J. Indian Inst. Sci., 64 (B), Nov. 1983, Pp. 277–289
© Indian Institute of Science, Printed in India.

Tree adjunct kolam array languages

K. RANGARAJAN AND K. G. SUBRAMANIAN
Department of Mathematics, Madras Christian College, Tambaram, Madras 600 059, India.

Received on January 1, 1983, Revised on August 22, 1983.

Abstract

Tree Adjunct Kolam Array Grammars (TAKAG) are proposed to generate rectangular arrays of terminal symbols. These grammars have two phases of derivations as in kolam array grammars of Siromoney *et al.* The notion of tree adjunction is made use of in their first phase of derivations. The family of Tree Adjunct Kolam Array Languages (TAKAL) is shown to properly include the family of context-free kolam array languages. The family of TAKAL's is closed under the operations of union and array catenations and star. The families of linear and regular kolam array languages are shown to be proper subfamilies of the families of linear and one-sided linear TAKAL's.

Key words: Chomsky grammars, adjunct languages, array languages.

1. Introduction

There has been considerable interest in recent years in adapting the techniques and in utilising and extending the existing results of formal language theory, for developing methods to study the problem of picture generation and description. Pioneering work in suggesting and applying a linguistic model for the solution of problems in picture processing has been done by Narasimhan[3]. Rosenfeld[4] has extensively investigated array grammars, whose rewriting rules allow the replacement of a subarray of a picture with another subarray.

Siromoney *et al*[6] considered the two-dimensional analogs of strings to be rectangular arrays and extended the notion of catenation of strings to row and column catenations of rectangular arrays. Siromoney *et al*[6] proposed a two-dimensional generative model, called the Matrix grammar, to describe digitized rectangular arrays. This model is capable of generating a wide variety of interesting classes of pictures but it cannot

generate pictures which maintain a fixed proportion between the horizontal and the vertical.

Siromoney *et al*[7] introduced array models, which provide a more powerful approach to defining two-dimensional grammars for rectangular array languages, necessitated by the need to generate picture classes that cannot be generated by the two-dimensional matrix models. The definitions of these array models were motivated by the patterns found in kolam design, a folk art of India and hence we call these models as kolam array grammars, in order to distinguish them from the array grammars of Rosenfeld[4]. These models involve two phases of derivations. The first phase consists of a finite set of horizontal or vertical CS, CF or regular rules which involve only the nonterminals and intermediates. The intermediates act as terminals of the first phase. During the first phase of derivations, derivations proceed making use of the rules, introducing parentheses at every stage to avoid ambiguity due to lack of associativity of the column and row catenation operators. The resultant of the first phase will consist of strings of intermediates catenated together with row and column catenation operators and with parentheses suitably introduced. The second phase consists of rules which generate languages, called intermediate array languages—one corresponding to each intermediate, with the intermediate as the start symbol. The intermediate array languages may be CS, CF or regular over a finite number of arrays all of which consist of a fixed number of rows or a fixed number of columns of elements from the set of terminal symbols. Instead of enumerating the rules, the intermediate array languages are usually given. During the second phase of derivations, starting from the innermost parentheses, each intermediate is replaced by the corresponding intermediate array language subject to the conditions of row and column catenations. When all the intermediates are replaced, we arrive at the rectangular arrays of terminals.

Joshi *et al*[1] introduced a new class of formal grammars, called string adjunct grammars, as an alternate means of describing the generation of formal languages. The rules of these grammars have a different formal character than the usual rewrite rules of phrase structure grammars of Chomsky[5]. The only operation allowed on strings in an adjunct grammar is the adjoining of a string to the left or right of a distinguished symbol in another string. Joshi *et al*[2] have extended the notion of adjunction in strings to trees and have studied tree adjunct grammars. In a tree adjunct grammar, each intermediate tree in a derivation is a sentential tree, *i.e.*, the derivations proceed from a structured sentence to another structured sentence. Thus, a tree adjunct grammar is a grammar of structural descriptions.

In formal language theory, it has been of interest to obtain and study new families of languages. In this paper, we incorporate the notion of tree adjunction in the first phase of derivation of a kolam array grammar[7,9] and introduce Tree Adjunct Kolam Array grammars, generating rectangular arrays of terminal symbols. We compare the family of Tree Adjunct Kolam Array Languages (TAKAL) with other families of kolam array languages. We prove that the family of TAKAL's includes properly the

family of context-free KAL's. We also define linear and one-sided linear TAKAL's and prove that they properly contain the families of linear and regular KAL's respectively. We also establish closure of the family of TAKAL's under union and the array operations of column and row catenations and column and row star.

2. Tree Adjunct Kolam Array Grammars

In this section, we present the necessary definitions and introduce the Tree Adjunct Kolam Array Grammars (TAKAG). The reader is referred to Salomaa[5] for standard notions in formal language theory, to Siromoney *et al*[6,7,9] for details regarding arrays and kolam array grammars and to Joshi *et al*[2] for the concepts of trees and adjoining of trees.

Notation: Let $\mathcal{N}^*$ be the free monoid generated by the set $\mathcal{N}$ of all natural numbers, with the binary operation . and identity 0. For $p, q \in \mathcal{N}^*$, $p \leqslant q$ iff there is an $r \in \mathcal{N}^*$ such that $q = p.r$ and $p < q$ iff $p \leqslant q$ and $p \neq q$.

We first informally describe the notion of a generalized tree.

A generalized tree is a tree whose leaf nodes are labelled with elements of a nonempty set $I \subseteq V$ and interior nodes with elements of $V - I$, where V is a finite nonempty set of symbols. It is a rooted tree with the root at the 'top' node. The descendants of any node have a specific order. All the branches at a node, *i.e.*, branches from a node to its immediate successors are either 'horizontal' or 'vertical' with reference to two fixed horizontal and vertical planes.

