6. Tree adjoining grammars

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 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.

\[
\begin{align*}
S & \rightarrow a \ T \ b \\
T & \rightarrow S \ a \\
T & \rightarrow T \ T \\
T & \rightarrow a \ b
\end{align*}
\]

\( S \) 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 \( \alpha \) is an initial tree if it is of the form in (1).

\[
\begin{align*}
(1) \quad \alpha &= \\
& \begin{array}{c}
\text{S} \\
\downarrow \\
\end{array}
\end{align*}
\]

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

\[
\begin{align*}
(2) \quad \beta &= \\
& \begin{array}{c}
\text{X} \\
\downarrow \\
\end{array}
\end{align*}
\]

That is, the root node of \( \beta \) 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 \( \beta \). 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 \( X \).

6. **Tree adjoining grammars**

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

\[
\begin{align*}
(3) \quad I: \quad \alpha_1 &= \\
& \begin{array}{c}
\text{S} \\
\downarrow \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
A: \quad \beta_1 &= \\
& \begin{array}{c}
\text{7} \\
\downarrow \\
\end{array}
\end{align*}
\]

That is, 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 adjunction), which composes an auxiliary tree \( \beta \) with a tree \( \gamma \). Let \( \gamma \) be a tree with a node labeled \( X \) and let \( \beta \) be an auxiliary tree with the root labeled \( X \) also. (Note that \( \beta \) must have, by definition, a node – and only one – labeled \( X \) on the frontier.) Adjoining can now be defined as follows: If \( \beta \) is adjoined to \( \gamma \) at the node \( n \), then the resulting tree \( \gamma' \) is as shown in (4).

\[
\begin{align*}
(4) \quad \gamma &= \\
& \begin{array}{c}
\text{S} \quad \theta = \\
\downarrow \\
\end{array}
\end{align*}
\]

The tree \( t \) dominated by \( X \) in \( \gamma \) is excised, \( \beta \) is inserted at the node \( n \) in \( \gamma \) and the tree \( t \) is attached to the foot node (labeled \( X \)) of \( \beta \); that is,
is inserted or adjoined to the node \( n \) in \( \gamma \), 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) \[ \gamma_0 = \alpha_1 = \begin{array}{c}
\begin{array}{c}
S \\
T
\end{array}
\end{array} \quad \beta = \beta_3 = \begin{array}{c}
\begin{array}{c}
T \\
S \\
a
\end{array}
\end{array} \]

\( \beta_3 \) will be adjoined to \( \gamma_0 \) at \( T \) as indicated in \( \gamma_0 \). The resulting tree \( \gamma_1 \) is then as in (6).

(6) \[ \gamma_1 = \begin{array}{c}
\begin{array}{c}
S \\
T \\
a
\end{array}
\end{array} \quad \beta_3
\]

We can continue the derivation by adjoining, say, \( \beta_4 \), at \( S \) as indicated in \( \gamma_1 \). The resulting tree \( \gamma_2 \) is then as in (7).

(7) \[ \gamma_2 = \begin{array}{c}
\begin{array}{c}
S \\
T \\
a
\end{array}
\end{array} \quad \beta_4
\]

Note that \( \gamma_0 \) is an initial tree, a sentential tree. The derived trees \( \gamma_1 \) and \( \gamma_2 \) 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 \( G' \).

6. Tree adjoining grammars

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) \[ I: \quad \alpha_1 = \begin{array}{c}
\begin{array}{c}
S \\
T
\end{array}
\end{array} \]

\[ A: \quad \beta_1 = \begin{array}{c}
\begin{array}{c}
S \\
T \\
a
\end{array}
\end{array} \quad \beta_2 = \begin{array}{c}
\begin{array}{c}
T \\
S \\
b
\end{array}
\end{array} \]
Let us look at some derivations in $G$.

\[(9) \quad \gamma_0 = \alpha_1 = S \quad \gamma_3 = \gamma_1 \text{ with } \beta_3 \text{ adjoined at } T \]

\[\gamma_1 = \gamma_0 \text{ with } \beta_1 \text{ adjoined at } S \]

