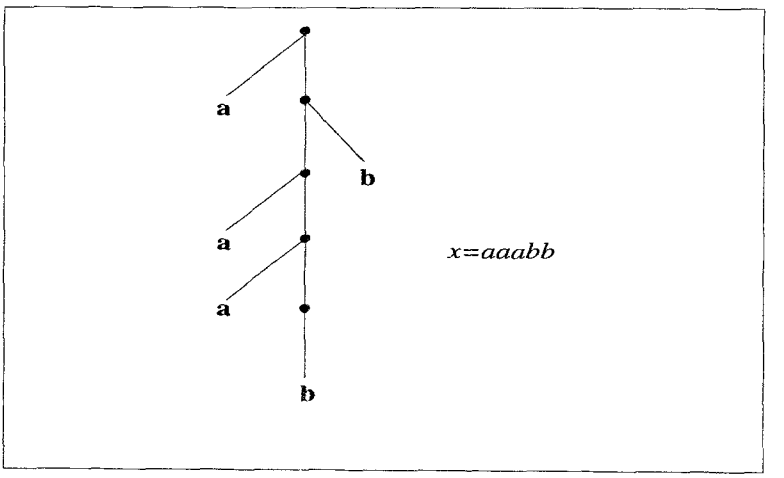

Fig. 1. An example of a skeleton for the string x=*aaabb*.

Definition 3. Given a string w, we denote by *Ter*(w) the set of symbols that appear in the string w.

In what follows, we are going to define several concepts that can distinguish every internal node in the skeleton from the others. So, we suppose that the internal nodes of the skeleton are ordered, and every node is denoted by N_{ij}. For every internal node, we can associate the pair $< x, y >$ or the singleton $< x >$ depending on the number of sons that the node has. If it has two sons then we associate to it the pair, otherwise the singleton. It is obvious that every internal node has two sons or only one. In Figure 2 we can see the different situations that can be held. If the node has a single son, then it is a terminal symbol and we associate to the internal node the singleton $< a >$, where a is the terminal symbol label, otherwise the node has two sons, that is, a terminal symbol and other internal node, and we associate to the node the pair $< N_{ij+1}, a >$ or $< a, N_{ij+1} >$ depending on the location of the terminal symbol label a.

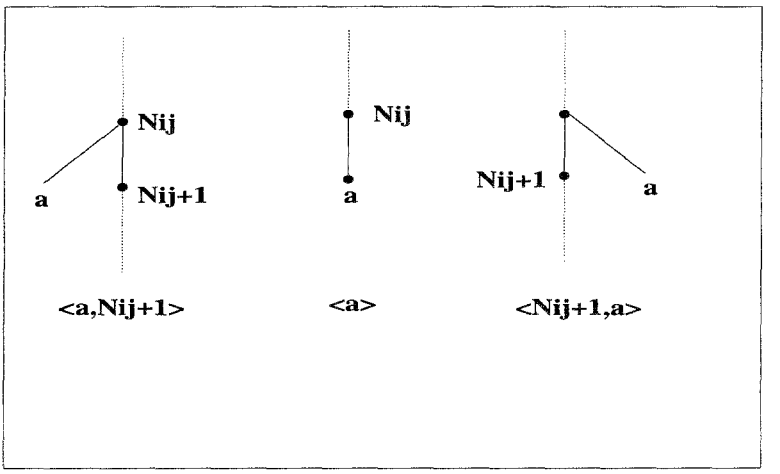

Fig. 2. Different situations for the succesors of an internal node.

Definition 4. Given an internal node N_{ij} of an skeleton for the string x, we define the left substring of the node, and we denote it by $lsubst(N_{ij})$, as the string formed as follows

1. Initially $lsubst(N_{ij}) = \lambda$ (*the empty string*).
2. If N_{ij} has the associated pair $< a, N_{ij+1} >$, then $lsubst(N_{ij})$=$a(lsubst(N_{ij+1}))$.
3. If N_{ij} has the associated singleton $< a >$ or the pair $< N_{ij+1}, a >$ then finish.