Definition 2.1 : Let V be a finite set of symbols and I be a nonempty subset of V. A generalized tree t over V is a function from D_t into V, where the domain D_t is a finite subset of $\mathcal{N}^*$ such that

(i) if $q \in D_t$, $p < q$, then $p \in D_t$

(ii) if $p.j \in D_t$ for $j \in \mathcal{N}$ then $p.1, p.2, \ldots, p.(j-1) \in D_t$

(iii) If q and $p \in D_t$ with $p = q.j$ and $q.(j+1) \notin D_t$ where $j \in \mathcal{N}$, then all the ordered pairs $(q, q.i)$, $i = 1, 2 \ldots, j$ are in P_h or P_v, where P_h and P_v are finite subsets of $D_t \times D_t$. We say that the branches at the node q join the node q with its descendants, $q.i\,(i = 1, \ldots, j)$ and that the branches are horizontal (respy. vertical) if $(q, q.i)$, $(i = 1, \ldots, j)$ are in P_h (respy. P_v).

The elements of D_t are called addresses of t. If $(p, X) \in t$, then X is called the label of the node at the address p in t. We write $t(p) = X$.

A node q in t is (i) a leaf node if for all nodes p of t, we have $q \not< p$ (ii) an interior node if q is not a leaf node. A node whose address is 0 is called the root node.

We give an example to illustrate the notion of a generalized tree.

Example 2.1 : Let $V = \{S, X, A, B, C\}$, $I = \{A, B, C\}$.

Figure 1 shows a generalized tree over V. The address and label of each node are given by the first and second components of the pair of elements marked at each node in fig. 1. For instance, the node marked (O, S) has address O and label S. The fixed horizontal and vertical planes are XOZ and XOY. The branches at node O are in P_h, *i.e.*, horizontal and at 1, 1.2 are in P_v, *i.e.*, vertical.

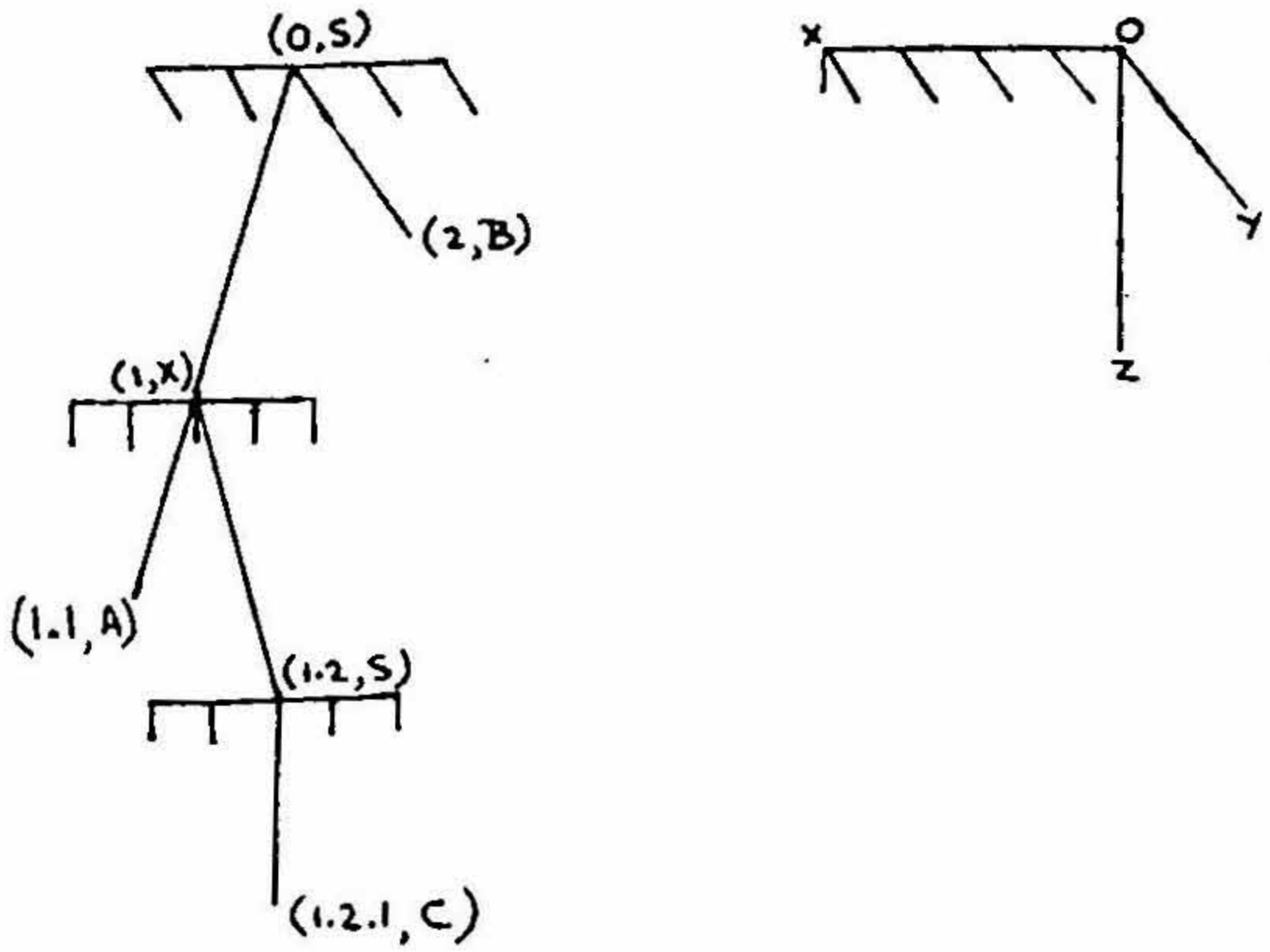


FIG. 1. A generalized tree.