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

\[(10) \quad L = \{a^n b^n c^n | n \geq 0\} \]

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

\[(11) \quad I: \quad \alpha_1 = S \quad \beta_1 = \beta_2 = \]

\[A: \quad \beta_1 = \]

\[\quad \beta_2 = \]

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

\[(12) \quad L = \{ (a b)^n c^n | n \geq 0 \} . \]

$L(G)$ 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:

\[(13) \quad L(G) = L_1 \{ w \in c^n | n \geq 0 \}, w \text{ is a string of } a \text{'s and } b \text{'s such that } \]

(i) the number of $a$'s is equal to the number of $b$'s,

(ii) for any initial substring of $w$, the number of $a$'s is greater than 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 $a^* b^* c^*$ results in the language

\[(14) \quad L_2 = \{ a^n b^n c^n | n \geq 0 \} = L_1 \cap a^* b^* c^* . \]

$L_2$ 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 part (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 limitation 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 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 in Example 6.1.3, that it will not be possible to avoid getting the $a$'s and $b$'s mixed. (It will be shown subsequently how $L_2$ can be generated by a TAG with local constraints, but in a rather special way.) The so-called copy language

\[(15) \quad L_3 = \{ w \in a^n c^n | w \in \{a,b\}^* \} . \]

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

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

$\text{CFL} \subset \text{TAL} \subset \text{Indexed Languages} \subset \text{CSL}$
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.

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.

6. Tree adjoining grammars

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 $G = (T,A)$, where

(17) $\alpha_1 = S \quad \beta_1 = \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
S \\
g \\
a \quad b \\
\end{array}
\end{array}
\end{array} \\
\beta_2 = \begin{array}{c}
\begin{array}{c}
S \\
g \\
a \quad b \\
\end{array}
\end{array}$

Let $\gamma_0 = \alpha_1 = \begin{array}{c}
\begin{array}{c}
S \\
g \\
a \quad b \\
\end{array}
\end{array}$

Adjoining $\beta_1$ at $S$ as indicated in $\gamma_0$, we have

(18) $\gamma_1 = \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
S \\
g \\
a \quad b \\
\end{array}
\end{array}
\end{array}$

The terminal string corresponding to $\gamma_1$ is $a \quad e \quad b$, where the dependency is indicated by the solid line.

Adjoining $\beta_1$ again at $S$ as indicated in $\gamma_2$, we have

(19) $\gamma_2 = \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
S \\
a \quad b \\
\end{array}
\end{array}
\end{array}$

$w = a \quad e \quad b \quad b$ (nested dependencies)
Adjoining $\beta_2$ at $T$ as indicated in $\gamma_2$, we have

$$\gamma_2 = \begin{array}{c}
\begin{array}{c}
S
\end{array}
\end{array}$$

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

$\beta_1$ and $\beta_2$ each have one link. $\gamma_2$ and $\gamma_3$ show how the linking is preserved in adjoining. In $\gamma_2$, 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 $\gamma_2$ the dependencies between the $a$'s and $b$'s, as reflected in the terminal string, are properly nested, while in $\gamma_3$ 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 $\beta_1$ and $\beta_2$ have only one link each. The nestings and crossings in $\gamma_2$ and $\gamma_3$ 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,

$$\beta = \begin{array}{c}
\begin{array}{c}
X
\end{array}
\end{array}$$

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

$$I: \alpha_1 = \begin{array}{c}
\begin{array}{c}
S
\end{array}
\end{array}$$

$$A: \beta_1 = \begin{array}{c}
\begin{array}{c}
A
\end{array}
\end{array}$$

$\in LR(\phi_1 - \phi_1) \land TB(\phi_1 - \phi_1)$

For the auxiliary tree $\beta_1$ there is a local constraint specified to the right of $\beta_1$. This constraint can be treated as a predicate that must be true of
a node, say, \( n \), of a tree \( t \) in order that \( \beta_1 \) 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

\[
(24) \quad \rho_1 \phi_1 A \psi_1 \rho_2,
\]

