

recorded

AN INTRODUCTION TO TREE ADJOINING GRAMMARS¹

Aravind K. Joshi
University of Pennsylvania

1. INTRODUCTION

In this paper, we will give a brief introduction to Tree Adjoining Grammars (TAG) and summarize some of the mathematical properties of TAG's. This paper is primarily based on Joshi, Levy, and Takahashi (1975); Joshi (1985 [1983]); Vijay-Shanker and Joshi (1985); Joshi, Vijay-Shanker and Weir (1986); and Vijay-Shanker (1986). The linguistic significance of TAG's is discussed in Joshi (1985 [1983]), Kroch and Joshi (1985, 1986).

The main characteristics of TAG's are as follows.

1. TAG is a tree generating system. It consists of a finite set of elementary trees (elaborated up to preterminal (terminal) symbols) and a composition operation (adjoining) which builds trees out of elementary trees and trees derived from elementary trees by adjoining. The terminal strings of a TAG constitute a string language. However, a TAG should be viewed primarily as a tree generating system in contrast to a string generating system such as a context-free grammar or some of its extensions.
2. TAG's factor recursion and dependencies in a novel way. The elementary trees are the domain of dependencies which are statable as co-occurrence relations among the elements of the elementary trees and also relations between elementary trees. Recursion enters via the operation of adjoining. Adjoining preserves the dependencies. Localization of dependencies in this manner has both mathematical and linguistic significance. Such localization cannot be achieved directly in a string generating system.
3. TAG's are more powerful than context-free grammars, but only "mildly" so. This extra power of TAG is a direct corollary of the way TAG factors recursion and dependencies.

The plan of the paper is as follows. In Section 2, we will describe the

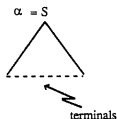
¹This work was partially supported by ARO grant DAA29-84-9-0027, NSF grant MCS-8219116-CER, NSF grants MCS-82-07294, DCR-84-10413, MCS 83-05221, DARPA grant N00014-85-K-0018

I want to thank A. Kroch, K. Vijay-Shanker, and D. Weir for their valuable comments.

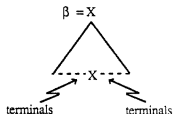
formalism for TAG giving several examples. We will also briefly state some of the mathematical properties. In Section 3, we will briefly describe multicomponent adjoining (used by Kroch in his paper in this volume) and some of its properties.

2. TREE ADJOINING GRAMMAR FORMALISM

A *tree adjoining grammar* (TAG) $G = (I, A)$ where I and A are finite sets of *elementary trees*. The trees in I will be called the *initial trees* and the trees in A , the *auxiliary trees*. A tree α is an initial tree if it is of the form in (1):



That is, the root node of α is labelled S and the frontier nodes are all non-terminals. A tree β is an auxiliary tree if it is of the form in (2):



That is, the root node of β is labelled X where X is a non-terminal and the frontier nodes are all terminals except one which is labelled X , the same label as that of the root. The node labelled X on the frontier will be called the foot node of β . The internal nodes are non-terminals. The initial and the auxiliary trees are not constrained in any manner other than as indicated above. The idea, however, is that both the initial and auxiliary trees will be *minimal* in some sense. An initial tree will correspond to a *minimal* sentential tree (i.e., without recursing on any

non-terminal) and an auxiliary tree, with root and foot node labelled X, will correspond to a *minimal* recursive structure that must be brought into the derivation, if one recurses on X.

We will now define a composition operation called *adjoining* (or *adjunction*), which composes an auxiliary tree β with a tree γ . Let γ be a tree containing a node n bearing the label X and let β be an auxiliary tree whose root node is also labelled X. (Note that β must have, by definition, a node (and only one such) labelled X on the frontier.) Then the adjunction of β to γ at node n will be the tree γ' that results when the following complex operation is carried out:

- 1) The sub-tree of γ dominated by n , call it t , is excised, leaving a copy of n behind.
- 2) The auxiliary tree β is attached at n and its root node is identified with n .
- 3) The sub-tree t is attached to the foot node of β and the root node n of t is identified with the foot node of β .

Figure 1 illustrates this operation.

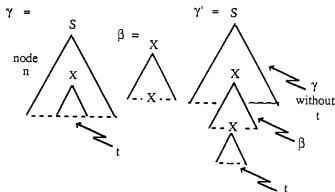
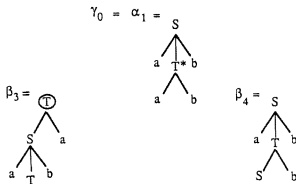


Figure 1

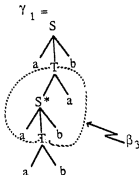
The intuition underlying the adjoining operation is a simple one but the operation is distinct from other operations on trees that have been discussed in the literature. In particular, we want to emphasize that adjoining is not a substitution

operation². Let us now look at some derivations in the TAG, $G=(I,A)$.