Definition 5. Given an internal node N_{ij} of an skeleton for the string x, we define the right substring of the node, and we denote it by $rsubst(N_{ij})$, as the string formed as follows

1. Initially $rsubst(N_{ij}) = \lambda$ (*the empty string*).
2. If N_{ij} has the associated pair $< N_{ij+1}, a >$, then $rsubst(N_{ij})$=$(rsubst(N_{ij+1}))a$.

3. If N_{ij} has the associated singleton $< a >$ or the pair $< a, N_{ij+1} >$ then finish.

We denote by $\mid x \mid$ the length of the string x.

Definition 6. For every internal node N_{ij} of an skeleton for the string x, we define the set of left successors of the node, and we denote it by $lsucc(N_{ij})$ as follows

- Initially $lsucc(N_{ij}) = \oslash$ (*the empty set*).
- If N_{ij} has the associated pair $< a, N_{ij+1} >$ then $lsucc(N_{ij})=\{N_{ij+1}\} \cup lsucc(N_{ij+1})$.
- If N_{ij} has the associated singleton $< a >$ or the pair $< N_{ij+1}, a >$ then finish.

Definition 7. For every internal node N_{ij} of an skeleton for the string x, we define the set of right successors of the node, and we denote it by $rsucc(N_{ij})$ as follows

- Initially $rsucc(N_{ij}) = \oslash$ (*the empty set*).
- If N_{ij} has the associated pair $< N_{ij+1}, a >$ then $rsucc(N_{ij})=\{N_{ij+1}\} \cup rsucc(N_{ij+1})$.
- If N_{ij} has the associated singleton $< a >$ or the pair $< a, N_{ij+1} >$ then finish.

From the definitions above, we can give a more global definition by summarizing the left and the right substring into the context of the string.

Definition 8. Given an internal node N_{ij} of an skeleton for the string x such that $\forall 1 \leq k < j \; N_{ij} \notin lsucc(N_{ik}) \cup rsucc(N_{ik})$, we define the context of the node and we denote it by $context(N_{ij})$ as follows

- If the node has the associated pair $< N_{ij+1}, a >$, then the context is defined by the tuple $context(N_{ij}) =< Right, rsubst(N_{ij}), Ter(rsubst(N_{ij})) >$.
- If the node has the associated pair $< a, N_{ij+1} >$, then the context is defined by the tuple $context(N_{ij}) =< Left, lsubst(N_{ij}), Ter(lsubst(N_{ij})) >$.
- If the node has associated the singleton $< a >$, then the context is defined by the tuple $context(N_{ij}) =< Final, a, \{a\} >$.

Finally, we can define the *projection functions* π_i of a tuple $(x_1, x_2, \ldots, x_n)$ as $\pi_j((x_1, x_2, \ldots, x_n)) = x_j$.

3 An adaptation of a previous algorithm

Our first approach to learn linear grammars has been done by adapting Radhakrishnan and Nagaraja's algorithm [RN1]. The adaptation has been quite easy, given that, from the linear skeletons we can obtain even linear ones, by creating new right or left sons of an internal node. We have labeled these new nodes with the special symbol $*$. In figure 3, we can see an example of the transformation applied to the original skeleton.