where \( \rho_1 \) and \( \rho_2 \) are arbitrary strings of terminals and nonterminals, and \( \phi_1 \) and \( \psi_1 \) 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 \( \phi_1 \) and a right context \( \psi_1 \) 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

\[
(25) \quad \rho_1 \phi_2 A \psi_2 \rho_2,
\]

where \( \rho_1 \) and \( \rho_2 \) are arbitrary strings of terminals and nonterminals, and \( \phi_2 \) and \( \psi_2 \) 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 \( \phi_2 \) and a bottom context \( \psi_2 \) 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 \( \beta_1 \) 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

\[
(26) \quad J: \quad \alpha_1 = S
\]

\[
A: \quad \beta_1 = \begin{array}{c}
S \\
\text{LR}(a- \text{)} \wedge TB(T-b) \\
\vee (\neg \text{LR}(b- \text{)} \wedge TB(-e))
\end{array}
\]

\[
\beta_2 = \begin{array}{c}
T \\
\text{LR}(a- \text{)} \wedge TB(S-b)
\end{array}
\]

\( \gamma_1 \) is derived by adjoining \( \beta_1 \) to \( \gamma_1 \) at the indicated node \( S \) in \( \gamma_1 \). Note that \( \beta_1 \) cannot be adjoined now to the lowermost \( S \) node in \( \gamma_1 \) because the local constraint is not satisfied. Note also that \( \beta_1 \) cannot be adjoined to the top node \( S \) in \( \gamma_2 \).

\[
(27) \quad \gamma_1 = \alpha_1 = S
\]

\[
\gamma_2 \text{ is derived by adjoining } \beta_1 \text{ to } \gamma_1 \text{ at the indicated node } S \text{ in } \gamma_1. \text{ Note that } \beta_1 \text{ cannot be adjoined now to the lowermost } S \text{ node in } \gamma_2 \text{ because the local constraint is not satisfied. Note also that } \beta_1 \text{ cannot be adjoined to the top node } S \text{ in } \gamma_2.
\]

\[
\gamma_2 = \begin{array}{c}
S \\
\text{LR}(a- \text{)} \wedge TB(T-b) \\
\vee (\neg \text{LR}(b- \text{)} \wedge TB(-e))
\end{array}
\]

\[
\beta_2 = \begin{array}{c}
T \\
\text{LR}(a- \text{)} \wedge TB(S-b)
\end{array}
\]

\( \gamma_3 \) is derived from \( \gamma_2 \) by adjoining \( \beta_2 \) to the indicated node labeled \( T \) in \( \gamma_2 \). Note that \( \beta_2 \) can be adjoined to \( \gamma_2 \) only at the node \( S \) in the middle of \( \gamma_3 \), but not at the top \( S \) node or the bottom \( S \) node of \( \gamma_3 \) because the local constraint is not satisfied. Also \( \beta_2 \) cannot be adjoined to either one of the \( T \) nodes of \( \gamma_3 \).
$\gamma_4$ is derived by adjoining $\beta_1$ to the indicated $S$ node in $\gamma_3$. The only adjoinment that is possible for $\gamma_4$ is a $\beta_2$ adjoined at the indicated $T$ node in $\gamma_4$.

It is clear that in this grammar at each stage of the derivation the only node that receives adjoinment 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

$$L_1 = \{a^n b^n c^n \mid n \geq 0\}$$

(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)

$$L_3 = \{wew \mid w \in \{a, b\}^*\}$$

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 at 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 $\beta_1$ there is a link between $b$ and $a$, and between $c$ and $a$. Thus

$$\beta_1 = \ldots$$

and similarly for $\beta_2$.

If we write the terminal string of $\gamma_4$ indicating the dependencies by the solid lines, we will have

Thus the $a$'s and $b$'s have nested dependencies and the $a$'s and $c$'s have cross-serial dependencies. If in $\beta_1$ 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.