EXAMPLE 2.1



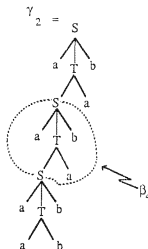
β_3 will be adjoined to γ_0 at T as indicated in γ_0 . (We will use * to indicate the node to which adjunction is made). The resulting tree γ_1 is then



We can continue the derivation by adjoining, say, β_4 , at S as indicated in γ_1 . The

²Adjoining reduces to substitution only in the special case where an auxiliary tree adjoins to the root node of another tree so that it "sits on top of" the tree to which it is adjoined. In this special case the adjoining operation has the same effect as would the substitution of a tree at its root node for the root node of the auxiliary tree.

resulting tree γ_2 is then



Note that γ_0 is an initial tree, a sentential tree. The derived trees γ_1 and γ_2 are also sentential trees. It can be shown that the string language of this TAG is a context-free language.

Let us now define two auxiliary notions, the tree set of a TAG grammar and the string language of a TAG. Suppose $G=(I,A)$ is a TAG with a finite set of initial trees, a finite set of auxiliary trees, and the *adjoining* operation, as above. Then we define the *tree set* of a TAG G , $T(G)$, to be the set of all trees derived in G starting from initial trees in I . We further define the *string language* (or *language*) of G to be the set of all terminal strings of the trees in $T(G)$. The relationship between TAG's, context-free grammars, and the corresponding string languages can then be summarized in the following theorems Joshi, Levy, and Takahashi (1975), Joshi (1985 [1983]):

THEOREM 2.1: For every context-free grammar, G' , there is a TAG, G , which is both weakly and strongly equivalent to G' . In other words, $L(G)=(G')$ and $T(G)=T(G')$.

THEOREM 2.2: There exists a non-empty set of TAG grammars G_1 such that for every $G \in G_1$, $L(G)$ is context-free but there is no CFG G' such that $T(G')=T(G)$.

THEOREM 2.3: There exists a non-empty set of TAG grammars G_2 such

that for every $G \in G_2$, $L(G)$ is strictly context sensitive; that is, there is no CFG grammar G' such that $L(G) = L(G')$.

Theorems 2.1 and 2.3 appear in Joshi, Levy, and Takahashi (1975). Theorem 2.2 is implicit in that paper, but we make it explicit here because of its linguistic importance. Examples 2.2 and 2.3 below illustrate theorems 2.2 and 2.3 respectively.

EXAMPLE 2.2: Let $G = (I, A)$ where

I:



A:

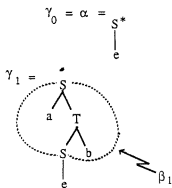
$\beta_1 =$



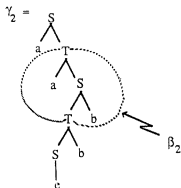
$\beta_2 =$



The language generated by G is context-free; but there is no CFG that is strongly equivalent to G . We can see this if we examine some derivations in G . Thus, consider the following trees:



$\gamma_1 = \gamma_0$ with β_1
with β_2
adjoined at S as
as indicated in γ_0 .



$\gamma_2 = \gamma_1$
adjoined at T as
indicated in γ_2 .

Clearly, $L(G)$ is $\{a^n e b^n / n \geq 0\}$, which is a context-free language. Thus, there must exist a context-free grammar, G' , which is at least weakly equivalent to G . It can be shown however that there is no context-free grammar G' which is strongly equivalent to G ; i.e., for which $T(G) = T(G')$. This follows from the fact that $T(G)$ is *non-recognizable*; i.e., there is no finite state bottom-up tree automaton that can recognize precisely $T(G)$. Thus a TAG may generate a context-free language, yet assign structural descriptions to the strings that cannot be assigned by any context-free grammar.

EXAMPLE 2.3: Let $G = (I, A)$ where

I:

$\alpha_1 =$



A:

$\beta_1 =$



$\beta_2 =$



The precise definition of $L(G)$ is as follows:

$L(G) = L_1 = \{w e c^n / n \geq 0, w \text{ is a string of } a\text{'s and } b\text{'s such that}$

(1) the number of a 's = the number of b 's = n , and

(2) for any initial substring of w , the number of a 's \geq the number of b 's. }

L_1 is a strictly context-sensitive language (i.e., a context-sensitive language that is not context-free). This can be shown as follows. Intersecting L with the finite state language $a^* b^* e c^*$ results in the language

$$L_2 = \{ a^n b^n e c^n / n \geq 0 \} = L_1 \cap a^* b^* e c^*$$

L_2 is well-known strictly context-sensitive language. The result of intersecting a context-free language with a finite state language is always a context-free language; hence, L_1 is not a context-free language. It is thus a strictly context-sensitive language. Example 2.3 thus illustrates Theorem 2.3.

