Natural Language Parsing, (Eds., D. Dowty, . Karttuhen, and A. Zwicky), Cambridge University Press, 1985.

MS-CIS-85-23

6 Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?

ARAVIND K. JOSHI

Since the late 1970s there has been vigorous activity in constructing highly constrained grammatical systems by eliminating the transformational component either totally or partially. There is increasing recognition of the fact that the entire range of dependencies that transformational grammars in their various incarnations have tried to account for can be captured satisfactorily by classes of rules that are *nontransformational* and at the same time highly constrained in terms of the classes of grammars and languages they define.

Two types of dependencies are especially important: subcategorization and filler-gap dependencies. Moreover, these dependencies can be unbounded. One of the motivations for transformations was to account for unbounded dependencies. The so-called nontransformational grammars account for the unbounded dependencies in different ways. In a tree adjoining grammar (TAG) unboundedness is achieved by factoring the dependencies and recursion in a novel and linguistically interesting manner. All dependencies are defined on a finite set of basic structures (trees), which are bounded. Unboundedness is then a corollary of a particular composition operation called *adjoining*. There are thus no unbounded dependencies in a sense.

This factoring of recursion and dependencies is in contrast to transformational grammars (TG), where recursion is defined in the base and the transformations essentially carry out the checking of the dependencies. The phrase linking grammars (PLGs) (Peters and Ritchie, 1982) and the lexical functional grammars (LFGs) (Kaplan and Bresnan, 1983) share this aspect of TGs; that is, recursion builds up a set a structures, some of which are then filtered out by transformations in a TG, by the constraints on linking in a PLG, and by the constraints introduced via the functional structures in an LFG. In a generalized phrase structure grammar (GPSG) (Gazdar, 1982), on the other hand, recursion and the checking of the dependencies in a sense go together. In a TAG, dependencies are defined initially on bounded structures and recursion simply preserves them.

TAGs have the following important properties: (1) We can represent the usual transformational relations more or less directly in TAGs; (2) the power of TAGs is only slightly more than that of context-free grammars (CFGs) in what appears to be just the right way; and (3) TAGs are powerful enough to characterize dependencies (e.g., subcategorization, as in verb subcategorization, and filler-gap dependencies, as in the case of moved constituents in *wh*-questions), which might be at unbounded distance and nested or crossed. It should be noted that the extra power of TAGs (beyond that of CFGs) is not due to some ad hoc modification of the context-free rewriting rule, but rather it is a direct consequence of factoring recursion and dependencies in a special way.

In the next section TAGs are defined and some of their properties are stated. TAGs with *links* are introduced later, and then TAGs with *local constraints*. Some of the the formal properties of TAGs, GPSGs, PLGs, and LFGs are compared with respect to three issues: the types of languages (reflecting different patterns of dependencies) that can or cannot be generated by the different grammars, a certain growth property, and parsing complexity. After some detailed linguistic examples illustrating the use of TAGs, some problems that need further investigation are listed.

In this paper excessive notation and formal proofs have been avoided for several reasons: (1) some of the notation and proofs have already appeared (see Joshi, Levy, and Takahashi, 1975); (2) detailed notation is not necessary to get across the main ideas of this paper; (3) some of the new results can be obtained using the formalism set up by Joshi, Levy, and Takahashi (1975); and (4) the main purposes of the paper are to examine the structure of TAGs and the structural descriptions they can support and to evaluate their linguistic adequacy. In summary, TAGs provide significant insight into the problem of identifying the necessary and sufficient power for a grammar to characterize adequately natural language structures.

6.1. Tree adjoining grammars

I will introduce tree adjoining grammar (TAG) by first describing an alternative way of looking at the derivation of the strings and the corresponding derivation trees of a context-free grammar (CFG). Later I will introduce TAGs in their own right. TAGs are more powerful than CFGs both weakly and strongly. Grammars *G1* and *G2* are *weakly equivalent* if the string language of *G1*, L(G1), is identical to the string language of *G2*, *L*(*G2*). *G1* and *G2* are *strongly equivalent* if they are weakly equivalent and for each *w* in *L*(*G1*) = *L*(*G2*), *G1* and *G2* assign the same structural description to *w*.

A grammar G is *weakly adequate* for a (string) language *L* if *L*(*G*) = *L*. G is *strongly adequate* for *L* if *L*(*G*) = *L* and for each *w* in *L*, *G* assigns an "appropriate" structural description to w. The notion of strong adequacy is undoubtedly not precise because it depends on the notion of appropriate structural descriptions.

Example 6.1.1. Let G' be a context-free grammar with the following productions.

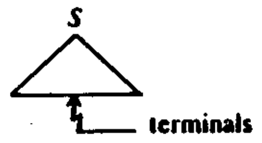
$S \rightarrow a\ T\ b$
$T \rightarrow S\ a$
$T \rightarrow T T$
$T \rightarrow a\ b$

S is the start symbol, S and T are the nonterminals, and a and b are the terminal symbols.

We can now define a *tree adjoining grammar* (*TAG*) G that is both weakly and strongly equivalent to G'. Let $G = (I,A)$, 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*.

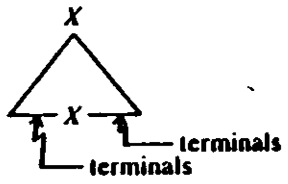
A tree α is an initial tree if it is of the form in (1).

(1) α =



That is, the root node of α is labeled S and the frontier nodes are all terminal symbols. The internal nodes are nonterminals. A tree β is an auxiliary tree if it is of the form in (2).

(2) β =

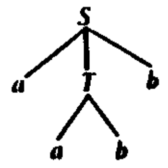


That is, the root node of β is labeled X, where X is a nonterminal and the frontier nodes are all terminals except one labeled X, the same label as that of the root. The node labeled by X on the frontier will be called the *foot* node of β. The internal nodes are nonterminals.

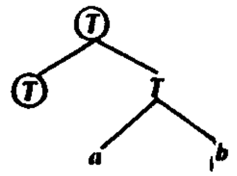
As defined, the initial and the auxiliary trees are not tightly constrained. The idea, however, is that both the initial and the auxiliary trees will be *minimal* in some sense. An initial tree will correspond to a *minimal* sentential tree (i.e., without recursion on any nonterminal) and an auxiliary tree, with root and foot node labeled X, will correspond to a *minimal* recursive structure that must be brought into the derivation, if there is recursion on X.

For the grammar in Example 6.1.1, we can define a TAG, $G = (I,$
as in (3).

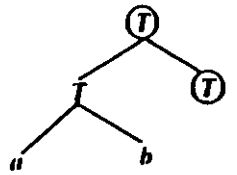
(3) I: α_1 =



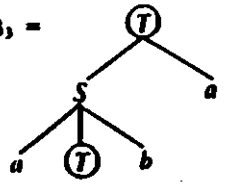
A: β_1 =



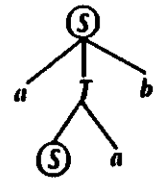
β_2 =



β_3 =



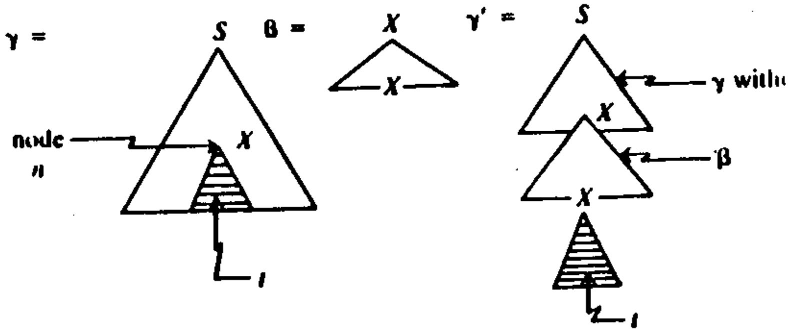
β_4 =



The root node and the foot node of each auxiliary tree are circled for convenience.

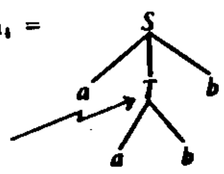
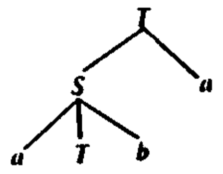
We will now define a composition operation called *adjoining* (or *ad- junction*), which composes an auxiliary tree β with a tree γ. Let γ be a tree with a node labeled X and let β be an auxiliary tree with the root labeled X also. (Note that β must have, by definition, a node – and only one – labeled X on the frontier.) Adjoining can now be defined as follows. If β is adjoined to γ at the node n, then the resulting tree γ_1' is shown in (4).

(4) γ =



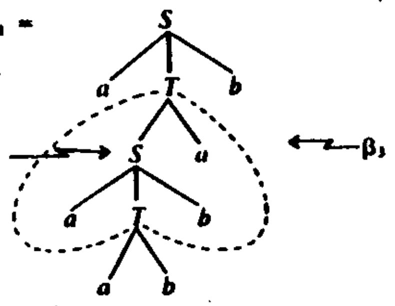
The tree t dominated by X in γ is excised, β is inserted at the node n in γ and the tree t is attached to the foot node (labeled X) of β; that is, is inserted or *adjoined* to the node *n* in γ, pushing *t* downward. Note that adjoining is not a substitution operation.

Let us now look at some derivations in the TAG, $G = (I,A)$ of Example 6.1.1.

(5) Let $\gamma_0 = \alpha_1 =$  $\beta = \beta_3 =$ 

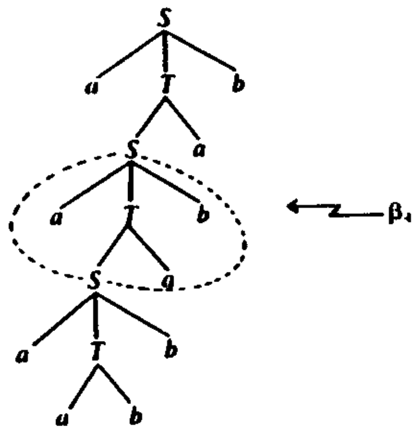
β_3 will be adjoined to γ_0 at T as indicated in γ_0. The resulting tree γ_1 is then as in (6).

(6) $\gamma_1 =$



We can continue the derivation by adjoining, say, β_4, at *S* as indicated in γ_1. The resulting tree γ_2 is then as in (7).

(7) $\gamma_2 =$



Note that γ_0 is an initial tree, a sentential tree. The derived trees γ_1 and γ_2 are also sentential trees. It is clear in this example that the TAG *G* will derive all and only the sentential trees of the CFG *G′*, starting from the initial tree of *G*. Thus *G* will also generate the string language *L*(*G′*) of *G′*.

We have introduced the TAG, *G*, in Example 6.1.1 with reference to the context-free grammar *G′*. We will now consider the TAGs in their own right. That is, *a TAG G = (I,A) will be a grammar with a finite set of initial trees, a finite set of auxiliary trees, and the adjoining operation as defined before*. We will now define *T*(*G*) and *L*(*G*).

Definition 6.1.1. *T*(*G*) is the set of all trees derived in G starting from initial trees in *I*. This set will be called the *tree set* of *G*.

Definition 6.1.2. *L*(*G*) is the set of all terminal strings of the trees in *T*(*G*). This set will be called the *string language* (*or language*) of *G*.