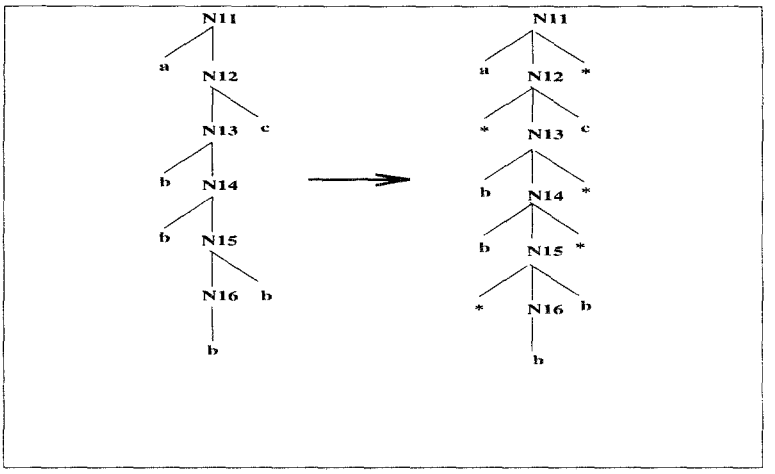

Fig. 3. Skeleton transformation for the adapted algorithm.

From this transformation, the application of the algorithm is made directly. After applying the algorithm, we can obtain an even linear grammar with an special terminal symbol $*$, which can be deleted in order to obtain a linear grammar. Let us see an example of how to apply the learning algorithm.

Taking the following target linear grammar

$S \to aB \qquad B \to Cc \qquad C \to bD$

$D \to bE \mid bC \qquad E \to Fb \qquad F \to b \mid c$

The input sample is the set $\{(a((b(b((b)b)))c)), (a((b(b(b(b((c)b)))))c))\}$, which after be adapted to become even linear strings is $\{a*bb*bb**c*, a*bbbb*cb****c*\}$

Then we can calculate all the sets defined in the algorithm [RN1]

$N_{11} =< \lambda, \lambda, (a,b,c) >$	$NS_1 = \{N_{11}, N_{21}\}$	$S_1 = \{abbbc, abbbbcbc\}$
$N_{12} =< a, \lambda, (b,c) >$	$NS_2 = \{N_{12}, N_{22}\}$	$S_2 = \{bbbc, bbbbcbc\}$
$N_{13} =< a, c, (b) >$	$NS_3 = \{N_{13}\}$	$S_3 = \{bbb\}$
$N_{14} =< ab, c, (b) >$	$NS_4 = \{N_{14}\}$	$S_4 = \{bb\}$
$N_{15} =< ab, bc, (b) >$	$NS_5 = \{N_{15}\}$	$S_5 = \{b\}$
$N_{21} =< \lambda, \lambda, (a,b,c) >$	$NS_6 = \{N_{23}\}$	$S_6 = \{bbbbcb\}$
$N_{22} =< a, \lambda, (b,c) >$	$NS_7 = \{N_{24}\}$	$S_7 = \{bbbcb\}$
$N_{23} =< a, c, (b,c) >$	$NS_8 = \{N_{25}\}$	$S_8 = \{bbcb\}$
$N_{24} =< ab, c, (b,c) >$	$NS_9 = \{N_{26}\}$	$S_9 = \{bcb\}$
$N_{25} =< abb, c, (b,c) >$	$NS_{10} = \{N_{27}\}$	$S_{10} = \{cb\}$
$N_{26} =< abbb, c, (b,c) >$	$NS_{11} = \{N_{28}\}$	$S_{11} = \{c\}$
$N_{27} =< abbbb, c, (b,c) >$		
$N_{28} =< abbbb, bc, (b,c) >$		

After this process, the inferred even linear grammar obtained by the algorithm is the following one

$1 \rightarrow a2*$	$4 \rightarrow *5b$	$7 \rightarrow b8*$	$10 \rightarrow 11b*$
$2 \rightarrow *3c \mid *6c$	$5 \rightarrow b$	$8 \rightarrow b9*$	$11 \rightarrow c$
$3 \rightarrow b4*$	$6 \rightarrow b7*$	$9 \rightarrow b10*$	

and, by deleting the special terminal symbol $*$, we obtain the linear grammar

$1 \rightarrow a2$	$4 \rightarrow 5b$	$7 \rightarrow b8$	$10 \rightarrow 11b$
$2 \rightarrow 3c \mid 6c$	$5 \rightarrow b$	$8 \rightarrow b9$	$11 \rightarrow c$
$3 \rightarrow b4$	$6 \rightarrow b7$	$9 \rightarrow b10$	

4 An algorithm to learn linear grammars

In what follows, we are going to propose another algorithm to obtain linear grammars from positive structural examples. The algorithm is inspired in that proposed by Radhakrishnan and Nagaraja in [RN1], in the sense that we use a similar notation and concepts like in their work. The basic idea is to observe similar context nodes, to label them with the same nonterminal symbol and to construct the grammar from the labeled skeletons or derivation trees.