Given a generalized tree t over V, the notions of (i) the subtree t/p at node p, (ii) the super-tree $t \setminus p$ at node p, (iii) $p.t$, (iv) the front $\hat{t}$ of t, (v) a path of t, can be defined for t, as done in the case of a tree[2].

Notation : Let V be a finite set of symbols. A horizontal string or word over V is of the form $w_1 = A_1 A_2 \ldots A_k$ and a vertical word is of the form $w_2 = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{matrix}$, $A_i \in V$, for $i = 1, \ldots, k$. We write w_1 as $(A_1 \textcircled{1} A_2 \textcircled{1} \ldots \textcircled{1} A_k)$ and w_2 as $(A_1 \theta A_2 \theta \ldots \theta A_k)$. We use the symbols $\textcircled{1}$, θ respy. to stand for horizontal and vertical catenations of letters of V. We use the symbol $\oplus$ to stand for either $\textcircled{1}$ *or* θ. We also write a horizontal word $AA \ldots A$ (n letters) as A^n or $(A)^n$, and a vertical word $\begin{matrix} A \\ \vdots \\ A \end{matrix}$ (n letters) as $(A)_n$.

Definition 2.2 : Let V be a finite set of symbols. We define the set $V_{\pm}^{+}$ recursively as follows :

(i) For $A_i \in V$, $i = 1, \ldots, k$, $(A_1 \textcircled{1} A_2 \textcircled{1} \ldots \textcircled{1} A_k)$ and $(A_1 \theta A_2 \theta \ldots \theta A_k)$ belong to $V_{\pm}^{+}$, for $k \geqslant 1$.

(ii) If u and v are in $V_{\pm}^{+}$, then $(u \oplus v) \in V_{\pm}^{+}$.

Definition 2.3 : Let T_v be the set of all generalized trees over an alphabet V. The yield f is a function from T_v into $V_{\pm}^{+} U \{\epsilon\}$ (ϵ is the empty word), defined as follows :

$f(t) = t(0)$, if $D_t = \{0\}$, $t \in T_v$;

$f(t) = t(1)$, if $D_t = \{0, 1\}$, $t \in T_v$;

$f(t) = (f(t/1)) \oplus \ldots \oplus (f(t/j))$, if $1, \ldots, j \in D_t$ and $j + 1 \notin D_t$, for $t \in T$, and some $j \in \mathcal{N}$ and $\oplus$ is the column catenation operator $\textcircled{1}$ if the ordered pairs $(0, 1)$, $\ldots, (0, j)$ with $0, 1, \ldots j \in D_t$, are in P_h and $\oplus$ is the row catenation operator θ if they are in P_v. In other words, $f(t)$ is the string of labels of the leaf node of t, connected by suitable column and row catenation operators $\textcircled{1}, \theta$ with parentheses introduced wherever necessary as $\oplus$ is not associative. We call $f(t)$ as the yield of the generalized tree t.

For instance, we note that in the case of the generalized tree t in fig. 1, the yield $f(t) = (A \theta C) \textcircled{1} B$.

We now define a rectangular array over an alphabet V and the operations of column and row catenations of arrays.

Definition 2.4 : An array M over an alphabet V is of the form

$$M = \begin{matrix} a_{11} \ldots a_{1n} \\ \ldots \\ \ldots \\ a_{m1} \ldots a_{mn} \end{matrix} \qquad a_{ij} \in V \text{ for } 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n \quad (m, n \geqslant 1)$$

$$\text{Let } M_1 = \begin{matrix} b_{11} \ldots b_{1q} \\ \ldots \\ \ldots \\ b_{p1} \ldots b_{pq} \end{matrix} \quad \text{and } M_2 = \begin{matrix} c_{11} \ldots c_{1s} \\ \ldots \\ \ldots \\ c_{r1} \ldots c_{rs} \end{matrix}$$

where b_{ij} $(1 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant q)$ and c_{ij} $(1 \leqslant i \leqslant r,\ 1 \leqslant j \leqslant s)$ $(p, q, r, s, \geqslant 1)$ are in V, be two given arrays. The column catenation of M_1 with M_2 is defined when $p = r$ and is given by

$$M_1 \textcircled{1} M_2 = \begin{matrix} b_{11} \ldots b_{1q}\ c_{11} & \ldots c_{1s} \\ \ldots & \ldots \\ \ldots & \ldots \\ b_{p1} \ldots b_{pq}\ c_{r1} & \ldots c_{rs} \end{matrix} \quad \text{and}$$

the row catenation of M_1 with M_2 is defined when $q = s$ and is given by

$$M_1 \theta M_2 = \begin{matrix} b_{11} \ldots b_{1q} \\ \ldots \\ \ldots \\ b_{p1} \ldots b_{pq} \\ c_{11} \ldots c_{1s} \\ \ldots \\ \ldots \\ c_{r1} \ldots c_{rs} \end{matrix}$$

We now introduce the Tree Adjunct Kolam Array model.

Definition 2.5 : A Tree Adjunct Kolam Array Grammar (TAKAG) is $G = (V, I, \Sigma, \mathcal{C}, \mathcal{A}, \mathcal{L})$ where V and Σ are finite nonempty sets of symbols ; $V \cap \Sigma = \phi$; I is a nonempty subset of V. The elements of $V - I$, I and Σ are called nonterminals, intermediates and terminals respectively. $\mathcal{C}$ and $\mathcal{A}$ are finite subsets of T_θ satisfying the following conditions :

(i) If $t_c \in \mathcal{C}$, then $f(t_c) \in I_{\ddagger}^{\ddagger} \cup \{\epsilon\}$ and $t_c(0) = S$, where S is a distinguished symbol of $V - I$.