We have seen that TAG's have more power than CFG's, but the extra power is quite limited. Joshi (1985 [1983]) characterizes this limitation in detail, but the above example gives some indication of its nature. The language L_1 has an equal

number of a's, b's and c's; however, the a's and b's are mixed in a certain way. The language L_2 is similar to L_1 , except that a's come before all b's. TAG's as defined so far are not powerful enough to generate L_2 . This can be seen as follows. Clearly, for any TAG for L_2 , each initial tree must contain equal number of a's, b's and c's (including zero), and each auxiliary tree must also contain equal number of a's, b's and c's. Further in each case the a's must precede the b's. Then it is easy to see from the grammar of Example 2.3, that it will not be possible to avoid getting the a's and b's mixed. However, L_2 can be generated by a TAG with local constraints (see Section 2.3) The so-called copy language

$$L = \{ wew / w \in \{a,b\}^* \}$$

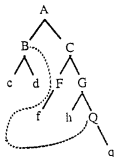
also cannot be generated by a simple TAG but can be by a TAG with local constraints. Furthermore, it can be shown that TAG's even with local constraints, cannot generate all context-sensitive languages (Joshi (1985 [1983])).

Although TAG's are more powerful than CFG's, this extra power is highly constrained and apparently it is just the right kind for characterizing certain structural descriptions. TAG's share almost all the formal properties of CFG's (more precisely, the corresponding classes of languages). The string languages of TAG's can also be parsed in polynomial time, in particular in time Kn^6 , or less, where K is a constant that depends on the grammar and n is the length of the string (see Vijay-Shanker and Joshi 1985 for further details).

2.1 TAG's WITH "LINKS"

Elementary trees (initial and auxiliary trees) are the appropriate domains for characterizing certain dependencies (e.g., subcategorization dependencies and filler-gap dependencies). The characterization of certain of these dependencies can be achieved by introducing a special relationship between certain specified pairs of nodes of an elementary tree. This relationship, which we shall call "linking," is pictorially exhibited by an arc (a dotted line) from one node to the other. For example, in the tree in (3) below, the nodes labelled B and Q are linked.

(3)



Linking can be defined for any two nodes in an elementary tree. However, in the linguistic context we will require the following conditions to hold for a link in an elementary tree:

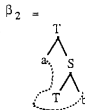
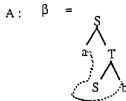
If a node n_1 is linked to a node n_2 then

1. n_2 c-commands n_1 , (i.e., n_2 does not dominate n_1 and there exists a node m which immediately dominates n_2 and also dominates n_1).
2. n_1 and n_2 have the same label.
3. n_1 dominates a null string (or a terminal symbol in the non-linguistic formal grammar examples).

The notion of linking thus defined is related to the one discussed in Peters and Ritchie (1982). A TAG with links is a TAG in which some of the elementary trees may have links as defined above. Henceforth, we may refer to a TAG with links just as a TAG.

Links are defined on the elementary trees. However, the important point is that the composition operation of adjoining will *preserve* the links. Links defined on the elementary trees may become *stretched* as the derivation proceeds. Example 2.4 will illustrate this point.

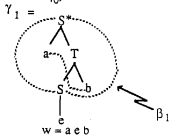
EXAMPLE 2.4: Let $G=(I, A)$ where



Let

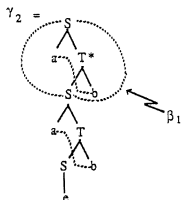


Adjoining β_1 at S as indicated in γ_0 , we have



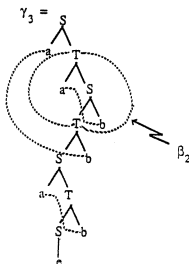
The terminal string corresponding to γ_1 is $a e b$, where the dependency is indicated by the solid line.

Adjoining β_2 again at S as indicated in γ_1 , we have the following tree



$w = a a e b b$ (nested dependencies)

Adjoining β_2 at T as indicated in γ_2 , we have



$w = a a a e b b b$ (cross-serial and nested dependencies)

In this example β_1 and β_2 each have one link, and the composed trees γ_2 and γ_3 show how linking is preserved under adjunction. In γ_3 , for example, one of the links is stretched. It should be clear now how, in general, the links will be preserved during the derivation and we shall not give a formal treatment of this property here.