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

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

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

\begin{equation}
\begin{aligned}
| & a_1 \ a_2 \ a_3 \ b_3 \ b_2 \ b_1 \\
| & \ | \\
\end{aligned}
\end{equation}

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

\begin{equation}
\begin{aligned}
| & a_1 \ a_2 \ b_2 \ b_1 \ c_1 \ c_2 \ c_3 \ d_3 \ d_2 \ d_1 \ d_0 \\
| & \ | \\
\end{aligned}
\end{equation}

or

\begin{equation}
\begin{aligned}
| & a_1 \ a_2 \ c_1 \ c_2 \ c_3 \ d_3 \ d_2 \ d_1 \ b_2 \ b_1 \\
| & \ | \\
\end{aligned}
\end{equation}

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

\begin{equation}
\begin{aligned}
| & a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ b_3 \\
| & \ | \\
\end{aligned}
\end{equation}

(Actually, we may 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, there is no cross-serial dependency between only

6. Tree adjoining grammars

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

\begin{equation}
\begin{aligned}
| & a_1 \ a_2 \ b_1 \ b_2 \ c_1 \ c_2 \\
| & \ | \\
\end{aligned}
\end{equation}

(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

\begin{equation}
\begin{aligned}
| & \ldots \ a_1 \ a_2 \ b_1 \ b_2 \ \ldots \ c_1 \ c_2 \ c_3 \ d_1 \ d_2 \ d_3 \ \ldots \\
| & \ | \\
\end{aligned}
\end{equation}

or

\begin{equation}
\begin{aligned}
| & \ldots \ a_1 \ a_2 \ \ldots \ c_1 \ c_2 \ c_3 \ d_1 \ d_2 \ d_3 \ \ldots \ b_1 \ b_2 \ \ldots \\
| & \ | \\
\end{aligned}
\end{equation}

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

\begin{equation}
L_4 = \{a^nb^nc^nd^n | n \geq 0\}
\end{equation}

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 de-
dependencies are either disjoint or one is properly nested inside the other; hence, languages such as

\[ L_3 = \{ a^n b^n c^n d^n e^n | n \geq 0 \} \]

and

\[ L_5 = \{ w e w e w | w \in \{a, b\}^* \} \] (double copy language)
cannot be generated by a TAG with local constraints. Both \( L_3 \) and \( L_5 \) 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 \( \gamma_{i+1} \) be derived from \( \gamma_i \) by adjoining \( \beta_i \) to \( \gamma_i \). Then the terminal strings of \( \gamma_i \) and \( \gamma_{i+1} \), 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 \( \beta_i \), say, \( w_j \) (not counting the single non-terminal symbol in the frontier of \( \beta_i \)), i.e.,

\[ |w_{i+1}| = |w_i| + |w_j| \]

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

\[ |w| = |w_1| + a_1 |w_1| + a_2 |w_2| + \cdots + a_m |w_m| \quad a_i \geq 0, \quad 1 \leq i \leq m \]

where \( w_1 \) is the terminal string of some initial tree and \( w_i, 1 \leq i \leq m \), the terminal string of the \( i \)-th 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

\[ L_1 = \{ a^n \} \]

\[ L_6 = \{ a^n \} \]

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-
6.3.4. Formal remarks

Let $L_0$ 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 $L_0$.

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

(52) $\alpha_1 = a \ b \ c \ a \ b \ c \ a \ b \ c$

This is not the simplest TAG for $L_0$. Note also that $\beta_2$ has a local constraint that requires a top context $S$ and a bottom context $T$ for $\beta_3$ to be adjoinable. The bar over some $a$'s serves to indicate the dislocated $a$'s. The TAG, $G$, above not only generates $L_0$, 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 $L_0$; 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) $G' : S \rightarrow \bar{a} S$
    $S \rightarrow \bar{a} b S c$
    $S \rightarrow c$

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.)

\[ L_{10} = \{ w w \in \{a,b,c\}^* \text{ and the number of } a's = \text{ the number of } b's = \text{ the number of } c's \} \]

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 Krich, 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.