(ii) if $t_a \in \mathcal{A}$ and $t_a(0) = X$, then $X \in V - I$ and $f(t_a) \in (I_{\ddagger}^{\ddagger} \cup \{\epsilon\}) \oplus (X) \oplus I_{\ddagger}^{\ddagger}$ or $I_{\ddagger}^{\ddagger} \oplus (X) \oplus (I_{\ddagger}^{\ddagger} \cup \{\epsilon\})$.

$\mathcal{C}$ is called the set of generalized center trees ; $\mathcal{A}$, the set of generalized adjunction trees and the elements of $\mathcal{C} \cup \mathcal{A}$ are called the generalized basic trees of G.

$\mathcal{L} = \{L_A / A \in I\}$, where L_A is a regular, CF or CS intermediate array language, generated by A, over a finite number of arrays over Σ, each of which has either a fixed number of rows or a fixed number of columns. In other words, the rules of the grammar generating the array language L_A are like the rules of a Chomskian string grammar except that the terminal symbols can be a finite number of arrays with the same number of rows or the same number of columns.

In particular, a TAKAG G is a (TA : R) KAG, (TA : CF) KAG or (TA : CS) KAG, according as (i) all the intermediate array languages are regular, (ii) at least one of them is CF but none of them is CS, (iii) at least one of them is CS.

Definition 2.6 : Let t_a be a generalized adjunction tree. Let $t \in T_\theta$ with $p \in D_t$ and $t(p) = t_a(0)$. Then t_a is adjoinable to t at p and $t[p, t_a]$ is the generalized tree obtained from t, by adjoining t_a at p, *i.e.*, the generalized tree

$$t[p, t_a] = t \setminus p \cup p.t_a \cup (p.r).\ (t/p) \text{ where } r \in D_{t_a},$$

$t_a(r) = t_a(0)$ and $(r, t_a(r)) \in t_a$, *i.e.*, r is the address of that node which is in the front of t_a and which has label $t_a(0)$. This operation is called adjunction. The branches of the node $p \in t[p, t_a]$ leading to its successors are either horizontal or vertical according as the branches of the node $0 \in t_a$ leading to its successors are either horizontal or vertical,

We note that if t_a is adjoinable to t at p, then $t\,[p, t_a](p) = t(p) = t_a(0)$ and so t_a is again adjoinable to $t\,[p, t_a]$ at p. We write $t\,[p, t_a]^n$ for the tree obtained by adjoining t_a, n times, starting with t.

We now describe derivations in a TAKAG. In the first phase of derivations, given a generalized tree t, we say that t derives a generalized tree t' and we write $t \Rightarrow t'$ iff t' is obtained from t by adjunction such that $t' = t\,[p, t_a]$, for $p \in D_t$ and for some t_a adjoinable to t at p. $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$. In the first phase of derivations, we obtain generalized trees t from the generalized center trees t_c by the operation of adjunction using the generalized adjunction trees t_a. The tree set of G obtained in the first phase is $T(G) = \{t \in T_\sigma / \text{for some } t_c \in \mathcal{C},\ t_c \Rightarrow^* t\}$.

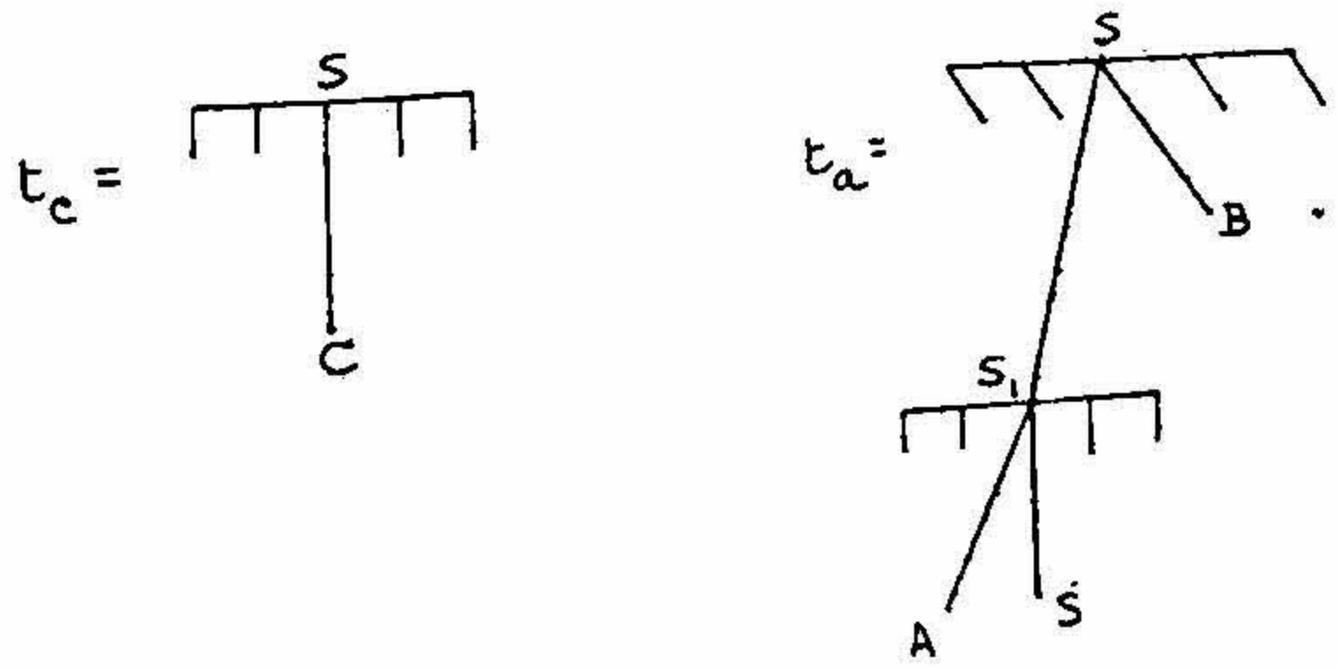
In the second phase of derivations, we consider only those generalized trees $t \in T(G)$ obtained in the first phase whose yields are words over intermediates connected by ⊕, θ symbols and with parentheses suitably introduced. In the second phase of derivations, an array M is said to be derived from $f(t)$, for $t \in T(G)$, if M is obtained by replacing each intermediate A in $f(t)$ by elements of L_A, subject to the conditions imposed by the row and column catenation operators, the replacements starting from the innermost parentheses and proceeding outwards. The replacements come to a dead end if the conditions for row or column catenation are not satisfied.