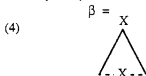
We should also note with regard to the above example that the dependencies in γ_2 between the a's and b's, as reflected in the terminal string, are properly nested, while in γ_3 two of them are properly nested, and the third one is cross-serial. The cross-serial one is crossed with respect to the nested ones (not, of course, a unique description). The two elementary trees β_1 and β_2 have only one link each so that the nestings and crossings in γ_2 and γ_3 are the result of adjoining. There are two points of importance here:

1. TAG's with links can characterize certain cross-serial dependencies (as well as, of course, nested dependencies, which is not a surprise).
2. The cross-serial dependencies (as well as the nested dependencies) arise as a result of adjoining. But this is not the only way they can arise. It is possible to have two links in an elementary tree which represent cross-serial or nested dependencies, which will then be preserved during the derivation. Thus cross-serial dependencies, as well as nested dependencies, will arise in two distinct ways - either by adjoining or by being present in some elementary trees to start with.

It is clear from our example that the string language of TAG with links is not affected by the links; that is, links do not affect weak generative capacity. However, they make certain aspects of the structural description explicit which are implicit in a TAG without links. In principle, any two nodes in an elementary tree are 'related' simply by virtue of the fact that they belong to the same tree, and this relationship will be symmetrical. 'Linking' as defined here is an asymmetrical relation introduced primarily to characterize filler-gap dependencies.

2.2 TAG'S WITH LOCAL CONSTRAINTS ON ADJOINING

The adjoining operation as defined in Section 2.1 is "context-free". An auxiliary tree β with the form in (4)



is adjoinable to a tree t at a node n if the label of node n is X , independently of the (tree) context around n . In this sense, adjoining is context-free. In Joshi (1983 [1983]), local constraints on adjoining similar to those investigated by Joshi and Levy (1978) were used. These are a generalization of the context-sensitive constraints studied by Peters and Ritchie (1969). It was soon recognized, however, that the full power of these constraints was never exploited, either in the linguistic

context or in the "formal language" cases. The so-called proper analysis contexts and domination contexts (as defined in Joshi and Levy (1978)) that were actually used in Joshi (1985 [1983]) always turned out to be such that the context elements were within a single elementary tree; that is, they were far more localized than the definitions required. Based on this observation and a suggestion in Joshi, Levy and Takahashi (1975), we will describe a new way of introducing local constraints. This approach not only captures the insight stated above, but it is more in the spirit of the TAG formalism, with its emphasis on locality. (For further details, see Vijay-Shanker and Joshi 1985) The earlier approach was less so, although it was certainly adequate for the investigation in Joshi (1985 [1983]). A precise characterization of the original approach remains an open problem.

Let $G = (I, A)$ be a TAG with local constraints if for each elementary tree $t \in I \cup A$, and for each node, n , in t , we specify the set \bar{P} of auxiliary trees that can be adjoined at the node n . Note that if there is no constraint then any auxiliary tree whose root has the same label as the label of the node n is adjoinable at n . Thus, in general, \bar{P} is a subset of the set of all auxiliary trees structurally adjoinable at n .

We adopt the following conventions for the statement of local constraints:

1. Since, by definition, no auxiliary trees are adjoinable to a node labelled by a terminal symbol, no constraint has to be stated for a node labelled by a terminal.
2. If there is no constraint, i.e., all auxiliary trees with the appropriate root label are adjoinable at a node n , then we will not state this explicitly, as this is the case we have discussed in Section 2.1.
3. If no auxiliary trees are adjoinable at a node n , then we will write the constraint as (ϕ) , where ϕ denotes the null set.
4. We will also allow for the possibility that for a given node at least one adjoining is obligatory from the set of all possible auxiliary trees adjoinable at that node, of course.

Hence, a TAG with local constraints is defined as follows. $G = (I, A)$ is a TAG with local constraints if for each node n , in each tree t , one (and only one) of the following constraints is specified:

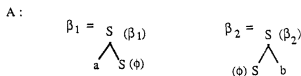
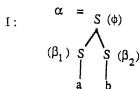
1. *Selective Adjoining (SA)*: Only a specified subset of the set of all auxiliary trees are adjoinable at n . SA is written as (\bar{P}) , where \bar{P} is a subset of the set of all auxiliary trees structurally adjoinable at n .

If \bar{P} equals the set of all auxiliary trees adjoinable at n , then we do

not explicitly state this at the node n.