6. Tree adjoining grammars

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

(55) Initial Trees:

\[
\alpha_1 = \begin{array}{c}
S \\
\text{NP} \quad \text{VP}
\end{array}
\]

\[
\alpha_2 = \begin{array}{c}
S \\
\text{NP} \quad \text{VP}
\end{array}
\]

Note that \(\alpha_1\) corresponds to a "minimal sentence" with a transitive verb, e.g.,

(56) The man met the woman.

and \(\alpha_2\) 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 frontier. In the linguistic context, the nodes on the frontier will be preterminal 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 \(\alpha_1\) as the initial tree, then

(58) \(\gamma_1 = \alpha_1 \text{ (with lexical items inserted)} = \)

As we continue the derivation by selecting auxiliary trees and adjoining them appropriately, we follow the same convention; that is, as each auxiliary tree is chosen, we make the lexical insertions. Thus in a derivation in a TAG, lexical insertion goes hand in hand with the derivation. Each step in the derivation selects an elementary tree together with a set of 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.

\[
\text{(59)} \\
S \\
\text{NP} \\
\text{Det} \\
\text{N} \\
\text{V} \\
\text{NP} \\
\text{VP} \\
\text{NP} \\
\text{V} \\
\text{NP} \\
\text{VP}
\]

This is because, in the first case, the number agreement is violated and in the second case, \textit{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 \(\alpha_2\), then the choice of lexical items as shown in (60) would be permitted.

\[
\text{(60)} \\
S \\
\text{NP} \\
\text{DET} \\
\text{N} \\
\text{V} \\
\text{NP} \\
\text{VP} \\
\text{N} \\
\text{be} \\
\text{V} \\
\text{by} \\
\text{NP} \\
\text{NP} \\
\text{N}
\]

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

\[
\text{(61)} \\
\alpha_3 = \\
S \\
\text{NP} \\
\text{N} \\
\text{V} \\
\text{NP} \\
\text{N} \\
\text{V} \\
\text{VP} \\
\text{NP} \\
\text{N} \\
\text{be} \\
\text{V} \\
\text{by} \\
\text{NP} \\
\text{NP} \\
\text{N}
\]

Note that \(\alpha_4\) and \(\alpha_5\) are the passive forms of \(\alpha_1\) and \(\alpha_2\), respectively. They have shown a flat structure for passive for convenience only. Nothing in this section hangs on this particular structure for passive. (For further details about the analysis of passive with respect to \textit{TAGs}, see Joshi and Kroch, 1984.)

\[
\text{(62)} \\
\alpha_4 = \\
S \\
\text{NP} \\
\text{DET} \\
\text{N} \\
\text{be} \\
\text{V} \\
\text{by} \\
\text{NP} \\
\text{NP} \\
\text{N}
\]

\[
\text{Mary was met by John}
\]

\[
\text{(63)} \\
\alpha_5 = \\
S \\
\text{NP} \\
\text{DET} \\
\text{N} \\
\text{be} \\
\text{V} \\
\text{PP} \\
\text{by} \\
\text{NP} \\
\text{NP} \\
\text{N}
\]

\[
\text{The book was given to Mary by John}
\]
In $\alpha_{10}$, 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 $\alpha_{10}$, 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) $\alpha_{15} = \ldots$

Who met Mary?

$\alpha_{16} = \ldots$

Who did John meet?

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

6. Tree adjoining grammars

tree rewriting rules, especially since both the domains and the co-domains of the rules will be finite. These rules will be abbreviated 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) $\alpha_{25} = \ldots$  \hspace{1cm} $\alpha_{15} = \ldots$

Pro to invite Mary \hspace{1cm} John to like Mary

Pro to invite Mary

Note that $\alpha_{25}$ and $\alpha_{15}$ are similar, except that in the first case the subject NP is realized as Pro and in the second case by a lexical item. $\alpha_{25}$ will be used in the derivation of sentences such as

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

$\alpha_{15}$ will be used in deriving sentences such as

(67) John seems to like Mary

(68) $\alpha_{26} = \ldots$
α₂₆ and α₂₈ differ in the same way α₂₃ and α₂₅. α₂₆ will be used in deriving sentences such as

(69) Who did John try to invite?