The Tree Adjunct Kolam Array Language (TAKAL) is $L(G) = \{M \in \Sigma^{**} / M$ is derived from some $f(t),\ t \in T(G)\}$.

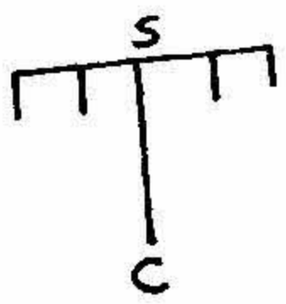
In particular, a TAKAL is a (TA : R) KAL, (TA : CF) KAL or (TA : CS) KAL, if the TAKAG G generating it is a (TA : R) KAG, (TA : CF) KAG or (TA : CS) KAG. We denote the family of (TA : X) KAL's by (TA : X) KAL itself, for $X \in \{R, CF, CS\}$.

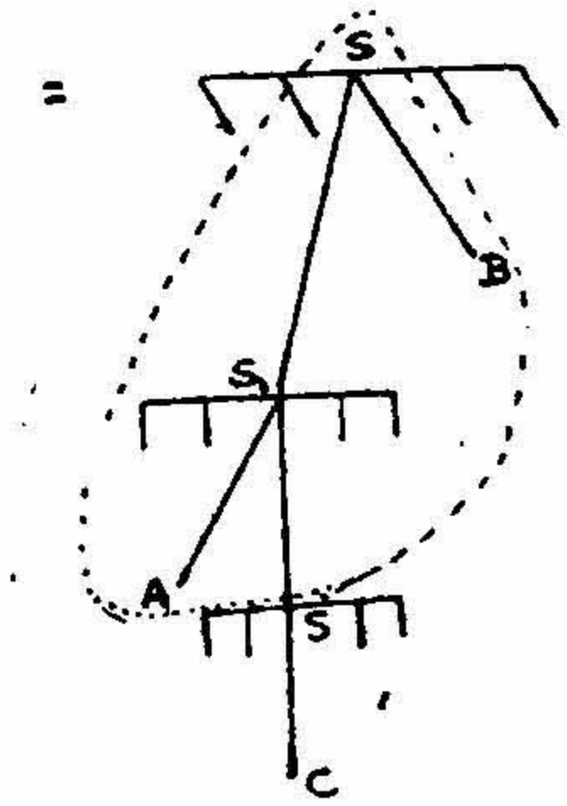
We illustrate TAKAG with an example.

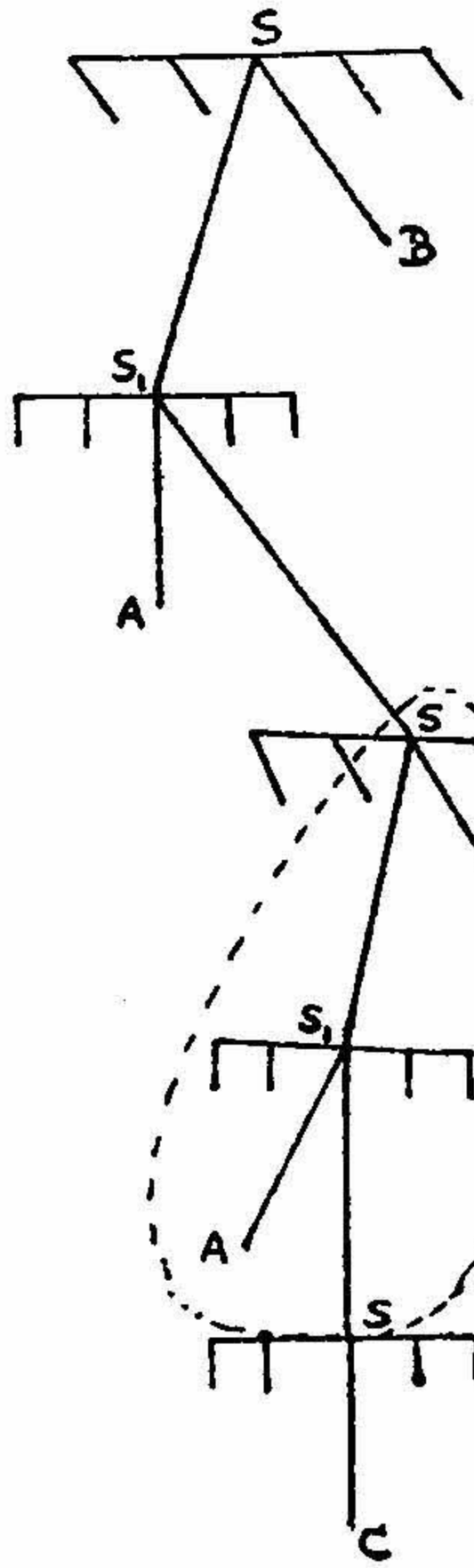
Example 2.2 : Let $G = (V, I, \Sigma, \mathcal{C}, \mathcal{A}, \mathcal{L})$ be a (TA : R) KAG where $V = \{S, S_1\}$, $I = \{A, B, C\}$, $\Sigma = \{., x\}$, $\mathcal{C} = \{t_c\}$, $\mathcal{A} = \{t_a\}$



$\mathcal{L} = \{L_A, L_B, L_C\}$ where $L_A = \{(.)^n / n \geqslant 1\}$, $L_B = \{(x)_n / n \geqslant 1\}$, $L_C = \{x\}$
We describe a sample derivation. In the first phase,