- **Input** A non empty positive sample of skeletons S^+.
- **Output** A linear grammar that generalize the sample.
- **Method**
  - **STEP 1** To enumerate the internal nodes of every skeleton according to the following notation. For the *j-th* skeleton, to start to enumerate every skeleton by levels from the root to the last level $N_{j1}, N_{j2}, \ldots$
  - **STEP 2** To calculate the context of every node N_{ij} according to definition 8.
  - **STEP 3** To define a relation between nodes $\equiv$ as follows $N_{ij} \equiv N_{pq}$ iff $\pi_1(context(N_{ij})) = \pi_1(context(N_{pq}))$ and $\pi_3(context(N_{ij})) = \pi_3(context(N_{pq}))$ With the defined relation, to form the classes of nodes NS_k by enumerating the classes for $k = 1, 2, \ldots$. The nodes without context do not belong to any class.

  (Creation of nonterminal symbols of the grammar)
  $\forall NS_k$ **do**
  - **STEP 4** If $\forall N_{ij} \in NS_k \ \pi_1(context(N_{ij})) = Final$ then $N_{ij} = A_{k,0}$.
  - **STEP 5** If NS_k only contains a single node N_{ij}, with $\mid \pi_2(context(N_{ij})) \mid = m$ then $N_{ij+p} = A_{k,p} \forall 0 \leq p \leq m-1$.
  - **STEP 6** If NS_k contains more than one node then
    - **STEP 6.1** To select N_{ij} such that $\mid \pi_2(context(N_{ij})) \mid = m$ is minimal. Then $N_{ij+p} = A_{k,p} \forall 0 \leq p \leq m-1$.
    - **STEP 6.2** To eliminate the node N_{ij} of STEP 6.1 from the set NS_k. If NS_k is a singleton then go to STEP 6.3, else go to STEP 6.1.
    - **STEP 6.3** Take the only node N_{ij} of the set NS_k with $\mid \pi_2(context(N_{ij})) \mid = n$ and take the value m of STEP 6.1. If $n \leq m$ then $N_{ij+p} = A_{k,p} \forall 0 \leq p \leq n-1$, else $N_{ij+p} = A_{k,(p \| m)} \ \forall 0 \leq p \leq n-1$ (where $p \| m$ denotes p *module* m).
  - **STEP 7** To rename the labels of all the skeleton roots as S, which will be the axiom of the grammar.
  - **STEP 8** To build a linear grammar as result of the derivation trees constructed by putting labels to the nodes. If the skeleton for the empty string belongs to S^+ then to add the production $S \rightarrow \lambda$ to the set P.

An example.

Taking the following target linear grammar

$S \to aB \qquad B \to Cc \qquad C \to bD$

$D \to bE \mid bC \qquad E \to Fb \qquad F \to b \mid c$

The input sample is the set $\{(a((b(b((b)b)))c)), (a((b(b(b(b((c)b)))))c))\}$ of figure 4.