2. *Null Adjoining (NA)*: No auxiliary tree is adjoinable at the node N. NA will be written as (ϕ) .
3. *Obligatory Adjoining (OA)*: At least one (out of all the auxiliary trees adjoinable at n) must be adjoined at n. OA is written as $O(\bar{\beta})$ where $\bar{\beta}$ is a subset of the set of all auxiliary trees adjoinable at n.

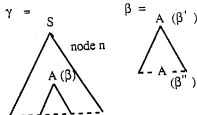
EXAMPLE 2.5: Let $G = (I, A)$ be a TAG with local constraints where



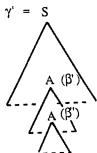
In α_1 no auxiliary trees can be adjoined to the root node. Only β_1 is adjoinable to the left S node at depth 1 and only β_2 is adjoinable to the right S node at depth 1. In β_1 only β_1 is adjoinable at the root node and no auxiliary trees are adjoinable at the foot node. Similarly for β_2 .

We must now modify our definition of adjoining to take care of the local constraints. Given a tree γ with a node n labelled A and given an auxiliary tree β with the root node labelled A, we shall modify our definition of adjoining as follows: β is adjoinable to γ at node n if $\beta \in \bar{\beta}$, where $\bar{\beta}$ is the constraint associated with node n in γ . The result of adjoining β to γ will be as defined in Section 2.1, except that the constraint $\bar{\beta}$ associated with n will be replaced by $\bar{\beta}'$, the constraint associated with the root node of β and by $\bar{\beta}''$; the constraint associated

with the foot node of B. Thus, given

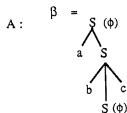
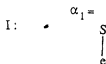


the resultant tree γ' is



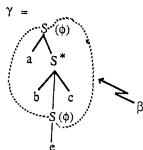
We also adopt the convention that any derived tree with a node which has an OA constraint associated with it will not be included in the tree set associated with a TAG, G. The string language L of G is then defined as the set of all terminal strings of all trees derived in G (starting with initial trees) which have no OA constraints left in them.

EXAMPLE 2.6: Let $G = (I, A)$ be a TAG with local constraints where

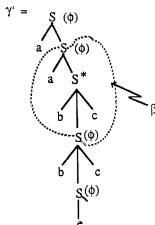


There are no constraints in α_1 . In β no auxiliary trees are adjoinable at the root node and the foot node and for the center S node there are no constraints.

Starting with α_1 and adjoining β to α_1 at the root node we obtain



Adjoining β to the center S node (the only node at which adjunction can occur) we have



It is easy to see that G generates the string language


$$L = \{ a^n b^n e c^n / n \geq 0 \}$$


EXAMPLE 2.7: Let G' be a TAG similar to G in Example 2.6, except that in G' there are no constraints in β . G' generates

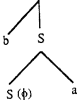
$$L = \{ w e c^n / n \geq 0, \# a\text{'s in } w = \# b\text{'s in } w = n, \\ \text{and for any proper initial string } u \\ \text{of } w, \# a\text{'s in } u \geq \# b\text{'s in } u. \}$$

This is the same language as in Example 2.3. This language is closely related to the context-sensitive language discussed in Higginbotham (1984), which can also be shown to be a TAG language.

EXAMPLE 2.8: Let $G = (I, A)$ be a TAG with local constraints where

$$I: \quad \alpha_1 =$$


$$A: \quad \beta_1 =$$


$$\beta_2 =$$


G generates the language

$$L = \{ w e w / w \in \{a,b\}^* \}$$

EXAMPLE 2.9: Let G' be a TAG which is the same as G in Example 2.8 but without any local constraints. The corresponding language is

$$L = \{ w e w' / w, w' \in \{a,b\}^*, w = w' = 2n, \\ \# \text{ a's in } w = \# \text{ a's in } w' = \# \text{ b's} \\ \text{ in } w = \# \text{ b's in } w' = n \}$$

This language is related to the Swiss-German example in Shieber (1984).

EXAMPLE 2.10: Let $G = (I, A)$ be a TAG with local constraints where

$I:$

$$\alpha_1 = \begin{array}{c} S \\ | \\ e \end{array}$$

$A:$

$$\beta = \begin{array}{c} S(\phi) \\ / \quad \backslash \\ a \quad S \quad d \\ | \quad / \quad \backslash \\ b \quad S \quad c \\ | \\ S(\phi) \end{array}$$

G generates

$$L = \{ a^n b^n e c^n d^n / n \geq 0 \}$$

Note that it can be shown that languages

$$L^1 = \{ a^n b^n c^n d^n e^n / n \geq 1 \}$$

and

$$L^2 = \{ w w w / w \in \{a,b\}^* \}$$

cannot be generated by TAG's either with or without local constraints (Joshi 1985 [1983]). Other languages such as $L' = \{ a^n / n \geq 1 \}$ also cannot be generated by TAG. This is because the strings of a TAG grow linearly (for a detailed definition of this property, called the "constant growth" property, see (Joshi (1985 [1983]))). L' does not satisfy this property.