$v_0 = t_c =$ 

$\Rightarrow v_1 = v_0[0, t_a] =$ 

$\Rightarrow v_2 = v_1[1.2, t_a] =$ 

Thus $f(v_2) = ((A\ 0\ ((A\ 0\ C) \textcircled{1}\ B))\ \textcircled{1}\ B)$

In the second phase $f(v_2)$ yields an array (fig. 2) on replacement of the intermediates A, B, C by elements from L_A, L_B, L_C respectively, from the innermost parentheses

subject to conditions for column and row caterations. The (TA : R) KAL gererated by G consists of rectangular arrays describing right triangles of x's.

$$\begin{matrix} ..x \\ .xx \\ xxx \end{matrix}$$

FIG. 2. A right triangle of x's.

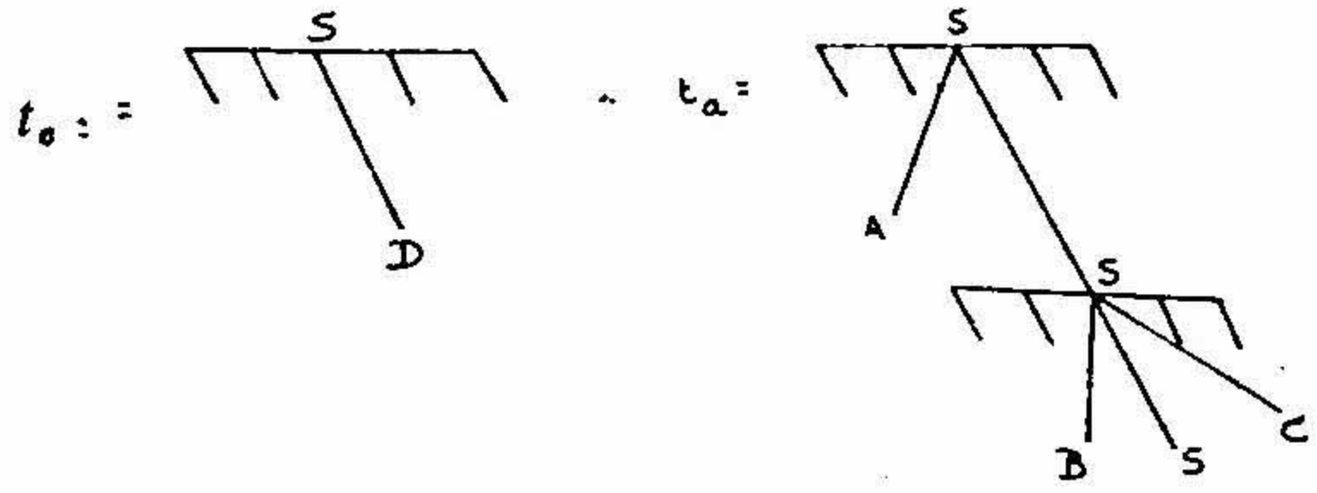
3. Hierarchy and comparisons

In this section, we exhibit a hierarchy of the families of TAKAL's and compare them with other families of KAL's.

Theorem 3.1 : (TA : R) KAL $\underset{+}{\subset}$ (TA : CF) KAL $\underset{+}{\subset}$ (TA : CS) KAL.

Proof : The inclusions follow from the definition of a TAKAL. The proper inclusions can be seen as follows :

Consider the (TA : CS) KAG $G_1 = (\{S\}, \{A, B, C, D\}, \{., x, +\}, \{t_c\}, \{t_a\}, \{L_A, L_B, L_C, L_D\})$ where



and $L_A = \begin{Bmatrix} (.)_n \\ (x)_n / n \geqslant 1 \\ (.)_n \end{Bmatrix}$, $L_B = \begin{Bmatrix} (x)_{2n} / n \geqslant 1 \\ (.)_n \end{Bmatrix}$, $L_C = \{(.)_{3n} / n \geqslant 1\}$,

$L_D = \{(+)_{3n} / n \geqslant 1\}$.

The array language generated by G_1 is a (TA : CS) KAL and this cannot be generated by any (TA : CF) KAG, as the intermediate language L_A is a CSL, requiring context sensitive rules to generate it. This proves that (TA : CF) KAL $\underset{+}{\subset}$ (TA : CS) KAL.

By changing the intermediate language L_A in G_1 as $\begin{Bmatrix} (.)_n \\ (x)_{2n} \end{Bmatrix} / n \geqslant 1$, we can obtain a (TA : CF) KAG G_2. The array language generated by G_2 is then a (TA : CF) KAL and this cannot be generated by any (TA : R) KAG. Thus (TA : R) KAL $\underset{+}{\subset}$ (TA : CF) KAL.

We now recall the definition of a CFKAG[7,9] and then prove that the family of CFKAL's is properly included in the family of TAKAL's.

Definition 3.1 : A Context-free Kolam Array Grammar (CFKAG) is $G = (V, I, \Sigma, P, \mathcal{L}, S)$ where V and Σ are finite sets of symbols ; $V \cap \Sigma = \phi$ and $I \subseteq V$; the elements of I, $V-I$ and Σ are respectively called intermediates, nonterminals and terminals. P is a finite set of rules of the form

$$A \to (B_1 \textcircled{1} \ldots \textcircled{1} B_k) \text{ or } A \to (B_1 0 \ldots 0 B_k),$$

$A \in V - I$, $B_i \in V$, for $i = 1, \ldots, k$. $S \in V - I$ is the start symbol. For each A in I, L_A is a regular, CF or CS intermediate array language generated by A, over a finite number of arrays in Σ^{**}, each of which has the same number of rows or columns.