α₂₈ 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

β₁ is used to build a subject relative clause around an NP as in

(72) The boy who met Mary left.

The link in β₁ links the extracted NP node to wh.

β₂ corresponds to the object relative clause.

(73) β₂ is used to build an object 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, β₁ from β₁, or even all of the auxiliary trees, say for example, β₁ from α₁, 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 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) \[ \beta_4 = \]

\[ S \]

\[ \begin{array}{c} NP \\ N \\ VP \\ S \\ N \\ V \\ NP \\ S \end{array} \]

John persuaded Bill S

Did John persuade Bill S

(78) \[ \beta_7 = \]

\[ S \]

\[ \begin{array}{c} NP \\ N \\ VP \\ S \\ N \\ V \\ S \end{array} \]

John expected S

Did John expect S

6. Tree adjoining grammars

(79) \[ \beta_10 = \]

\[ S \]

\[ \begin{array}{c} NP \\ V \\ VP \\ S \end{array} \]

\[ \begin{array}{c} that \\ S \end{array} \]

\[ \begin{array}{c} seems \\ VP \end{array} \]

John knew that S

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

\[ \beta_4, \beta_5, \beta_7, \beta_8, \text{ and } \beta_9 \text{ correspond to the sentences involving sentential complements. For example,} \]

(80) John persuaded Bill to invite Mary.

would be derived starting with an initial tree corresponding to

(81) Pro to invite Bill

to which \( \beta_4 \) is adjoined giving

(82) John persuaded Bill Pro to invite Mary.

\( \beta_4 \) would be used in deriving

(83) Who did John persuade Bill to invite?

\( \beta_7 \) and \( \beta_8 \) will be used in a similar fashion. \( \beta_9 \) 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.

(87) $\gamma_i = \alpha_1$

```
  S
 / | \
NP VNP
  /     |
Det N    V
 the girl e
```

The girl is a senior.

I will now derive several sentences.

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

Let us take $\beta_1$ (with appropriate lexical insertions).

(89) $\beta_1 =$

```
  S
 / | \
NP VNP
  /     |
NP S NP
   /     |
wh S VNP
    /     |
e V NP
    /     |
met N
    Bill
```

6. Tree adjoining grammars

$\beta_1$ is then adjoined to $\alpha_1$ at the indicated node labeled NP resulting in $\gamma_1$.

(90) $\gamma_1 =$

```
  S
 / | \
NP VNP
  /     |
Det N wh
 the girl e
```

The girl who met Bill is a senior.

(91) John persuaded Bill to invite Mary.

We will start with the initial tree $\alpha_{25}$.

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

```
  S
 / | \
NP VNP
  /     |
Pro to VP
   /     |
NP V NP
   /     |
V NP
   /     |
met N
    Bill
```

Then we take $\beta_6$. 
(93) \[ \beta_a = \begin{array}{c}
S \\
NP \\
V \\
NP \\
S \\
John \\
persuaded \\
N \\
Bill
\end{array} \]

John persuaded Bill S

\[ \beta_a \] is adjoined to \( \gamma_1 \) of the indicated node labeled S resulting in \( \gamma_2 \).

(94) \[ \gamma_2 = \begin{array}{c}
S \\
NP \\
V \\
NP \\
NP \\
S \\
John \\
persuaded \\
N \\
VP \\
VP \\
Bill \\
Pro \\
to \\
VP \\
V \\
NP \\
invite \\
N \\
Mary
\end{array} \]

John persuaded Bill to invite Mary.

Note that if we start with \( \alpha_{23} \), which is like \( \alpha_{23} \) except that instead of Pro, we have a lexical NP

(95) \[ \gamma_1 = \alpha_{23} = \begin{array}{c}
S \\
NP \\
V \\
NP \\
S \\
Jim \\
to \\
VP \\
V \\
NP \\
invite \\
N \\
Mary
\end{array} \]

Jim to invite Mary

Adjoining \( \beta_a \) to \( \alpha_{23} \) at the indicated node labeled S, we get \( \gamma_1 \).