For those familiar with Joshi (1985 [1983]), it is worth pointing out that the SA constraint is only abbreviating i.e., it does not affect the power of TAG's. The NA and OA constraints, however, do affect the generative power of TAG's. Thus,

NA is needed to generate the languages in Examples 2.6, 2.7, and 2.8. OA is needed to generate the language in Example 2.11 below:

EXAMPLE 2.11: Let $G=(I, A)$ be a TAG with local constraints

$$I: \quad \alpha_1 = \begin{array}{c} S \text{ } o(\beta_1, \beta_2) \\ | \\ e \end{array}$$

$$A: \quad \beta_1 = \begin{array}{c} S(\phi) \\ \swarrow \quad \searrow \\ a \quad S \text{ } o(\beta_1, \beta_2) \\ \swarrow \quad | \quad \searrow \\ b \quad S(\phi) \quad c \end{array}$$

$$\beta_2 = \begin{array}{c} S(\phi) \\ \swarrow \quad \searrow \\ f \quad S(\phi) \end{array}$$

G generates

$$L = \{a^n f b^n e c^n \mid n \geq 0\}$$

0 or more adjunctions of β_1 generates

$$\{a^n b^n e c^n \mid n \geq 0\}$$

The resulting trees each have one node still with an OA constraint, which can be removed by adjoining β_2 , generating L .

In contrast to Joshi (1985 [1983]), where we stated for each auxiliary tree the constraints on its adjoinability, we have here stated, for each node in each elementary tree, the constraints on what auxiliary trees can be adjoined there. This way of looking at local constraints has not only greatly simplified their statement, but it has also allowed us to capture the insight that the 'locality' of the constraints is storable in terms of the elementary trees themselves!

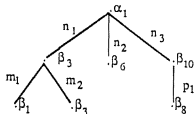
2.3 DERIVATION IN A TAG

Although we shall not describe formally the notion of derivation in a TAG, we want to give the reader a more precise understanding of the concept than (s)he

might form from our illustrative examples. Adjoining is an operation defined on an elementary tree, say γ , an auxiliary tree, say β , and a node (i.e., an address) in γ , say n . Thus, every instance of adjunction is of the form " β is adjoined to γ at n ," and this adjunction is always and only subject to the local constraints associated with n . Although we very often speak of adjoining a tree to a node in a complex structure, we do so only for convenience. Strictly speaking, adjoining is always at a node in an elementary tree; and, therefore, it is more precise to talk about adjoining at an address in an elementary tree. More than one auxiliary tree can be adjoined to an elementary tree as long as each tree is adjoined at a distinct node. After these auxiliary trees are adjoined to the elementary tree, only nodes in the auxiliary trees are available for further adjunction. This precision in the definition of adjunction will be necessary when we define multicomponent adjunction in section 3 below.

Now suppose that α is an initial tree and that β_1, β_2, \dots are auxiliary trees in a TAG, G . Then the derivation structure corresponding to the generation of a particular string in $L(G)$ might look as follows:

D:



α_1 is an initial tree. β_3 , β_6 and β_{10} are adjoined at nodes n_1 , n_2 , and n_3 respectively in α_1 , where n_1 , n_2 , and n_3 are all distinct nodes. β_1 and β_3 are adjoined to β_3 at nodes m_1 and m_2 respectively. Again, m_1 and m_2 are distinct. β_6 has no further adjunctions but β_8 is adjoined to β_{10} at node p_1 . Note that the derivation structure D implicitly characterizes the 'surface' tree that is generated by it. D also serves as the basis for defining a compositional semantic interpretation (Vijay-Shanker 1986). In this way the derivation structure can be seen as the basic formal object constructed in the course of sentence generation. Associated with it will be two mappings, one to a surface syntactic tree and the other to a semantic interpretation, as below:

surface tree <---- derivation structure ----> semantic interpretation

2.4 SOME FORMAL PROPERTIES OF TAG's

Here we will state without proof some additional formal properties of TAG's.

1. Closure properties (Vijay-Shanker and Joshi 1985, Vijay-Shanker 1986) •

Tree Adjoining Languages (TAL) are used under

- union
- concatenation
- Kleene-star
- intersection with regular languages
- substitution
- homomorphism
- inverse homomorphism

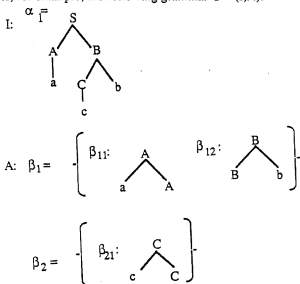
It follows, therefore, that TAL form a Full AFL (Abstract Family of Languages).