Derivations proceed as follows. In the first phase of derivations starting with the start symbol S, rules are applied just as in a string CF grammar till all the nonterminals are replaced, introducing parentheses wherever necessary, since the operators $\textcircled{1}$, θ are not associative. Then, in the second phase of derivations, each intermediate A in a string generated in the first phase is replaced by elements from the intermediate array language L_A, subject to the conditions for row and column catenation operators. The replacements start from the innermost parentheses and proceed outwards. The derivation comes to a dead end if the condition for row or column catenation is not satisfied.

The CF Kolam Array Language (CFKAL) generated by G is $L(G) = \{M \in \Sigma^{**} / S \Rightarrow_G^* M\}$.

Definition 3·2 : Let $G = (V, I, \Sigma, P, \mathcal{L}, S)$ be a CFKAG. Let D_G be the smallest subset of T_v such that (i) $\phi \in D_G$, (ii) if $A \in V$, then $\{(0, A)\} \in D_G$, (iii) if $X \to A_1 \oplus \ldots \oplus A_k$ is a rule in P, $X \in V - I$, $A_i \in V$ for $i = 1, \ldots, k$, $\oplus \in \{\textcircled{1}, 0\}$ and $t_j \in D_G$, $t_j(0) = A_j$, $j = 1, \ldots, k$, then $t = \{(0, X)\} \cup (\bigcup_{j=1}^{k} j.t_j) \in D_G$ and the ordered pairs $(0, 1) \ldots, (0, j)$ with $0, 1, \ldots, j$ in D_t are in P_h or P_v according as $\oplus$ is $\textcircled{1}$ or 0, where P_h or P_v are finite subsets of $D_t \times D_t$.

If $t \in D_G$, $t(0) = X$ and $f(t) = w \in V_+^+ \cup \{c\}$, then we say that t is the generalized derivation tree of $X \Rightarrow_G^* w$. We write $t : X \Rightarrow_G^* w$. Let $T(G) = \{t / t : S \Rightarrow_G^* w \in V_+^+ \cup \{c\}\}$. $T(G)$ is the set of all generalized sentential derivation trees of G *i.e.*, trees whose roots are labelled with S, the start symbol of G and whose leaf nodes are labelled with terminal symbols of G.

Theorem 3.2 : For any CFKAG G_1, there is a TAKAG G_2 such that $T(G_1) = T(G_2)$ and $L(G_1) = L(G_2)$ such that the generalized basic trees of G_2 satisfy the following restrictions : if t_c is a generalized center tree of G_2 and t_a is a generalized adjunct tree of G_2, then (i) no nonterminal appears more than once in any path in t_c and (ii) no nonterminal appears more than once in any path in t_a, not counting the nonterminal labelling the root node of t_a.

Proof : Let V be the collection of nonterminals and intermediates of G_1 and I be the set of intermediates of G_1.

We describe the construction of the sets $\mathcal{C}$ and $\mathcal{A}$ of the required TAKAG G_2 as follows : Define

$$\mathcal{C} = \{t_e \in D_{G_1}/t_e : S \Rightarrow_G^* w \in I_+^+ \cup \{e\} \text{ and } t_e \text{ satisfies restriction (i) in the theorem}\}$$

$$\mathcal{A} = \bigcup_{X \in V-I} \mathcal{A}_X \quad \text{where}$$

$$\mathcal{A}_X = \{t_a \in D_{G_1}/t_a : X \Rightarrow_{G_1}^* (w_1) \oplus (X) \oplus (w_2),\ w_1, w_2 \in I_+^+ \cup \{\epsilon\},\ X \in V - I \text{ and } t_a \text{ satisfies restriction (ii) in the theorem}\}.$$

It is easy to see that $T(G_1) = T(G_2)$ and $L(G_1) = L(G_2)$.

Theorem 3.3 : For $X = R$, CF or CS, the family of (CF : X) KAL's is properly contained in the family of (TA : X) KAL's.

Proof : The inclusion in this theorem follows from Theorem 3.2. For X = CF or CS, proper inclusions in the theorem follow from examples given in Theorem 3.1 noting that the array languages generated by G_1 and G_2 are respectively (CS : CS) KAL and (CS : CF) KAL. For $X = R$, we can modify G_1 in Theorem 3.1 by changing L_A as $\{(.)_{3n}/n \geq 1\}$, $L_B = \{(x)_{3n}/n \geq 1\}$, and obtain a (TA : R) KAG G_3 generating a (TA : R) KAL. This array language is a (CS : R) KAL and this proves that (CF : R) KAL. $\subset$ (TA : R) KAL.
$+$

Theorem 3.4 : The family of (TA : X) KAL's for $X = R$, CF or CS, is closed under union, column and row catenations and column star and row star.

Proof : Closure under union is obvious. We outline the proof of closure under array catenations.

If L_1 and L_2 are (TA : X) KAL's generated respectively by (TA : X) KAG's $G_1 = (V_1, I_1, \Sigma, \mathcal{C}_1, \mathcal{A}_1, \mathcal{L}_1)$ and $G_2 = (V_2, I_2, \Sigma, \mathcal{C}_2, \mathcal{A}_2, \mathcal{L}_2)$, $V_1 \cap V_2 = \phi$ and $X = R$, CF or CS, then a (TA : X) KAG $G = (V, I, \Sigma, \mathcal{C}, \mathcal{A}, \mathcal{L})$ can be formed to generate $L_1 \oplus L_2$ as follows :

$V = V_1 \cup V_2 \cup \{S\}$, $S \notin V_1 \cup V_2$; $I = I_1 \cup I_2$; $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$; $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$. For $t_1 \in \mathcal{C}_1$ and $t_2 \in \mathcal{C}_2$, $t = (0, S) \cup 1.t_1 \cup 2.t_2$ is in $\mathcal{C}$, such that the branches (0, 1), (0, 2) of t are horizontal or vertical according as $\oplus$ is ① or 0. It is clear that $L(G) = L_1 \oplus L_2$.