The relationship between TAGs, context-free grammars, and the corresponding string languages can be summarized as follows (Joshi, Levy, and Takahashi, 1975).

Theorem 6.1.1. For every context-free grammar *G′* there is TAG *G′* equivalent to *G*, both weakly and strongly.

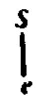
In Example 6.1.1, *G* is strongly (and therefore weakly) equivalent to *G′*. It can be shown also that the equivalent TAG *G* can be obtained effectively.

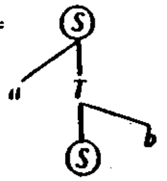
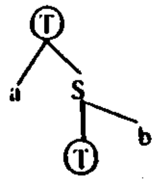
Theorem 6.1.2. Each of the following statements holds of some TAG, *G*.

(a) there is a context-free grammar *G′* that is both weakly and strongly equivalent to *G*;
(b) there is a context-free grammar *G′* that is weakly equivalent to *G* but not strongly equivalent to *G*;
(c) there is no context-free grammar *G′* that is weakly equivalent to *G*.

Parts (a) and (c) of Theorem 6.1.2 appear in Joshi, Levy, and Takahashi, 1975. Part (b) is implicit in that paper, but it is important to state it explicitly as done here. Example 6.1.1 illustrates part (a). Parts (b) and (c) will now be illustrated.

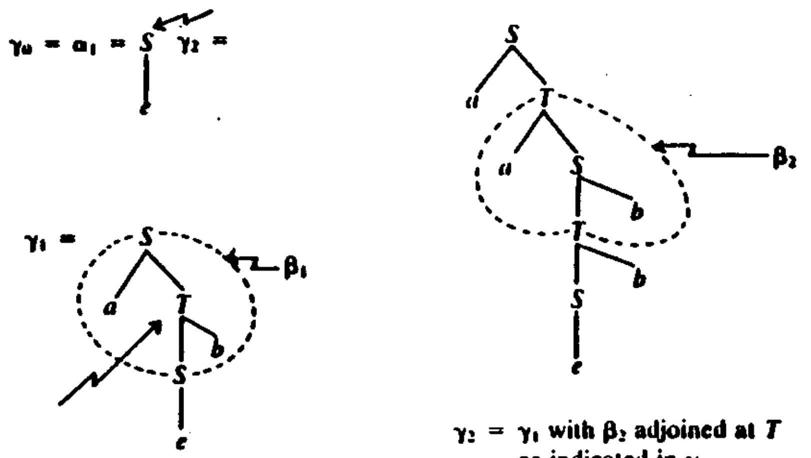
Example 6.1.2. Let $G = (I,A)$, where

(8) I: $\alpha_1 =$ 

A: $\beta_1 =$  $\beta_2 =$ 

Let us look at some derivations in G.

(9)



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

$\gamma_2 = \gamma_1$ with β_2 adjoined at T as indicated in γ_2.

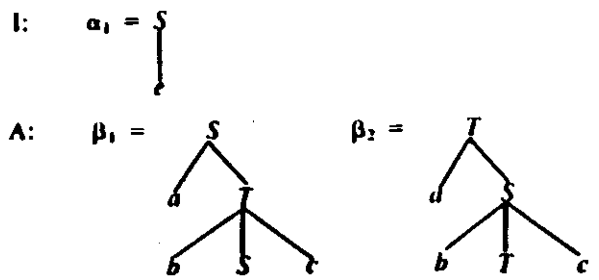
Clearly, L(G), the string language of G, is

(10) $L = \{a^n e b^n \mid n \geq 0\}$

which is a context-free language. Thus there must exist a context-free grammar G' that is at least weakly equivalent to G. It can be shown, however, that there is no context-free grammar G' that is strongly equivalent to G; that is, $T(G) = T(G')$. This follows from the fact that the set $T(G)$ (the tree set of G) is *unrecognizable*; that is, no finite state bottom-up tree automaton can recognize precisely $T(G)$ (see Bresnan et al., 1982; sec. 4). *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 6.1.3. Let $G = (I,A)$, where

(11)



The string language of G, $L(G)$, can be characterized as follows. We start with the language (which is a CFL)

(12) $L = \{ (a\,b)^n e\, c^n \mid n \geq 0 \}$.

$L(G)$ is then obtained by taking strings in L and moving (dislocating) some a's to the left. The precise definition of $L(G)$ is as follows:

(13) $L(G) = L_1 \{ w\, e\, c^n \mid n \geq 0$, w is a string of a's and b's such that
(i) the number of a's = the number of b's = n, and
(ii) for any initial substring of w, the number of a's $\geq$ the number of b's.$\}$

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

(14) $L_2 = \{ a^n b^n e\, c^n \mid n \geq 0 \} = L_1 \cap a^*\, b^*\, e\, c^*$.

L_2 is a 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 6.1.3 thus illustrates pa (c) of Theorem 6.1.2. (In example (14), if the dislocated a's are all moved to the left of all b's, then we obtain another strictly context-sensitive language (Peters and Ritchie, 1982). See section 6.3 for a TAG with local constraints for this language. This language can be generated by the phrase linking grammar of Peters and Ritchie. See section 6.3 for further details.)

TAGs have more power than CFGs; however, the extra power is quite limited. Both the qualitative and quantitative characterization of this lim itation will be discussed in detail in section 6.3. The language L_1 has 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 all a's come before all b's. TAGs are not powerful enough to generate L_2. This can be seen a 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 mus 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 o Example 6.1.3, that it will not be possible to avoid getting the a's and b' mixed. (It will be shown subsequently how L_2 can be generated by a TA with local constraints, but in a rather special way.) The so-called cop language

(15) $L_3 = \{ w\, e\, w \mid w \in \{a,b\}^* \}$

also cannot be generated by a TAG (although it can be generated by TA with local constraints). The reason for this is somewhat similar to th for L_2, but it is not so obvious. It is thus clear that TAGs can genera more than context-free languages but cannot generate all context-sensiti languages.

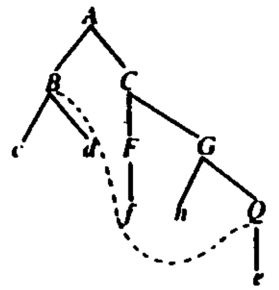
Theorem 6.1.3. (Joshi, Levy, and Takahashi, 1975)

CFL $\subsetneq$ TAL $\subsetneq$ Indexed Languages $\subsetneq$ CSL where CFL, TAL, and CSL are the classes of context-free, tree adjoining, and context-sensitive languages, respectively.

Indexed languages correspond to the indexed grammars (Aho, 1969). The fact that TAGs cannot generate L_2 and L_3 is important, because it shows that TAGs are only slightly more powerful than context-free grammars. The way TAGs acquire this power is linguistically significant and will be commented upon later. With some (linguistically motivated) modifications of TAGs, or rather the operation of adjoining, it is possible to generate L_2 and L_3, but only in some special ways. Thus L_2 and L_3 in some ways characterize the limiting cases of context sensitivity that can be achieved by TAGs and their slight extensions.

6.2. TAGs with "links"

The elementary trees (initial and auxiliary trees) are the appropriate domains for characterizing certain dependencies (e.g., subcategorization dependencies and filler-gap dependencies). This characterization is achieved by introducing a special relationship between certain specified pairs of nodes of an elementary tree. This relationship is pictorially exhibited by an arc (a dotted line) from one node to the other. For example, in tree (16), the nodes labeled B and Q are linked.

(16)


We will require the following conditions to hold for a link in an elementary tree.

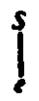
If a node n_1 is linked to a node n_2 then
(i) n_2 c-commands n_1 (i.e., n_2 precedes n_1 and there exists a node m that immediately dominates n_2 and also dominates n_1).
(ii) n_1 dominates a null string (represented as a terminal symbol in the non-linguistic formal grammar examples).

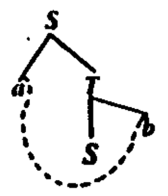
Linking is thus an asymmetric relation. In the linguistic context both n_1 and n_2 will be of the same category and only n_1 will dominate a null string.

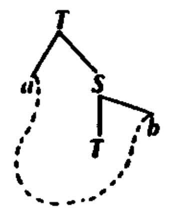
A TAG with links is a TAG where some of the elementary trees may have links as defined before. Henceforth, a TAG with links will often be called just a TAG.

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 6.2.1 will illustrate this point.

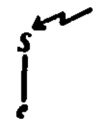
Example 6.2.1. Let $G = (I, A)$, where

(17) I: α_1 =


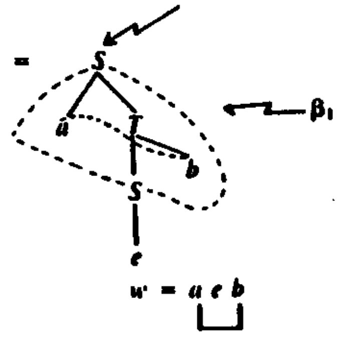
A: β_1 =


β_2 =


Let

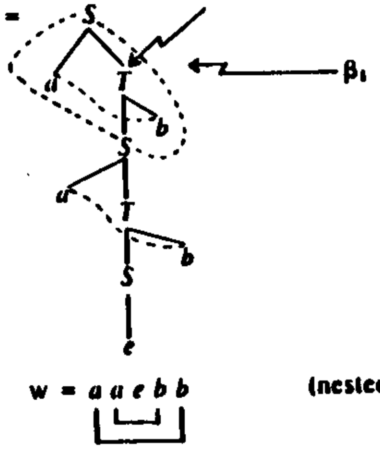
(18) $\gamma_0 = \alpha_1$ =


Adjoining β_1 at S as indicated in γ_0, we have

(19) γ_1 =


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

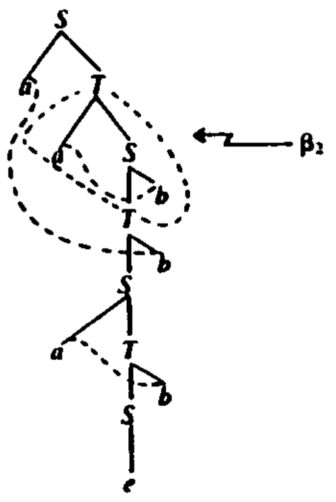
Adjoining β_1 again at S as indicated in γ_2, we have

(20) γ_2 =


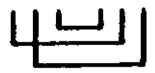
(nested dependencies)

Adjoining β_2 at T as indicated in γ_2, we have

(21) γ_3 =



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



β_1 and β_2 each have one link. γ_2 and γ_3 show how the linking is preserved in adjoining. In γ_3 one of the links is stretched. It should be clear now how, in general, the links will be preserved during the derivation. I will not give a formal definition here.

Also note in this example that in γ_2 the dependencies 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 and it is crossed with respect to the nested ones (this, of course, is not a unique description). The two elementary trees β_1 and β_2 have only one link each. The nestings and crossings in γ_2 and γ_3 are the result of adjoining. There are two points to note here.