2. A pumping lemma, similar to that for context-free languages has been established for TAL's. This lemma allows one to establish that certain languages are not TAL's, for example, L^1 and L^2 in Example 2.10 (Vijay-Shanker 1986).
3. The tabular parsing algorithm for context-free grammars (the so-called CKY algorithm) can be extended in a natural fashion for parsing TAL's, although the extension is not immediate because adjoining is not a substitution operation. The time bound of the parsing algorithm is proportional to n^6 , where n is the length of the string to be parsed, as compared to the n^3 bound that has been established for context-free grammars (Vijay-Shanker and Joshi 1985).
4. Head Grammars (HG) were introduced by Pollard (1984). The wrapping operation in HG is undefined for empty strings. If this formal deficiency of HG is fixed then it can be shown that HG's are equivalent to TAG's, in the weak sense. For details see (Vijay-Shanker, Weir and Joshi 1986 and Joshi, Vijay-Shanker and Weir 1986).
5. Recently, Vijay-Shanker (1986) has established several string and tree automata related results about TAG's, including a variant of a push-down automata that corresponds exactly to TAL's.

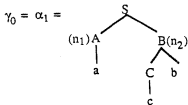
6. Vijay-Shanker, Weir and Joshi (this volume) have studied a variety of relations between regular languages, linear context free languages, context-free languages, linear tree adjoining languages, tree adjoining languages, Dyck languages and a hierarchy of languages that are appropriate generalizations of tree adjoining languages.

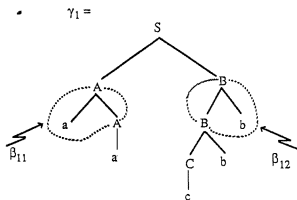
3. MULTICOMPONENT ADJOINING

In Joshi, Levy, Takahashi (1975) a version of the adjoining operation is introduced under which, instead of a single auxiliary tree, a set of such trees is adjoined to a given elementary tree. We define the adjunction of such a set as the simultaneous adjunction of each of its component trees to a distinct node (address) in an elementary tree. This adjunction can, of course, take place only if the local constraints associated with each affected node of the elementary tree are satisfied. Consider, for example, the following grammar $G = (I, A)$:



β_1 is an auxiliary set consisting of the two trees β_{11} and β_{12} . Here is a sample derivation in G:





γ_1 above, it should be clear, is obtained by the adjunction of the components β_{11} and β_{12} of the auxiliary set β_1 to γ_0 at the nodes n_1 and n_2 respectively. In the current example, the set β_1 has two component trees and β_2 has only one component. If every auxiliary tree set of a TAG has only one component, we have a TAG as defined in Section 2. It can be shown that the number of components in the auxiliary sets does not make any difference to the generative capacity i.e., both the weak and strong (with respect to tree sets generated and not, of course, with respect to the derivation structures) generative capacities of multicomponent TAG are the same as that for TAG where each auxiliary set has exactly one component. On the other hand, derived auxiliary sets can be defined by adjoining an auxiliary set, say β_1 , to another auxiliary set, say β_2 , as follows. Each component of β_1 is adjoined to one (and exactly one) component of β_2 and all adjunctions are at distinct nodes. Note that since it is not required that each component of β_1 adjoins to the same component of β_2 , one component may adjoin to one component and another component to a different component of β_2 , i.e., adjunctions of components are not to the same component (elementary tree) of β_1 , but they are all adjunctions to the *same auxiliary set*. Thus, *locality* of adjoining can be defined in two ways: (1) by requiring that all components of an auxiliary set adjoin to the *same*

elementary tree, (2) by requiring that all components of an auxiliary set adjoin to the *same auxiliary set*, not necessarily to the same elementary tree. The first type of locality does not add to the generative capacity of the multicomponent TAG. The second type of locality does add to the weak generative capacity of the multicomponent TAG; however, the resulting class of languages still falls within the class of "mildly context sensitive" languages characterized in Joshi (1985 [1983]). With the second type of locality a multicomponent TAG can be defined for the language $L' = \{a^n b^n \mid n \geq 1\}$ such that the *a*'s all hang from one path from the root node *S* and the *b*'s all hang from another path from the root node. Such a structural description cannot be provided by TAG where each auxiliary set has exactly one component (see also Joshi (1985 [1983])). A study of these two types of localities will be pursued in a later paper.