Closure under star can be proved in a similar manner.

Finally, we define the notions of linear and one-sided linear TAKAG's and compare the families of TAKAL's generated by them with the families of linear[8] and regular KAL's.

Definition 3.3 : A generalized tree is linear iff at any depth there is atmost one non-terminal. A TAKAG is linear iff all of its basic trees are linear. The array language generated by a linear TAKAG is a linear TAKAL.

Therorem 3.5 : For $X = R$, CF or CS, the family of (Linear : X) KAL is properly included in the family of (Linear TA : X) KAL.

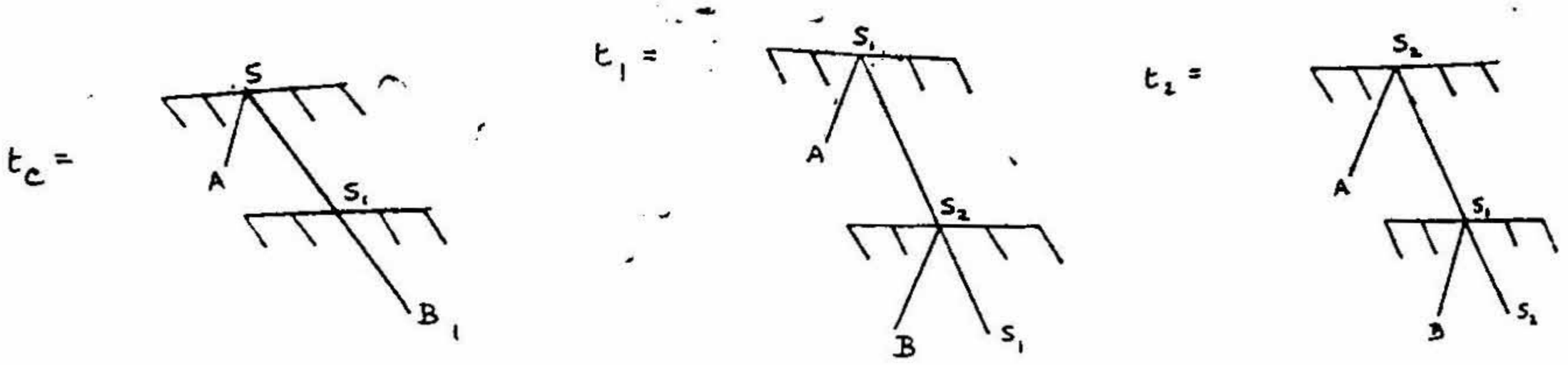
Proof : The theorem follows from Theorem 3.3 noting that the trees t_c and t_a in the examples of Theorem 3.1 are, in fact, linear.

Definition 3.4 : A generalized tree is up-right linear iff it is linear and at any depth the nonterminal is (i) the left most symbol at that depth if the branch connecting it to its predecessor is vertical and (ii) the right most symbol at that depth if the branch connecting it to its predecessor is horizontal. We can similarly define generalized up-left, down-right, down-left linear trees. A generalized tree is one-sided linear if it is up-right or up-left or down-right or down-left linear. A TAKAG is one-sided linear iff its basic trees are one-sided linear. The language generated by it is a one-sided linear TAKAL.

Theorem 3.6 : For $X = R$, CF or CS, the family of $(R \cdot X)$ KAL is properly included in the family of (one-sided linear TA : X) KAL.

Proof : The inclusion can be seen by noting that if L is a regular kolam array language then it can be generated by either an up-right or up-left or down-right or down-left linear KAG and hence can be generated by a one-sided linear TAKAG.

To prove the proper inclusion for $X = R$, consider the one-sided linear TAKAG $G = (\{S, S_1, S_2\}, \{A, B\}, \{., x\}, \{t_c\}, \{t_1, t_2\}, \{L_A, L_B\}$ where



and $L_A = \{(.)_n / n \geq 1\}$, $L_B = \{(x)_n / n \geq 1\}$. Clearly the array language generated by this one-sided TAKAG cannot be generated by any $(R : R)$ KAG, since the language generated in the first phase is non-regular[2]. Similarly, for the cases, $X =$ CF or CS.

Acknowledgements

The authors would like to thank Prof. Rani Siromoney and the referees for useful comments which improved the presentation of the paper. The first author acknowledges the kind encouragement of Prof. R. Narasimhan and Dr. R. K. Shyamasundar and the financial support from the UGC.

References

1. JOSHI, A. K., KOSARAJU, S. R. AND YAMADA, H. — String adjunct grammars, *Inf. Control*, 1972, **21**, 93–116.

2. JOSHI, A. K. AND LEVY, L. S. — Tree adjunct grammars, *J. Comput. Sys. Sci.*, 1975, **10**, 136–163.

3. NARASIMHAN, R. — Labelling schemata and syntactic description of pictures, *Inf. Control*, 1964, **7**, 151–179.

4. ROSENFELD, A. — Array grammar normal forms, *Inf. Control*, 1973, **23**, 173–182.

5. SALOMAA, A. — *Formal languages*, Academic Press, 1973.

6. SIROMONEY, G., SIROMONEY, R. AND KRITHIVASAN, K. — Abstract families of matrices and picture languages, *Comput. Graph. Image Processing*, 1972, **1**, 284–307.

7. SIROMONEY, G., SIROMONEY R. AND KRITHIVASAN, K. — Picture languages with array rewriting rules, *Inf. Control*, 1973, **22**, 447–470.

8. SIROMONEY, R., SUBRAMANIAN, K. G. AND RANGARAJAN, K. — Control on kolam arrays, *Inf. Control*, 1976, **32**, 272–275.

9. SUBRAMANIAN, K. G. — Studies in array languages, Ph.D. Thesis, University of Madras, 1979.