1. TAGs 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 representing 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 Example 6.2.1 that the string language of TAG with links is not affected by the links; that is, we have

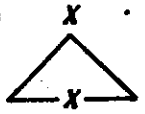
Theorem 6.2.1. Let G be a TAG with links. Then $L(G) = L(G')$, where G' is a TAG that is obtained from G by removing all the links in the elementary trees of G.

Thus links do not affect the weak generative capacity. However, they make certain aspects of the structural description explicit, which is implicit in the TAG without links. Thus the trees derived in G of Example 6.2.1 show the dependencies explicitly. The trees derived in G' (i.e., G with the links removed) also have these dependencies, but they are implicit. In the following section, the use of links is illustrated in the context of a linguistic example.

6.3. TAGs with constraints on adjoining

The adjoining operation as defined in section 5.1 is context free. An auxiliary tree, say,

(22) β =



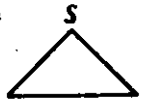
is adjoinable to a tree t at a node, say, n, if the label of that node is X; the adjoining does not depend on the context (tree context) around the node n. In this sense, adjoining is context free.

We will now consider certain types of constraints that must be checked in order that an auxiliary tree is adjoinable at a node n. These constraints are similar to those called *local constraints* (Joshi and Levy, 1978). These constraints are a generalization of the context-sensitive constraints studied by Peters and Ritchie, 1969.

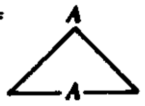
A TAG with local constraints is a TAG $G = (I, A)$, where I is the set of initial trees and A is the set of auxiliary trees and for each auxiliary tree there is a local constraint (possibly null). Rather than define local constraints precisely (for a detailed definition see Joshi and Levy, 1978), I will give some examples of TAGs with local constraints, which should be adequate to convey the main idea. First, consider a rather general example.

Example 6.3.1. Let $G = (I, A)$, where

(23) I: α_1 =



A: β_1 =



$:LR(\phi_1 - \psi_1) \wedge TB(\phi_2 - \psi_2)$

For the auxiliary tree β_1 there is a local constraint specified to the right of β_1. This constraint can be treated as a predicate that must be true of a node, say, n, of a tree t in order that β_1 is adjoinable to t at n. In our example the predicate $LR(\phi_1 - \psi_1)$ is a *proper analysis* predicate that is true of the node n in tree t if there exists a proper analysis (a cut) of the tree t that passes through the node n and that is of the form

(24) $\rho_1\, \phi_1\, A\, \psi_1\, \rho_2$,

where ρ_1 and ρ_2 are arbitrary strings of terminals and nonterminal symbols, ϕ_1 and ψ_1 are some specified strings of terminals and nonterminals, and A is the label of the node n. What this means is that if the predicate $LR(\phi_1 - \psi_1)$ holds at n, then there is a *left* context ϕ_1 and a *right* context ψ_1, around the node n.

The predicate $TB(\phi_2 - \psi_2)$ is called a *domination predicate*, which is true of the node n in the tree t if there is a path from the root to the frontier of t passing through n, which is of the form

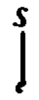
(25) $\rho_1\, \phi_2\, A\, \psi_2\, \rho_2$,

where ρ_1 and ρ_2 are arbitrary strings of terminals and nonterminals, ϕ_2 and ψ_2 are some specified strings of terminals and nonterminals, and A is the label of the node n. If the predicate $TB(\phi_2 - \psi_2)$ holds at n, this means that there is a *top* context ϕ_2 and a *bottom* context ψ_2 around the node n.

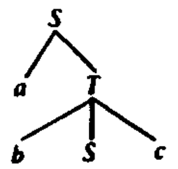
The set of trees $T(G)$ and the string language of G, $L(G)$ are defined in the same way as for a TAG without local constraints. In Example 6.3.1 we have only one auxiliary tree. In general, there will be more than one auxiliary tree and each tree will have a local constraint associated with it (possibly null). The local constraint associated with the auxiliary tree β_1 is a conjunction of a LR and a TB predicate. In general, a local constraint can be a Boolean combination of LR and TB predicates.

Example 6.3.2. Let $G(I,A)$, where

(26) I: α_1 =

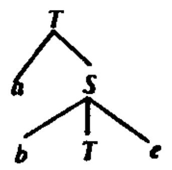


A: β_1 =



$(LR(a-\ ) \wedge TB(T-b)$
$\vee(\neg LR(b-\ ) \wedge TB(\ -e))$

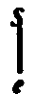
β_2 =



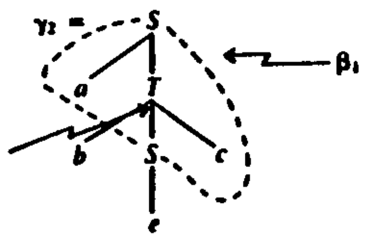
$:LR(a-\ ) \wedge TB\ (S-b)$

This TAG is the same as that in Example 6.1.3, except that β_1 and β_2 have local constraints associated with them. For example, in order for β_1 to be adjoinable to a node labeled S, we must have a left context a and a top context T and a bottom context b (or we must *not* have a left context b and must have a bottom context e; this part of the local constraint is to take care of the initial adjoining of β_1 to α_1). Similarly, for β_2 to be adjoinable to a node labeled T, we must have a left context a, a top context S, and a bottom context b. Some of the trees derived in G are

(27) $\gamma_1 = \alpha_1 =$

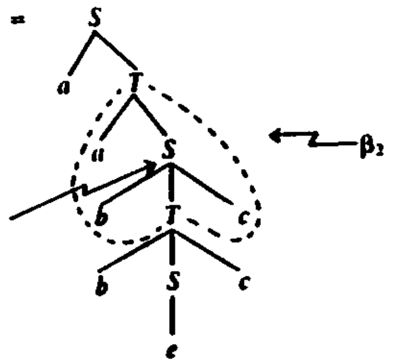


γ_2 =



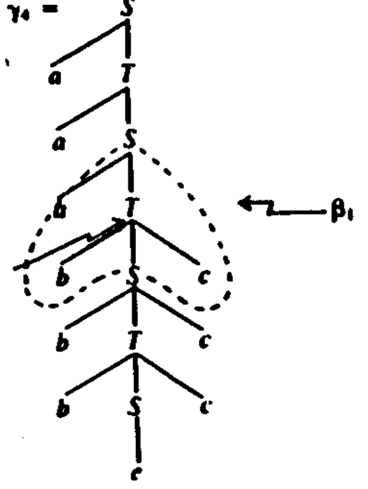
γ_2 is derived by adjoining β_1 to γ_1 at the indicated node S in γ_1. Note that β_1 cannot be adjoined now to the lowermost S node in γ_2 because the local constraint is not satisfied. Note also that β_1 cannot be adjoined to the top node S in γ_2.

(28) γ_3 =



γ_3 is derived from γ_2 by adjoining β_2 to the indicated node labeled T in γ_2. Note that β_1 can be adjoined to γ_3 only at the node S in the middle of γ_3, but not at the top S node or the bottom S node of γ_3 because the local constraint is not satisfied. Also β_2 cannot be adjoined to either one of the T nodes of γ_3.

(29) γ_4 =



γ_4 is derived by adjoining β_1 to the indicated S node in γ_3. The only adjunction that is possible for γ_4 is a β_2 adjoined at the indicated T node in γ_4.

It is clear that in this grammar at each stage of the derivation the only node that receives adjunction is the center node of the tree, which will be either S or T (compare the derivations in Example 6.1.3). Since the adjoining always takes place at the center node, the string language $L(G)$ of G is

(30) $L_2 = \{a^n b^n e c^n \mid n \geq 0\}$

(compare the string language in Example 6.1.3). L_2 cannot be generated by a TAG without local constraints, as we have seen in Example 6.1.3.

Similarly, the string language (copy language)

(31) $L_3 = \{w e w \mid w \in \{a,b\}^*\}$

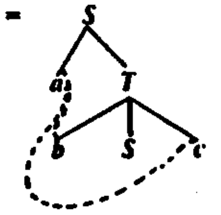
can be generated by a TAG with local constraints. (*Hint*: Use a TAG similar to the TAG in Example 6.1.2 and provide suitable local constraints.)

Thus TAGs with local constraints are more powerful than TAGs without local constraints. However, this extra power is very limited and it is much less than the full power of context-sensitive grammars, as will be shown later.

TAGs have three important properties, which restrict their generative power severely, but apparently in just the right way from the point of view of language structure.

First let us look the derivations in Example 6.3.2 somewhat more carefully. Let us call the a, b, and c in each auxiliary tree the *dependent* set of elements. Alternatively, we can assume that in the auxiliary tree, β_1, there is a link between b and a, and between c and a. Thus

(32) β_1 =

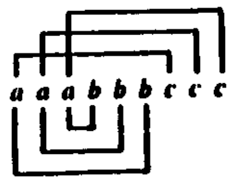


:(local constraint)

and similarly for β_2.

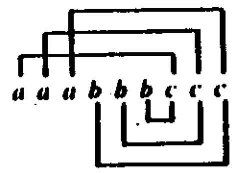
If we write the terminal string of γ_4 indicating the dependencies by the solid lines, we will have

(33)



Thus the a's and b's have nested dependencies and the a's and c's have cross-serial dependencies. If in β_1 we had a link between c and b, and between c and a, then the b's and c's would be nested and the a's and c's would be cross-serially dependent as shown below.

(34)

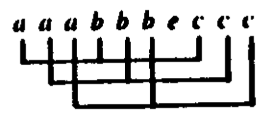


It should be clear that it will not be possible to construct a TAG with local constraints that will generate

(35) $L = \{a^n b^n e c^n \mid n \geq 0\}$,

where the dependencies between a's, b's and c's are all cross-serial; for example,

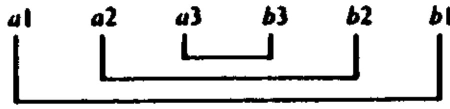
(36)



Thus although L_2 can be generated by a TAG with local constraints, the only permissible structure descriptions are of the form where the *a*'s and *b*'s (or the *b*'s and *c*'s) are nested and the *a*'s and *c*'s are cross-serially dependent, but not of the form where the *a*'s, *b*'s, and *c*'s are all cross-serially dependent. This property can be cast in a somewhat general form as a property of TAGs (with or without local constraints) as follows.

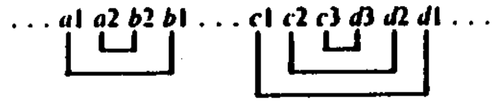
A context-free grammar allows characterizing dependencies between two sets when these dependencies are nested, as in

(37)



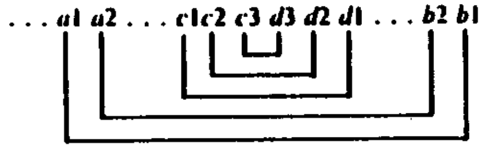
Further, there may be arbitrarily many such pairs of dependent sets; however, their dependencies do not cross. *Two pairs of dependent sets are either disjoint or else one pair is properly nested inside the other.* Thus we have either

(38)



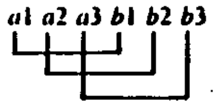
or

(39)



Similarly, TAGs can characterize arbitrarily many pairs of dependent sets, where the dependencies are nested, and two pairs of such dependent sets such that they are either disjoint or one is properly nested inside the other, just as in the case of context-free grammars. However, in the case of TAGs we can also have a pair of dependent sets where the dependencies are cross-serial, as in

(40)

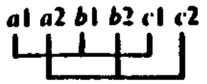


(Actually, we can have the dependencies mixed, i.e., some nested and some cross-serial. Here we will consider only cross-serial ones to keep the discussion simple. However, the statements below apply for this general case also.)