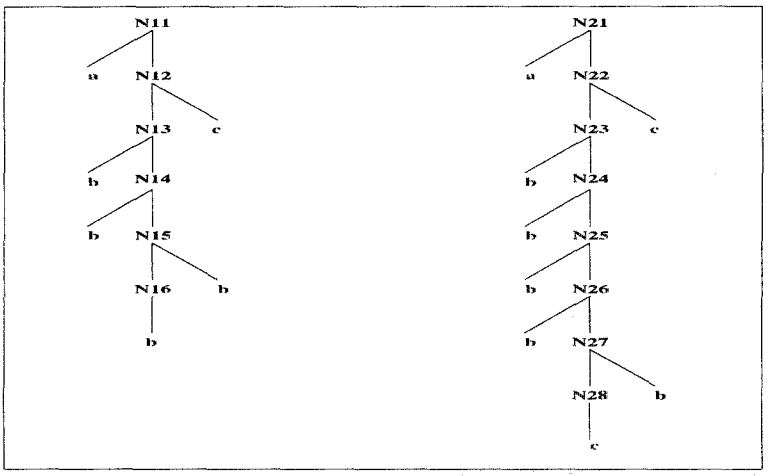

Fig. 4. Input sample for the proposed algorithm.

We can calculate the contexts of every node in the following way

$N_{11} =< Left, a, \{a\} >$	$N_{22} =< Right, c, \{c\} >$	$NS_1 = \{N_{11}, N_{21}\}$
$N_{12} =< Right, c, \{c\} >$	$N_{23} =< Left, bbbb, \{b\}) >$	$NS_2 = \{N_{12}, N_{22}\}$
$N_{13} =< Left, bb, \{b\} >$	$N_{27} =< Right, b, \{b\} >$	$NS_3 = \{N_{13}, N_{23}\}$
$N_{15} =< Right, b, \{b\} >$	$N_{28} =< Final, c, \{c\} >$	$NS_4 = \{N_{15}, N_{27}\}$
$N_{16} =< Final, b, \{b\} >$		$NS_5 = \{N_{16}\}$
$N_{21} =< Left, a, \{a\} >$		$NS_6 = \{N_{28}\}$

After this process, the inferred linear grammar obtained by the algorithm is the following one

$S \to aA_{2,0} \qquad A_{3,1} \to bA_{4,0} \mid bA_{3,0} \qquad A_{6,0} \to c$

$A_{2,0} \to A_{3,0}c \qquad A_{4,0} \to A_{5,0}b \mid A_{6,0}b$

$A_{3,0} \to bA_{3,1} \qquad A_{5,0} \to b$

5 Acknowledgements

We would like to thank Professor G. Nagaraja's interest and all the mail that we have interchanged about this work. We would like to thank Dr. Pedro García's

original contribution to the transformation of linear skeletons to even linear skeletons.

References

[AP1] Amar, V., Putzolu, G.: On a Family of Linear Grammars. Information and Control **7** (1964) 283-291.

[An1] Angluin, D., Smith, C.: Inductive Inference : Theory and Methods. Computing Surveys **15** No. 3 (1983) 237-269.

[Ga1] García, P.: Learning K-Testable Tree Sets from positive data. Technical Report DSIC-II/46/93. Universidad Politécnica de Valencia. (1993)

[HU1] Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley Publishing Company. (1979)

[RN1] Radhakrishnan, V., Nagaraja, G.: Inference of Even Linear Grammars and its Application to Picture Description Languages. Pattern Recognition **21** No. 1 (1988) 55-62.

[RG1] Ruiz, J., García, P.: The Algorithms RT and k-TTI : A First Comparison. Lecture Notes in Artificial Intelligence. Proceedings of the Second International Colloquium on Grammatical Inference ICGI94. Ed. Springer-Verlag. (1994) 180-188.

[Sa1] Sakakibara, Y.: Efficient Learning of Context-Free Grammars from Positive Structural Examples. Information and Computation **97** (1992) 23-60.

[Sa2] Salomaa, A.: Formal Languages. Academic Press. (1973)

[SG1] Sempere, J., P. García, P.: A Characterization of Even Linear Languages and its Application to the Learning Problem. Lecture Notes in Artificial Intelligence. Proceedings of the Second International Colloquium on Grammatical Inference ICGI94. Ed. Springer-Verlag. (1994) 38-44.

[Ta1] Takada, Y.: Grammatical Inference of Even Linear Languages based on Control Sets. Information Processing Letters **28** No.4 (1988) 193-199.