The notion of local constraints extends to TAG's with auxiliary sets in a straightforward way. The null adjoining constraint (NA) carries over directly. For the selective adjoining constraint (SA) and the obligatory adjoining (OA) constraint, which specify one or more auxiliary trees to be adjoined at a node, we need to modify the definition of the constraints because adjoining now takes place at two distinct nodes (more than two nodes, if there are more than two components in the auxiliary set). Let us say that if an auxiliary set β_1 (with components β_{11} and β_{12}) is adjoinable at nodes n_1 and n_2 in an elementary tree γ , then β_{11} and β_{12} will be specified at the nodes n_1 and n_2 respectively. Since the two components β_{11} and β_{12} are to be adjoined at distinct nodes, this specification guarantees that if β_1 is adjoined to γ as specified, β_{11} and β_{12} will be adjoined at n_1 and n_2 simultaneously. The convention that the constraints on the auxiliary tree replace those that appear at the nodes where adjoining takes place remains as before.

Linking can also be defined for any two nodes belonging to the trees of the same auxiliary set. When two nodes are in the same tree, linking reduces to the case already discussed. When the two nodes are in different trees, the *c-command* condition is required to hold between the linked nodes after the auxiliary set is adjoined to the elementary tree. Thus, if there is a link across two components β_{11} and β_{12} of an auxiliary set β_1 with say, the link mother in β_{11} and the link daughter in β_{12} , then the required *c-command* condition for the link will have to be satisfied by virtue of an appropriate dominance relation between the nodes of the tree into which β_1 is adjoined. If β_{11} is adjoined at node n_1 and β_{12} at node n_2 , for example, then n_2 must dominate n_1 . Linking will be preserved under adjunction as before.

Finally, it is possible to extend the notion of derivation structure to the case of TAG with auxiliary sets, but we shall not describe that extension here. We simply wish to emphasize in this connection that multi-component adjunction is defined over an auxiliary set, an elementary tree, and a set of addresses in that elementary tree. In other words, the distinct nodes to which the different components of an auxiliary set adjoin always belong to the same elementary tree.

Multicomponent adjoining has been used by Kroch in his paper in this volume. Kroch and Joshi (1986) have also used multicomponent adjoining in their analysis of extraposition. In both cases the multicomponent adjoining has the locality of the first kind.

REFERENCES

- Joshi, A. 1985. "How much context-sensitivity is required to provide reasonable structural descriptions: tree adjoining grammars." In D. Dowty, L. Karttunen, and A. Zwicky, eds. *Natural Language Processing: Psycholinguistic, Computational And Theoretical Perspectives*. New York: Cambridge University Press. (Originally presented in May 1983 at the Workshop on Natural Language Parsing at the Ohio State University.)
- Joshi, A., L. Levy, and M. Takahashi. 1975. "Tree adjunct grammars". *Journal of the Computer and System Sciences*, 10:1 pp. 136-163.
- Joshi, A.K., Vijay-Shanker, K., and Weir, D. 1986. "Tree Adjoining grammars and head grammar", Technical Report MS-CIS-86-01. Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.
- Higginbotham, J. 1984 "English is not a context-free language". *Linguistic Inquiry* 15:2 pp. 225-235.
- Kroch, A. and Joshi, A.K. 1985. Linguistic significance of tree adjoining grammars, to appear in *Linguistics and Philosophy*, 1986.
- Kroch, A. and Joshi, A.K., 1986. "Analyzing extraposition in a tree adjoining grammar". To appear in *Syntax and Semantics (Discontinuous Constituents)*, (Eds. G. Huck and A. Ojeda), Academic Press 1986.
- Peters, S. and R. Ritchie. 1982. "Phrase linking grammars". Unpublished paper University of Texas.
- Pollard, C., 1984. "Generalized Phrase Structure Grammars, Head Grammars, and Natural language", *Ph.D dissertation, Stanford University*.
- Shieber, S. 1984. "Evidence against context-freeness of natural language". To appear in *Linguistics and Philosophy* 1986.
- Vijay-Shanker, K. and Joshi, A.K. 1985. "Some computationally significant properties of tree adjoining grammars", in *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*.
- Vijay-Shanker, K., Weir, D., and Joshi, A.K. 1986. "Tree Adjoining and head wrapping", in *Proceedings of the International Conference on Computational Linguistics (COLING)* Bonn, August 1986.
- Vijay-Shanker, K. 1986. "A study of tree adjoining grammars", Ph.D. Dissertation Proposal,

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

Weir, D., Vijay-Shanker, K., and Joshi, A.K. 1986. "The relationship of tree adjoining grammars and head grammars", in *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, New York, June 1986.