In a TAG we can have cross-serial dependencies between only

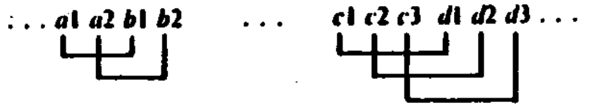
two dependent sets and not more than two; hence, we cannot represent the cross-serial dependencies as in

(41)



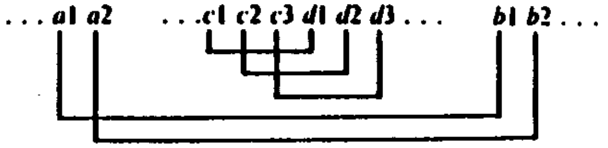
(involving three dependent sets). As long as the cross-serial dependencies involve only two dependent sets, as in the case of context-free grammars, we can have arbitrarily many such pairs of dependent sets, each with its cross-serial dependencies; however, any two pairs of such dependent sets are either disjoint or one is properly nested inside the other. Thus we have either

(42)



or

(43)



Thus TAGs allow a limited amount of cross-serial dependencies, and the dependent sets have the nesting properties as in the case of context-free grammars.

In the preceding discussion the dependent sets consisted of single letters, *a*'s and *b*'s for example. Since the substitution property holds for context-free languages (CFLs) and the languages of TAGs (TALs)(i.e., a CFL or a TAL continues to be a CFL or a TAL, respectively, if a terminal symbol is substituted by a CFL or a TAL, respectively), the elements of the dependent sets can be strings from CFLs or TALs. We will not consider this more complex situation; single letters are enough for our purpose.

As further examples, we note that the language

(44) $L_4 = \{a^n b^n e c^n d^n \mid n \geq 0\}$

can be generated by a TAG with local constraints but with the structural descriptions of the form where *a*'s and *b*'s are nested, *c*'s and *d*'s are nested, and the *a*'s and *c*'s are cross-serially dependent (alternatively, *a*'s and *d*'s are nested, *b*'s and *c*'s are nested, and the *a*'s and *c*'s are cross-serially dependent) but not of the form where the *a*'s, *b*'s, *c*'s, and *d*'s are all cross-serially dependent.

As shown earlier, two pairs of dependent sets with cross-serial dependencies are either disjoint or one is properly nested inside the other; hence, languages such as

(45) $L_5 = \{a^n\, b^n\, c^n\, e\, d^n\, f^n \mid n \geq 0\}$

and

(46) $L_6 = \{w\, e\, w\, e\, w \mid w \in \{a,b\}^*\}$ (double copy language)

cannot be generated by a TAG with local constraints. Both L_5 and L_6 are strictly context-sensitive languages.

6.3.2. Constant growth property

This property is connected with the so-called semilinear property, a property also possessed by context-free languages. It can be shown that languages of TAGs also have this property, but I will not give the proof here (Joshi and Yokomori, 1983). Rather, I will give an informal discussion in terms of the constant growth property.

In a TAG, at each step of the derivation, we have a sentential tree so that the terminal string therefore is a sentence. The derivation thus proceeds from a sentential tree to a sentential tree, and, therefore, from a sentence to a sentence. Let γ_{i+1} be derived from γ_i by adjoining β_j to γ_i. Then the terminal strings of γ_i and γ_{i+1}, say, w_i and w_{i+1}, are both sentences, and the length of w_{i+1} is equal to the length of w_i plus the length of the terminal string of β_j, say, w_j (not counting the single nonterminal symbol in the frontier of β_j), i.e.,

(47) $|w_{i+1}| = |w_i| + |w_j|$

where $|x|$ denotes the length of x. Thus the lengths of the terminal strings (which are sentences) increase by a constant (from a fixed set of constants corresponding to the lengths of the terminal strings of the auxiliary trees of the given TAG).

It is thus clear that for any string, w, of $L(G)$, we have

(48) $$|w| = |w_k| + a_1|w_1| + a_2|w_2| + \cdots + a_i|w_i| + \cdots + a_m|w_m| \qquad a_i \geq 0, \quad 1 \leq i \leq m$$

where w_k is the terminal string of some initial tree and w_i, $1 \leq i \leq m$, the terminal string of the ith auxiliary tree, assuming there are m auxiliary trees. Thus w is a linear combination of the length of the terminal string of some initial tree and the lengths of the terminal strings of the auxiliary trees.

The constant growth property severely restricts the class of languages generated by TAGs. Languages such as

(49) $L_7 = \{a^{2^n} \mid n \geq 1\}$

(50) $L_8 = \{a^{n^2} \mid n \geq 1\}$

are not languages of any TAG. They do not satisfy the constant growth property.

Tree adjoining languages (TALs) have the constant growth property, as we have just seen. Now if we consider a TAG with local constraints, it is also the case that the corresponding TAL has the constant growth property. *The local constraints filter out some strings, but those that remain still satisfy the constant growth property.*

It can thus be seen that TAGs (with or without local constraints) are only slightly more powerful than CFGs. This extra power is highly constrained, at least because of the two properties discussed before.

6.3.3. Polynomial parsing

TAGs also have the following property.

Polynomial parsing. TAGs can be parsed in time $O(n^4)$ (Joshi and Yokomori, 1983). Whether or not an $O(n^3)$ algorithm exists for TAGs is not known yet. Thus the parsing performance of TAGs is comparable to that of CFGs, possibly only slightly worse.

It should be noted that the extra power of TAGs (beyond that of CFGs) is not due to some ad hoc modification of the context-free rewriting rule, but, rather, it is the direct consequence of factoring recursion and the domains of dependencies in a particular manner, which is linguistically significant (see section 6.4 for linguistic examples). I would like to propose that the three properties

1. limited cross-serial dependencies,
2. constant growth, and
3. polynomial parsing

roughly characterize a class of grammars (and associated languages) that are only slightly more powerful than context-free grammars (context-free languages). I will call these *mildly context-sensitive grammars (languages)*, MCSGs (MCSLs). This is only a rough characterization because conditions 1 and 3 depend on the grammars, while condition 2 depends on the languages; further, condition 1 needs to be specified much more precisely than I have done so far. I now would like to claim that grammars that are both weakly and strongly adequate for natural language structures will be found in the class of MCSGs. TAGs are a specific instantiation of such a class of grammars. PLGs and TAGs are so different in their formulations that it has been difficult to compare them directly. However, on the basis of the work done so far (see examples in sections 6.1 and 6.2 and this section thus far; see also the examples in the remarks at the end of this section), I believe that PLGs and TAGs have nearly the same power; that is, they are both MCSGs. LFGs, on the other hand, have much more power than CFGs, TAGs and PLGs. Indexed languages ap-

Table 6.1. *Comparison of generalized phrase structure grammar (and context-free grammar), tree adjoining grammar, phrase linking grammar, and lexical functional grammar*

Languages	GPSG (and CFG)	TAG[a]	PLG	LFG
1. Language obtained by starting with $L = \{(ba)^n c^n \mid n \geq 1\}$ and then dislocating some a's to the left.	no	yes	yes	yes
2. Same as language 1 except that the dislocated a's are to the left of all b's.	no	yes	yes	yes
3. $L = \{w \mid w$ is string of equal number of a's, b's and c's but mixed in any order$\}$.	no	no(?)	yes	yes
4. $L = \{x\, c^n y \mid n \geq 1$, x,y are strings of a's and b's such that the number of a's in x and y = the number of b's in x and $y = n\}$.	no	no	yes	yes
5. Same as language 4 except that the length of x = length of y.	no	yes	no(?)	yes(?)
6. $L = \{w\, c^n \mid n \geq 1$, w is string of a's and b's and the number of a's in w = the number of b's in $w = n\}$.	no	yes	yes(?)	yes(?)
7. $L = \{a^n b^n c^n \mid n \geq 1\}$.	no	yes	no	yes
8. $L = \{a^n b^n c^n d^n \mid n \geq 1\}$.	no	yes	no	yes
9. $L = \{a^n b^n c^n d^n e^n \mid n \geq 1\}$.	no	no	no	yes
10. $L = \{w\, w \mid w$ is string of a's and b's$\}$ (copy language).	no	yes	no(?)	yes
11. $L = \{w\, w\, w \mid w$ is string of a's and b's$\}$ (double copy language).	no	no	no(?)	yes
12. $L = \{a^n c^m b^n d^m \mid m \geq 1, n \geq 1\}$.	no	no	no(?)	?
13. $L = \{a^n b^n c^p \mid n \geq 1, p \neq n\}$.	no	yes	?	yes(?)
14. $L = \{a^{2^n} \mid n \geq 1\}$.	no	no	no(?)	yes
15. $L = \{a^{n^2} \mid n \geq 1\}$.	no	no	no(?)	yes
16. Limited cross-serial dependencies.	no	yes	?	no(?)
17. Constant growth property	yes	yes	yes(?)	no
18. Polynomial parsing	yes	yes	?	no(?)

Note: ?: answer unknown to the author; yes(?): conjectured yes; no(?): conjectured no.
[a] With or without local constraints.

pear to be generable by LFGs (as communicated by Robert Berwick) and some nonindexed languages are also generable by LFGs (as communicated by Fernando Pereira). I believe that LFGs as formulated at present are far more powerful than required. It has not been shown, to the best of my knowledge, that this extra power of LFGs is really needed. Whether meaningful ways of constraining LFGs exist so that the corresponding grammars will be in MCSGs is an open problem. (See Table 6.1.)

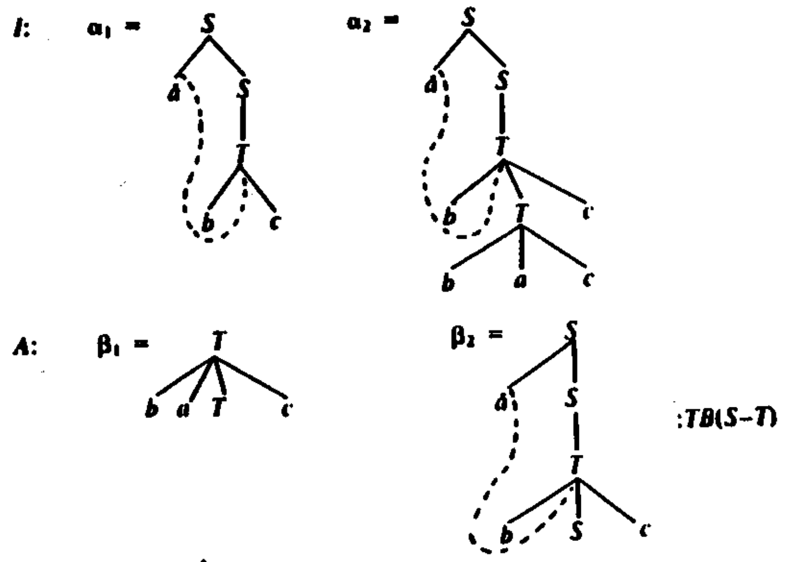
6.3.4. Formal remarks

Let L_9 be the language obtained from the language

(51) $\{(b\,a)^n\, c^n \mid n \geq 1\}$

by dislocating some number of a's and moving them to the left; all dislocated a's precede all b's. (This language is described in Peters and Ritchie, 1982, and can be generated by a phrase linking grammar.) Note that this language is different from that in Example 6.1.3, because here we require that all dislocated a's precede all b's. Let us now consider a TAG with local constraints that generates L.