(96) \[ \gamma_1 = \begin{array}{c}
S \\
NP \\
V \\
NP \\
NP \\
V \\
NP \\
S \\
John \\
persuaded \\
N \\
Bill \\
to \\
VP \\
V \\
NP \\
Jim \\
icvite \\
N \\
Mary
\end{array} \]

* John persuaded Bill Jim to invite Mary.

\( \gamma_1 \) can be disallowed if while adjoining \( \beta_a \) to \( \gamma_1 \) (or \( \gamma_1 \)) 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 \( \beta_a \) as follows.

(97) \[ \beta_a = \begin{array}{c}
S \\
NP \\
V \\
NP \\
NP \\
V \\
NP \\
S \\
:TB (-NP Pro)
\end{array} \]

(98) Who did John persuade Bill to invite?

We will start with the initial tree \( \alpha_{23} \).

(99) \[ \gamma_1 = \alpha_{23} = \begin{array}{c}
S \\
NP \\
V \\
NP \\
S \\
Jim \\
to \\
VP \\
V \\
NP \\
invite \\
N \\
Mary
\end{array} \]

Then we take \( \beta_4 \).
$\beta_6$ is adjoined to $\gamma_1$ at the indicated node labeled S, resulting in $\gamma_2$.

Note that the link in $\gamma_1$ is preserved in $\gamma_2$; it is stretched resulting in the so-called unbounded dependency.

It is now easy to see that starting from $\alpha_{15}$ and then adjoining $\beta_7$ to $\alpha_{25}$ at the root node S, we obtain

(102) John expected Bill to invite Mary.

Starting with $\alpha_{16}$ and adjoining $\beta_7$ to $\alpha_{16}$ 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,

we can obtain (105) as follows.

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

We will start with $\alpha_{25}$ corresponding to

(105a) Pro to invite Mary.

Then we adjoin $\beta_{30}$ to the S node in $\alpha_{25}$, giving

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

Finally, adjoining $\beta_6$ to the root node S of the tree corresponding to (105a) 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 derivations for sentences containing the so-called equi and raising verb. This observation is due to Tony Kroch. (For further details, see Joshi and Kroch, 1984.) Thus

(106) John tried to please Mary.

will be derived as follows. We will start with $\alpha_{25}$

(107) $\gamma_1 = \alpha_{35}$

We then take $\beta_7$. 
6. Tree adjoining grammars

(112) \( \beta_{15} = \)

Adjoining \( \beta_{15} \) to \( \gamma_1 \) at this indicated node labeled VP, we obtain \( \gamma_2 \).

(113) \( \gamma_1 = \)

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

\( \ldots \) Jan Piet Marie saw help swim

\( \ldots \) Jan saw Piet help Marie swim

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

(115) \( \ldots a_1 a_2 \overline{a_3} b_1 b_2 b_3 \ldots \)
The string language is of the form \( a^n b^n | n \geq 1 \), 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 zeg.
(b) Piet helpen.
(c) Marie zwemmen.

with the derivation beginning with

\[(117)\]  
Jan zeg.

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.

6. Tree adjoining grammars

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

\[(119)\]

\[\begin{align*}
&\alpha_1 = \\
&\beta_1 = \\
&\gamma_1 = \alpha_1 = \\
&\text{Marie zwemmen.}
\end{align*}\]

Let us look at some derivations. We will start with \( \alpha_1 \). (The Ns and Vs are indexed for reading convenience.)

\[(120)\]  
\[\text{Marie zwemmen.}\]

Adjoining \( \beta_1 \) to \( \gamma_1 \) at the indicated node we obtain \( \gamma_2 \).

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 \( \gamma_1 \), the corresponding Ns and Vs are together.
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 adjoinings have 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.

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 Sieber, R. Shyamadasan, Hans Uszkoreit, Bonnie Webber, Scott Weinstein, and Takashi Yokomori for their valuable comments. I am particularly indebted to Tony Kroch for comments on certain linguistic 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