Let $G = (I,A)$, where

(52)


This is not the simplest TAG for L_9. Note also that β_2 has a local constraint that requires a top context S and a bottom context T for β_2 to be adjoinable. The bar over some a's serves to indicate the dislocated a's. The TAG, G, above not only generates L, but also generates the appropriate linked tree sets of the corresponding PLG (see Peters and Ritchie, 1982). It is not clear yet whether a TAG without local constraints can be constructed for L_9; this is probably not possible.

Those familiar with PLGs may be interested in the following PLG for the language in Example 6.1.3.

(53) G': $S \rightarrow \bar{a}\, S$
$S \rightarrow \bar{a}\, b\, S\, c$
$S \rightarrow e$

The a's with bars have to be properly linked as defined by Peters and Ritchie (1982).

Example 6.3.4. The language L_{10} is of considerable interest. (This language was suggested by William Marsh (private communication) and some of his students. This language is also referred to as Bach language. Marsh has shown that this language can be generated by a PLG.)

(54) $L_{10} = \{w/w \in \{a,b,c\}^*$ and the number of a's = the number of b's = the number of c's$\}$.

That is, L_{10} has the same number of a's, b's, and c's, but the a's, b's and c's appear in all possible orders. L_{10} can be generated by a phrase linking grammar (PLG).

This language is interesting because, in a sense, it represents the extreme case of the degree of free word order permitted in a language. This extreme case is linguistically not relevant. Languages 1–6 in Table 6.1 represent different degrees of free word order, the language in 3 being the extreme case. Indicated in the table is the degree of free word order permitted by each type of grammar in terms of these languages. GPSGs cannot generate this language. TAGs also cannot generate this language, although for TAGs the proof is not in hand yet. LFGs can generate this language.

In a TAG, for each elementary tree, we can add more elementary trees, systematically generated from the given elementary trees to provide additional free word order (in a somewhat similar fashion to Pullum, 1982). Since the adjoining operation in a TAG gives some additional power to a TAG beyond that of a CFG, this device of augmenting the elementary trees should give more freedom, for example, by allowing some limited scrambling of an item outside of the constituent to which it belongs. Even then a TAG does not seem to be capable of generating the language in the preceding example (the language in 3 in Table 6.1). Thus there is extra freedom, but it is quite limited. The extra power of TAGs may be just adequate to handle the free word order phenomenon; however, we need to know much more about this phenomenon before we can be sure of the claim.

Table 6.1 lists (1) a set of languages reflecting different patterns of dependencies that can or cannot be generated by different types of grammars, and (2) the three properties of TAGs mentioned earlier.

6.4. Some linguistic examples

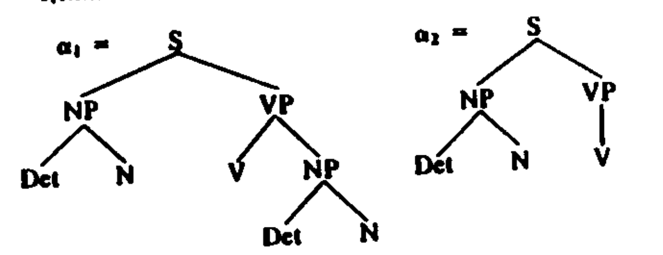
In this section, which gives some detailed linguistic examples of TAGs, many details that do not serve the purpose of illustrating the power of TAGs have been ignored or simplified. (For a more detailed account of linguistic relevance of TAGs, see Joshi and Kroch, 1984.)

6.4.1. An English example

Example 6.4.1. Let $G = (I,A)$, where I is the finite set of initial trees and A is the finite set of auxiliary trees. Only some of these trees in I and A will be listed,

especially those relevant to the derivations of certain sentences. Rather than troducing all the trees at once, I will introduce them a few at a time and some appropriate remarks as we go along. Later I will show some sample d vations.

(55) *I(Initial Trees):*



Note that α_1 corresponds to a "minimal sentence" with a transit verb, e.g.,

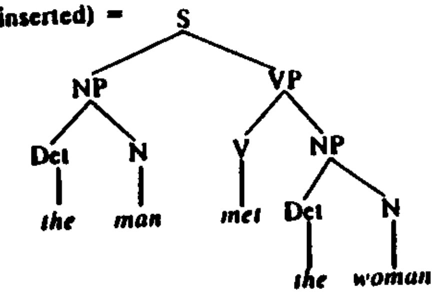
(56) The man met the woman.

and α_2 corresponds to a minimal sentence with an intransitive verb; example,

(57) The man fell.

The initial trees, as defined earlier, require terminal symbols on the fr tier. In the linguistic context, the nodes on the frontier will be pretermi symbols such as N, V, A, P, Det, etc. The lexical items are inserted each one of the preterminal symbols as each elementary tree enters derivation. Thus if we choose α_1 as the initial tree, then

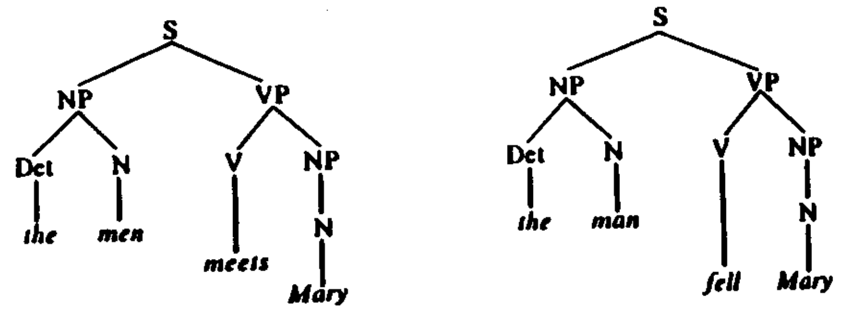
(58) $\gamma_1 = \alpha_1$ (with lexical items inserted) =



As we continue the derivation by selecting auxiliary trees and adjoi them appropriately, we follow the same convention; that is, as each iliary tree is chosen, we make the lexical insertions. *Thus in a deriva in a TAG, lexical insertion goes hand in hand with the derivation.* step in the derivation selects an elementary tree together with a s appropriate lexical items.

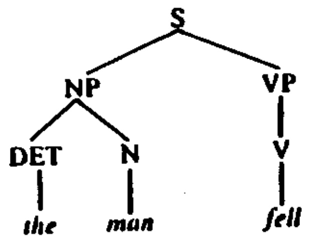
Note that as we select the lexical items for each elementary tree we can check a variety of constraints, e.g., agreement and subcategorization constraints on the set of lexical items. Thus, for example, the following choices of lexical items will not be permitted.

(59)



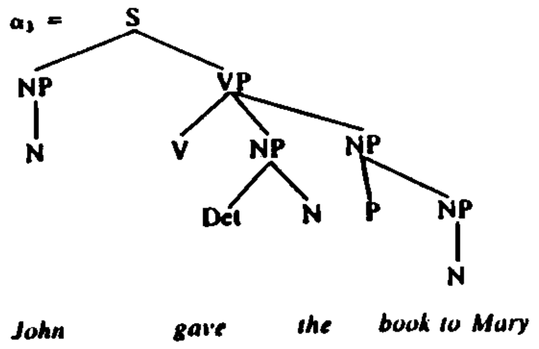
This is because, in the first case, the number agreement is violated and in the second case, *fell* does not take NP object. The point here is that these constraints can be easily checked because the entire elementary tree which is the domain of these constraints is available as a single unit at each step in the derivation. If we had started with α_2, then the choice of lexical items as shown in (60) would be permitted.

(60)



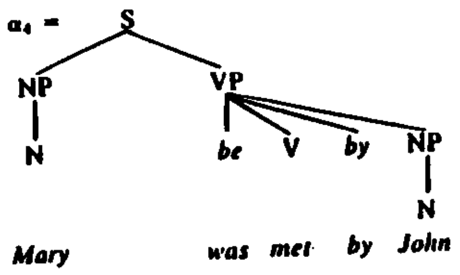
When an auxiliary tree enters the derivation, similar considerations hold. In addition, further constraints, both contextual and lexical, can be checked by means of *local constraints*. We will illustrate some of these later as we proceed with our example.

(61) α_3 =

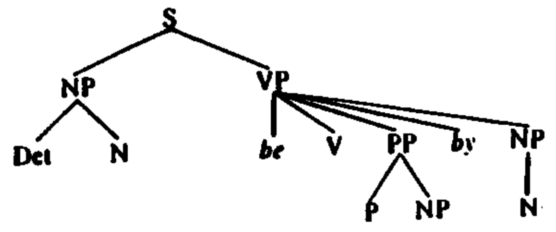


Note that henceforth a possible lexical choice will be stated by giving example below each tree. It is clear that for each subcategorization fra we will have an initial tree. We could, of course, represent this finite s of trees by some schema. This is only a matter of convenience and it not relevant to our current purpose.

(62) α_4 =



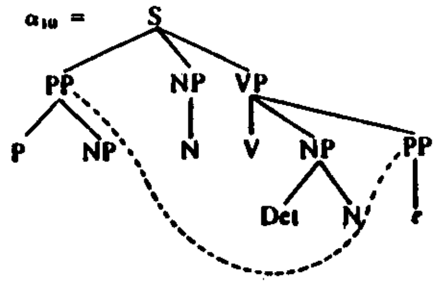
α_5 =



The book was given to Mary by John

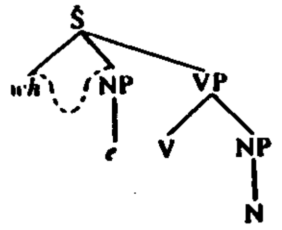
Note that α_4 and α_5 are the passive forms of α_1 and α_3, respectively have shown a flat structure for passive for convenience only. Nothing this section hangs on this particular structure for passive. (For furth details about the analysis of passive with respect to TAGs, see Joshi a Kroch, 1984.)

(63) α_{10} =



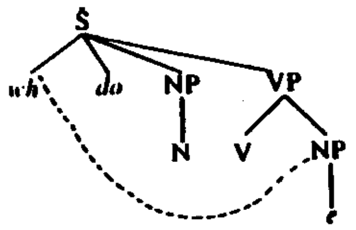
In α_{10}, a link is shown from the lower PP node to a higher PP node. It should be noted that when lexical items are inserted for the preterminal nodes in α_{10}, not only can we check that a verb requiring NP PP object has been inserted, but also that the preposition P is *to* as required by the verb, for example.

(64) α_{15} =



Who met Mary?

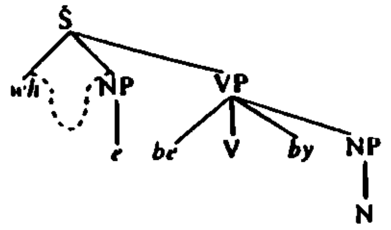
α_{16} =



Who did John meet?

⋮

α_{18} =



Who was met by John?

⋮

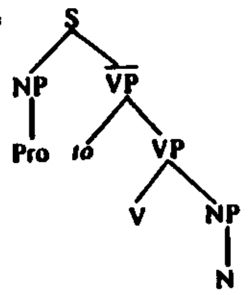
Each one of the finite sets *I* and *A* can be quite large. The set of initial trees contains so far all the *minimal* sentential trees corresponding to different subcategorization frames together with their *transforms*. It should be noted that this set is finite and these trees have been listed explicitly.

We could, of course, provide rules for obtaining some of these trees from a given subset of trees. These rules will achieve the effect of conventional transformational rules; however, they need not be formulated as the usual transformational rules. We can formulate them directly as

tree rewriting rules, especially since both the domains and the co-domains of the rules will be finite. *These rules will be abbreviatory in the sense that they will generate only finite sets of trees. Hence, incorporation of such rules will be only a matter of convenience and will not affect the TAG in any essential manner.*

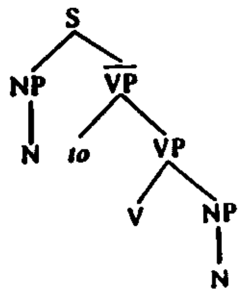
So far, all the initial trees defined correspond to minimal sentences. I will now introduce some initial trees that are minimal but are not matrix sentences. The motivation for introducing these trees will be clear from the examples and the subsequent use of these trees in the derivations. Some problems associated with the introduction of these trees will be discussed in the next section.

(65) α_{25} =



Pro to invite Mary

α'_{25} =



John to like Mary

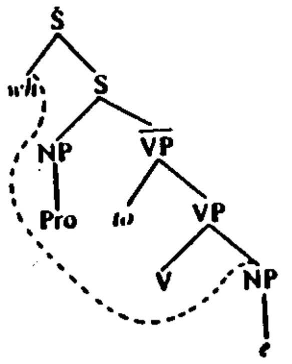
Note that α_{25} and α'_{25} are similar, except that in the first case the subject NP is realized as Pro and in the second case by a lexical item. α_{25} will be used in the derivation of sentences such as

(66) John persuaded Bill Pro to invite Mary.
John tried Pro to invite Mary.

α'_{25} will be used in deriving sentences such as

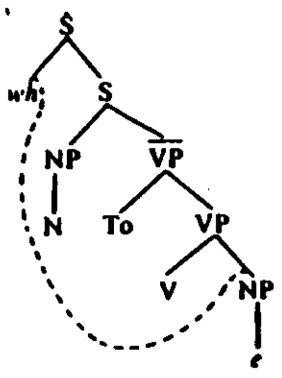
(67) John seems to like Mary

(68) α_{26} =



Who Pro to invite

α'_{26} =



α_{26} and α'_{26} differ in the same way α_{25} and α'_{25}. α_{26} will be used in deriving sentences such as

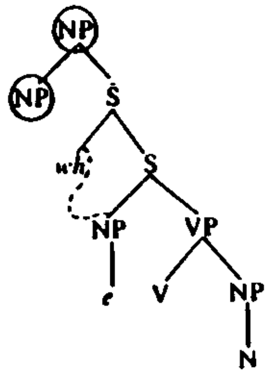
(69) Who did John try to invite?

α'_{26} will be used in deriving sentences such as

(70) Who did John expect Bill to invite?

So far we have considered some initial trees. Now let us examine auxiliary trees.

(71) An auxiliary tree β_1 =



who met Mary

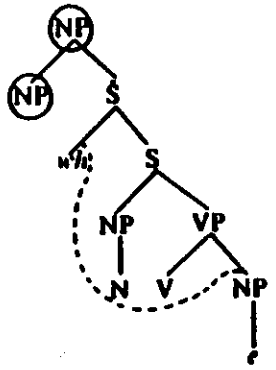
The terminal nodes of an auxiliary tree should all be terminal symbols, except one that is a nonterminal identical to the label of the root node. In the linguistic context, instead of terminals we will have preterminals on the frontier. (See the remarks on lexical insertion for initial trees.) In β_1 the circled NP nodes correspond to the root node and the foot node of an auxiliary tree. These nodes have been circled for convenience.

β_1 will be used to build a subject relative clause around an NP as in

(72) The boy who met Mary left.

The link in β_1 links the extracted NP node to *wh*.

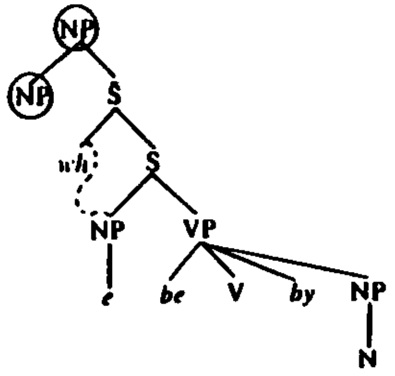
(73) β_2 =



who Mary met

β_2 corresponds to the object relative clause.

(74) β_3 =



who was met by Mary

β_3 corresponds to a subject relative clause where the verb is in the passive form. It is clear that we will have a large number of auxiliary trees; however, the set is finite. As in the case of initial trees, it is possible to write rules for obtaining some of the auxiliary trees, say for example, β_3 from β_1, or even all of the auxiliary trees, say for example, β_1 from α_1, etc. These *rules* will correspond more or less directly to the usual transformations. However, there are important differences: (1) The rules will be only abbreviatory in the sense that only a finite set of trees will be derivable from a finite set of trees. (2) The rules can be defined directly as tree rewriting rules, where both the domain and the co-domain are trees, unlike the transformational rules, which are mediated by structural descriptions based on proper analyses. *It is in this sense that the trees in I and A capture the usual transformational relations more or less directly.*

It should be pointed out that the so-called Island Constraints do not have to be stated as constraints in a TAG. They are simply corollaries of the TAG formulation. This observation is due to Tony Kroch (see Joshi and Kroch, 1984, for further details). Thus

(75) *To whom what did John do?

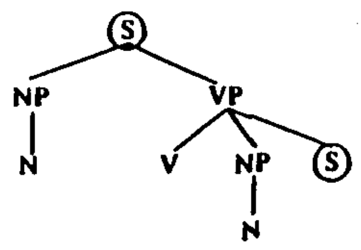
is disallowed if there is no elementary tree (an initial tree) corresponding to (75). Then (76) is automatically prevented

(76) *To whom did you wonder what John did?

The prevention of (75) is a matter of what elementary trees (initial trees, in this case) and what links are allowed on them. Once the elementary trees are defined, the links are *preserved* throughout the derivation in a TAG, as shown in section 6.3. No new linking relations are added between a tree and an auxiliary tree that is being adjoined. The so-called Island Constraints then follow as corollaries. Similar considerations hold for the Complex-NP Constraint.

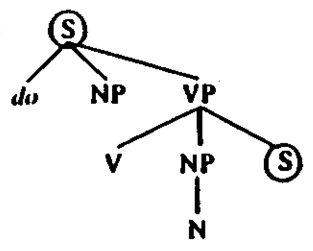
The *preservation* of the linking relations during the derivation in a TAG accounts for the so-called *unbounded movements*. In a sense, in a TAG, there are no unbounded movements. All movements are defined on the elementary trees; thus they are bounded. The unboundedness then is a

(77) β_6



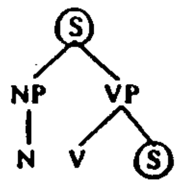
John persuaded Bill S

$\beta_6' =$



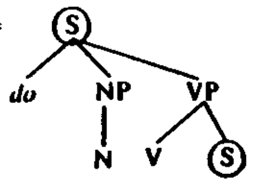
Did John persuade Bill S

(78) $\beta_7 =$



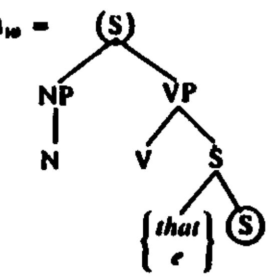
John expected S

$\beta_7' =$



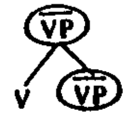
Did John expect S

(79) $\beta_{10} =$



John knew that S

$\beta_{15} =$



seems $\overline{\mathrm{VP}}$

corollary of the fact that the links are *preserved* during the derivation in a TAG.

β_6, β_6', β_7, β_7', and β_{10} correspond to the sentences involving sentential complements. For example,

(80) John persuaded Bill to invite Mary.

would be derived starting with an initial tree corresponding to

(81) Pro to invite Bill

to which β_6 is adjoined giving

(82) John persuaded Bill Pro to invite Mary.

β_6' would be used in deriving

(83) Who did John persuade Bill to invite?

β_7 and β_7' will be used in a similar fashion. β_{15} will be used in deriving

(84) John seems to like Mary.

Sentence (82) will not be derived in the same way as

(85) John tried to invite Mary.

So far I have not shown any local constraints for any one of the auxiliary trees. This was done for simplicity. Clearly, many of the auxiliary trees listed will be accompanied by local constraints. As I illustrate some of the derivations in the above TAG, I will point out some of the local constraints needed and how they can be stated for particular auxiliary trees. These examples should be adequate to show the use of the local constraints in a TAG.

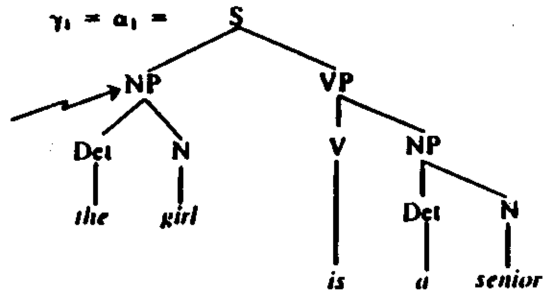
6.4.2. Derivations in the TAG, G = (I, A)

A derivation always begins with an initial tree. Sentences such as

(86) a. John met Mary.
b. The girl is a senior.
c. The rock fell.
d. John gave the book to Mary.
e. The book was given to Mary by John.
f. To Mary John gave the book.

correspond directly to initial trees. In particular, (84b) corresponds to α_1 (with appropriate lexical insertions).

(87) $\gamma_1 = \alpha_1 =$



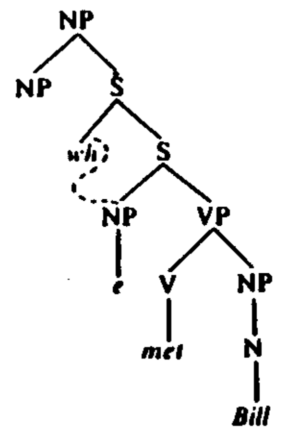
The girl is a senior.

I will now derive several sentences.

(88) The girl who met Bill is a senior.

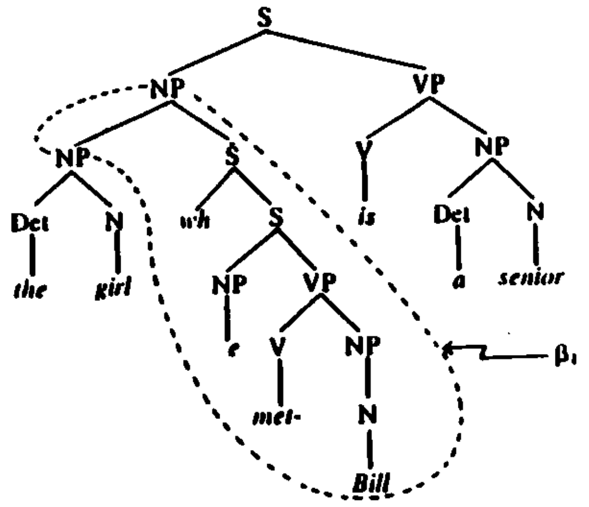
Let us take β_1 (with appropriate lexical insertions).

(89) $\beta_1 =$



β_1 is then adjoined to α_1 at the indicated node labeled NP resulting in γ_2.

(90) $\gamma_2 =$

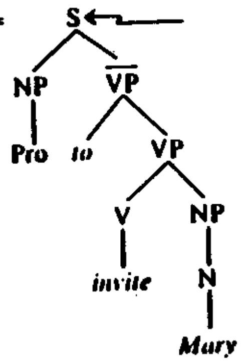


The girl who met Bill is a senior.

(91) John persuaded Bill to invite Mary.

We will start with the initial tree α_{25}.

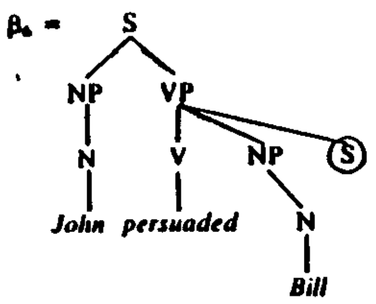
(92) $\gamma_1 = \alpha_{25} =$



Pro to invite Mary

Then we take β_6.

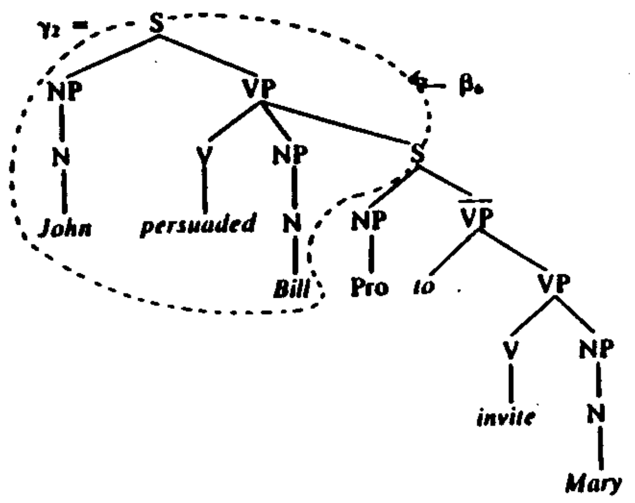
(93) β_6 =



John persuaded Bill S

β_6 is adjoined to γ_1 of the indicated node labeled S resulting in γ_2.

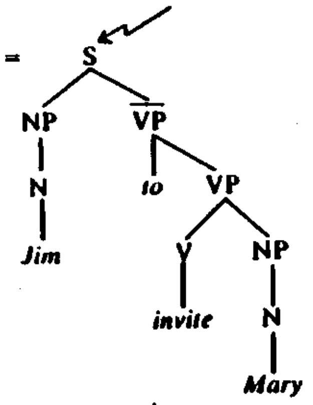
(94) γ_2 =



John persuaded Bill to invite Mary.

Note that if we start with α'_{25}, which is like α_{25} except that instead of Pro, we have a lexical NP

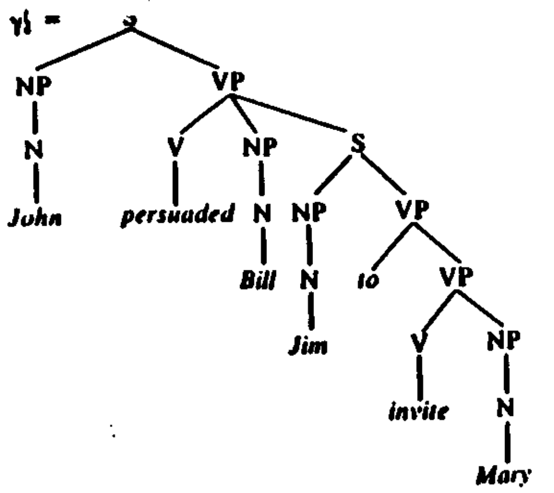
(95) $\gamma'_1 = \alpha'_{25}$ =



Jim to invite Mary

Adjoining β_6 to α'_{25} at the indicated node labeled S, we get γ'_2.

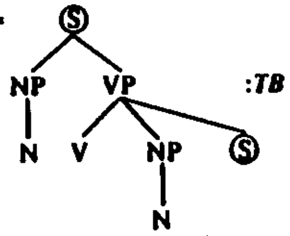
(96) γ'_2 =



* John persuaded Bill Jim to invite Mary.

γ'_2 can be disallowed if while adjoining β_6 to γ_1 (or γ'_1) we can check whether the subject NP is a Pro and allow adjoining only in that case. This can be achieved by associating a local constraint with β_6 as follows.

(97) β_6 =

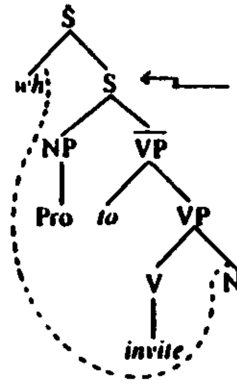


:*TB* (—NP Pro)

(98) Who did John persuade Bill to invite?

We will start with the initial tree α_{26}.

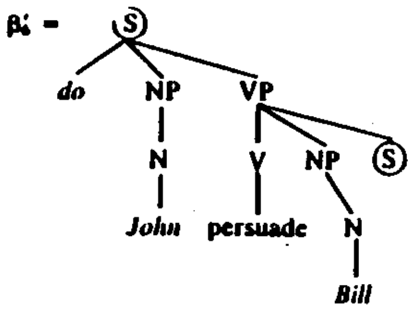
(99) $\gamma_1 = \alpha_{26}$ =



Who Pro to invite

Then we take β'_6.

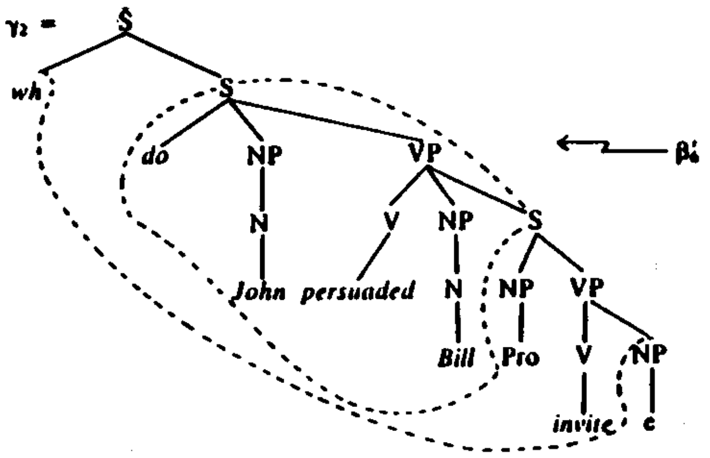
(100) β'_6 =



did John persuade Bill S

β_6 is adjoined to γ_1 at the indicated node labeled S, resulting in γ_2.

(101) γ_2 =



Who did John persuade Bill to invite

Note that the link in γ_1 is *preserved* in γ_2; it is *stretched* resulting in the so-called unbounded dependency.

It is now easy to see that starting from α'_{25} and then adjoining β_7 to α_{25} at the root node S, we obtain

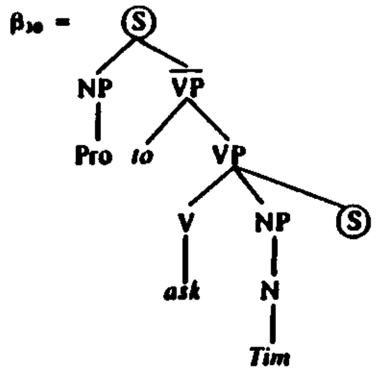
(102) John expected Bill to invite Mary.

Starting with α'_{26} and adjoining β'_7 to α'_{26} at the node labeled S, we obtain

(103) Who did John expect Bill to invite?

By setting up further auxiliary trees such as, for example,

(104) β_{30} =



we can obtain (105) as follows.

(105) John persuaded Bill to ask Tim to invite Mary.

We will start with α_{25} corresponding to

(105a) Pro to invite Mary.

Then we adjoin β_{30} to the S node in α_{25}, giving

(105b) Pro to ask Tim Pro to invite Mary.

Finally, adjoining β_6 to the root node S of the tree corresponding to (10
we obtain

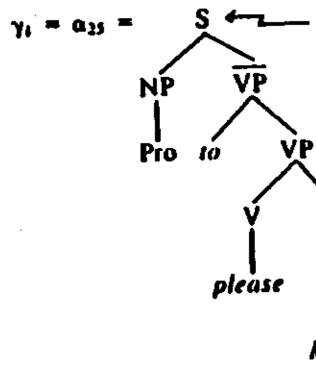
(105c) John persuaded Bill Pro to ask Tim Pro to invite Mary.

A very important aspect of TAGs is that *we can provide distinct d ivations for sentences containing the so-called equi and raising ve* This observation is due to Tony Kroch. (For further details, see J and Kroch, 1984.) Thus

(106) John tried to please Mary.

will be derived as follows. We will start with α_{25}

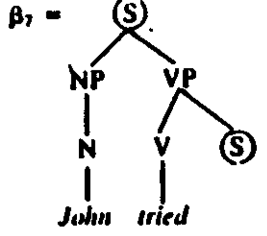
(107) $\gamma_1 = \alpha_{25}$ =



Pro to please Mary

We then take β_7.

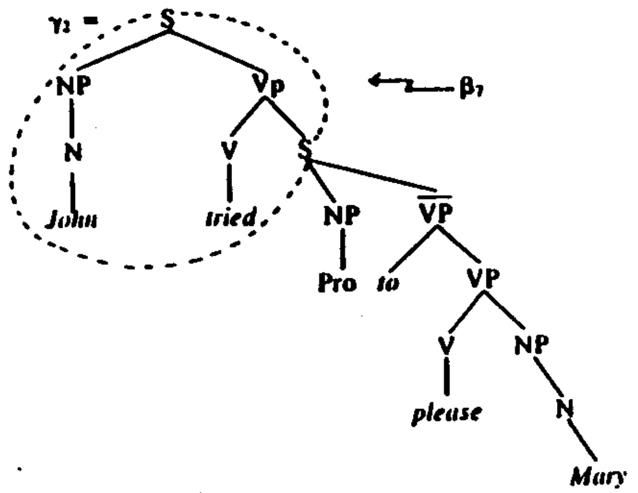
(108) β_7 =



John tried S

Adjoining β_7 to γ_2 at the indicated node label S, we obtain γ_2.

(109) γ_2 =



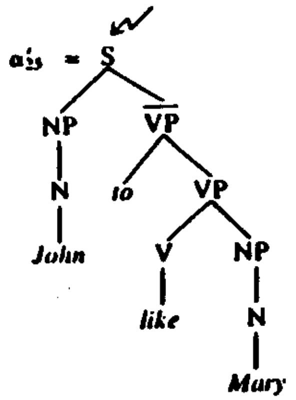
John tried Pro to please Mary.

On the other hand

(110) John seems to like Mary.

will be derived as follows. We will start with α'_{25}

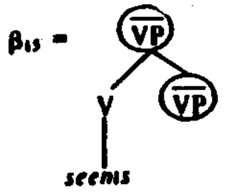
(111) γ_1 = α'_{25} =



John to like Mary

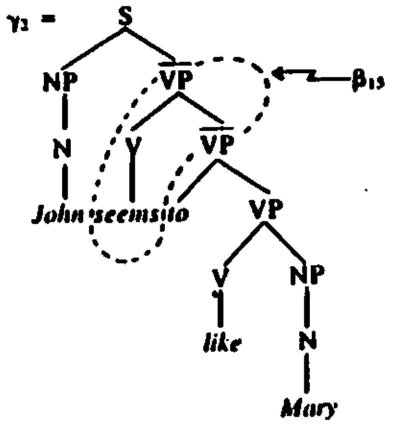
We then take β_{15}.

(112) β_{15} =



Adjoining β_{15} to γ_1 at this indicated node labeled VP, we obtain γ_2.

(113) γ_2 =



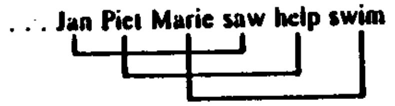
John seems to like Mary.

6.4.3. *Cross-serial dependencies in Dutch*

There are infinitely many sentences in Dutch that are of the following form (Bresnan et al., 1983).

(114) *. . . Jan Piet Marie zag helpen zwemmen*

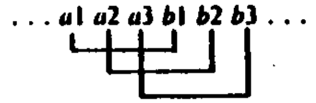
. . . Jan Piet Marie saw help swim



. . . Jan saw Piet help Marie swim

where the dependencies are as indicated by the solid lines. Thus we have strings of the form

(115) . . . *a*1 *a*2 *a*3 *b*1 *b*2 *b*3 . . .



The string language is of the form $\{a^n b^n \mid n \geq 1\}$, which is a context-free language, but the structural description required to capture the cross-serial dependencies cannot be achieved by a context-free grammar. I have already shown in the preceding test how a TAG can be constructed to provide structural descriptions corresponding to the cross-serial dependencies. The TAG in Examples 6.1.1 and 6.2.1 could therefore be adapted for characterizing dependencies in (114). However, a TAG in which the elementary trees correspond to

(116) (a) *Jan zag.*
(b) *Piet helpen.*
(c) *Marie zwemmen.*

with the derivation beginning with

(117) *Jan zag.*

will not do because, although the TAG will give the correct cross-serial dependencies, the resulting derivation will be linguistically defective and also will not be quite in the spirit of TAGs. It is clear that we must construct a TAG in which the derivation will begin with an initial tree corresponding to

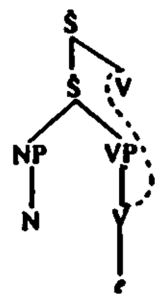
(118) *Marie zwemmen.*

and, of course, we must get the appropriate cross-serial dependencies. Bresnan et al. (1983) have stated certain facts about conjoining verbs and conjoining NP PP sequences, and they have proposed a structure to account for these facts. Their structure is characterized by the fact that the corresponding Ns and Vs branch out from two distinct paths from the root to the frontier. Such a structure can be regarded as having two spines, one to support the Ns and the other to support the Vs. (The unrecognizability of such tree sets follows directly, if we require that the Ns and the Vs match.) TAGs as defined so far will allow us to construct only structures with one spine. The TAG described below captures the cross-serial dependencies, and the derivations are in the spirit of TAGs. I must emphasize that the TAG given here is primarily for the purpose of illustrating how a TAG can be constructed to capture the cross-serial dependencies in the right manner, keeping the derivations in the spirit of TAGs. I do not wish to claim any detailed linguistic justification for the structure proposed here; however, the dependencies are correctly represented.

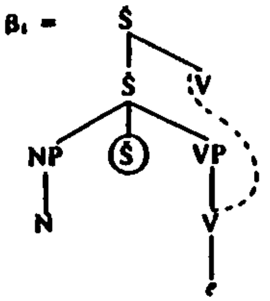
In Joshi, Levy, and Takahashi, 1975, a variant of TAG was considered. This variant will allow construction of structures with two (or even more) spines, as described in Bresnan et al., 1983. The matter will not be discussed here.

Let $G = (I,A)$ be a TAG, where

(119) I: $\alpha_1 =$

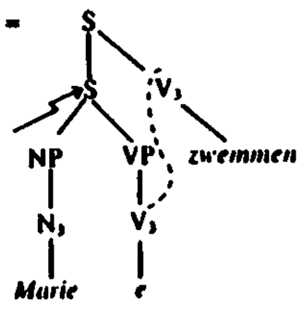


A: $\beta_1 =$



Let us look at some derivations. We will start with α_1. (The Ns and Vs are indexed for reading convenience.)

(120) $\gamma_1 = \alpha_1 =$

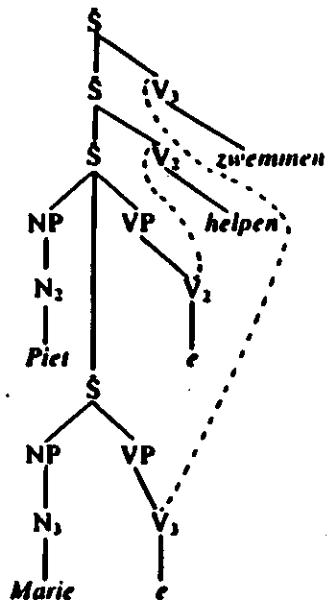


Marie zwemmen.

Adjoining β_1 to γ_1 at the indicated node we obtain γ_2.

Note that in the TAG described, in each step of the derivation the corresponding Ns and Vs enter the derivation at the same time. Also even if one goes bottom-up on the tree γ_3, the corresponding Ns and Vs are together.

(121) γ_2 =



Piet Marie helpen zwemmen.

6.5. Some further problems

A number of issues are only lightly treated in this paper. In particular, some of the discussion of linguistic relevance is very brief. A fuller discussion is given in Joshi and Kroch, 1984. Some other problems need further investigation. I will only mention these here.

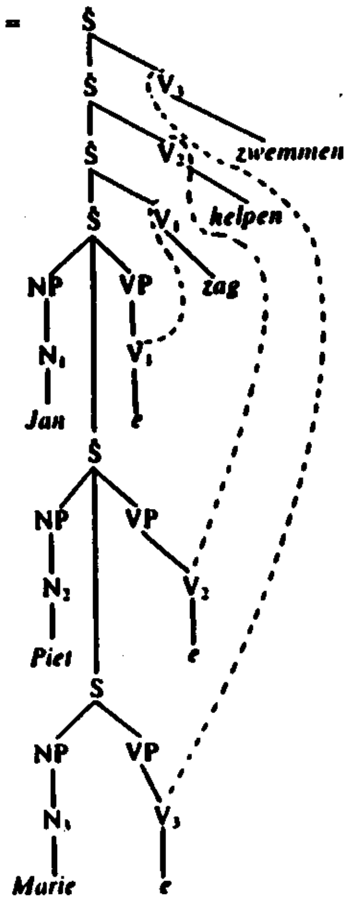
1. Some initial trees are not matrix sentences. Thus they have to have an auxiliary tree adjoined before they become sentences. This requires filtering out those derivation trees where such adjoining has not taken place. It is easy to set up a mechanism for achieving this, without affecting the general character of TAGs, especially with respect to their generative capacity.

2. To accommodate coordination, both *I* and *A* sets have to be enlarged by introducing some schemas. The *I* and *A* sets then can become potentially infinite, an extension that will not affect generative capacity in any essential way. However, this aspect needs much further investigation, which has not been carried out yet.

3. Each step in the derivation in a TAG introduces an elementary tree (a large structural chunk) together with the associated lexical items. This property of TAGs may turn out to be highly relevant for modeling some aspect of sentence production. I am currently studying this suggestion.

Adjoining β_1 at the indicated node in γ_2 we get γ_3.

(122) γ_3 =



Jan Piet Marie zag helpen zwemmen.

Acknowledgments

This work is partially supported by the NSF Grant MCS 81-07290.

I wish to thank Bob Berwick, Jean Gallier, Gerald Gazdar, Ron Kaplan, Lauri Karttunen, Tony Kroch, Mitch Marcus, Bill Marsh, Stan Peters, Ellen Prince, Geoff Pullum, Bob Ritchie, Stu Shieber, R. Shyamasundar, Hans Uszkoreit, Bonnie Webber, Scott Weinstein, and Takashi Yokomori for their valuable comments. I am particularly indebted to Tony Kroch for comments on certain lingustic issues and to Takashi Yokomori for comments on some formal aspects. Some forthcoming publications will discuss these issues in greater detail (see Joshi and Kroch, 1984, and Joshi and Yokomori, 1984).

References

Aho, Alfred. 1969. Indexed grammars. *Proceedings of the 8th IEEE meeting on Switching and Automata Theory.*

Bresnan, Joan W., Ronald Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in Dutch. *Linguistic Inquiry.*

Gazdar, Gerald. 1982. Phrase structure grammars. In: Pauline Jacobson and Geoffrey Pullum (eds.), *The nature of syntactic representations.* Dordrecht, Holland: D. Reidel.

Joshi, Aravind K., and Leon Levy. 1982. Phrase structure trees bear more fruit than you would have thought. *American Journal of Computational Linguistics.*

Joshi, Aravind K., Leon S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences.*

Joshi, Aravind K., and Takashi Yokomori. 1983. Parsing of tree adjoining grammars. Technical Report. Department of Computer and Information Science, University of Pennsylvania.

Joshi, Aravind K., and Leon S. Levy. 1978. Local constraints. *SIAM Journal of computing.*

Joshi, Aravind K., and Takashi Yokomori. 1983. Some characterization theorems for tree adjoining grammars and recognizable sets. Technical Report. Department of Computer and Information Science, University of Pennsylvania.

Joshi, Aravind K., and Tony Kroch. Forthcoming 1984. Linguistic significance of TAG's. (tentative title).

Kaplan, Ronald, and Joan W. Bresnan. 1983. Lexical functional grammar – a formal system for grammatical representation. In Joan Bresnan (ed.), *The mental representation of grammatical relations.* Cambridge, Mass.: MIT Press.

Peters, Stanley, and Robert Ritchie. 1969. Immediate constituent analysis revisited. *Proceedings of the ACM Symposium on Theory of Computing.*

Peters, Stanley, and Robert Ritchie. 1982. Phrase linking grammars. Technical Report. Department of Linguistics, University of Texas at Austin.

Pullum, Geoffrey K. 1982. Free word order and phrase structure rules. In J. Pustejovsky and P. Sells (eds.), *Proceeding of NELS 12.* Amherst, Mass.