Using Descriptions of Trees in a Tree Adjoining Grammar

K. Vijay-Shanker*
University of Delaware

This paper describes a new interpretation of Tree Adjoining Grammars (TAG) that allows the embedding of TAG in the unification framework in a manner consistent with the declarative approach taken in this framework. In the new interpretation we present in this paper, the objects manipulated by a TAG are considered to be descriptions of trees. This is in contrast to the traditional view that in a TAG the composition operations of adjoining and substitution combine trees. Borrowing ideas from Description Theory, we propose quasi-trees *as a means to represent partial descriptions of trees. Using quasi-trees, we are able to justify the definition of feature structure–based Tree Adjoining Grammars (FTAG) that was first given in Vijay-Shanker (1987) and Vijay-Shanker and Joshi (1988). In the definition of the FTAG formalism given here, we argue that a grammar* manipulates *descriptions of trees (i.e., quasi-trees); whereas the structures derived by a grammar are trees that are obtained by taking the minimal readings of such descriptions. We then build on and refine the earlier version of FTAG, give examples that illustrate the usefulness of embedding TAG in the unification framework, and present a logical formulation (and its associated semantics) of FTAG that shows the separation between descriptions of well-formed structures and the actual structures that are derived, a theme that is central to this work. Finally, we discuss some questions that are raised by our new interpretation of the TAG formalism: questions dealing with the nature and definition of the adjoining operation (in contrast to substitution), its relation to multi-component adjoining, and the distinctions between auxiliary and initial structures.*

1. Introduction

A number of grammatical systems and linguistic theories, such as Functional Unification Grammars (FUGs), Lexical Functional Grammars (LFGs), Generalized Phrase Structure Grammars (GPSGs), and Head-driven Phrase Structure Grammars (HPSGs), are said to take the unification-based approach to grammars. A common aspect shared by these grammars or theories is that they are based on specifying constraints that define well-formed structures. This work discusses viewing Tree Adjoining Grammars (TAG) in such a manner and embedding it in a unification-based framework.

Tree Adjoining Grammars (TAG) were first introduced by Joshi, Levy, and Takahashi (1975). A preliminary study of this formalism, from the point of view of its formal properties and linguistic applicability, was carried out by Joshi (1985). A detailed study of the linguistic relevance of TAG was done by Kroch and Joshi (1985). Abeille et al. (1990) discuss a fairly substantial grammar for English using the lexicalized approach to TAG that was originally proposed by Schabes, Abeille, and Joshi (1988).

* Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716.

TAG is defined as a tree rewriting system. In the definition given *traditionally*, a TAG is defined by a finite set of trees and an operation called **adjoining** to compose trees. One of the basic intuitions underlying the use of the TAG formalism is that these trees provide a large enough structure that most cooccurrence restrictions (dependencies) can be stated over (localized within) these trees. Predicate-argument, wh-dependencies, and filler-gap dependencies are examples of dependencies that can be localized in a TAG.

Our aim is to view a TAG as defining constraints on well-formed structures (according to the linguistic intuitions being instantiated in the grammar). In this paper, we argue that if we chose to interpret the objects manipulated by a TAG as trees (as is done currently) then it is not possible to embed TAG in a unification framework in a straightforward manner. We show that this follows from the fact that the adjoining operation on trees is such that it does not preserve the structural relationships that have been specified in the structures being combined. We argue that instead we should view the objects *manipulated* (to be distinguished from *derived*) by a TAG as (partial) descriptions of trees. In particular, these descriptions include the partial specification of *domination*, as in description theory or D-theory (Marcus, Hindle, and Fleck 1983), in addition to the specification of *immediate domination*. We argue that this is a well-motivated interpretation that is consistent with certain assumptions made in the lexicalized approach to TAG. We introduce **quasi-trees** as a means to structurally depict partial specifications of trees. Using this interpretation, we show that the resulting structure obtained after adjoining preserves the structural relationships described in the structures being composed.

1.1 Outline of the Paper

For the sake of contrasting the two definitions, we start by giving the currently used definition of TAG. In Section 2, we show that this definition is not consistent with the assumptions made in the unification framework. We propose a novel way of interpreting the basic objects of a TAG, borrowing ideas from description theory (D-theory). By means of an example, we introduce the notion of quasi-trees. We then show how TAG can be embedded in a unification-based framework. This interpretation of the objects manipulated by a TAG grammar as quasi-trees not only leads to our current definition of FTAG, but also explains the earlier definition (Vijay-Shanker 1987; Vijay-Shanker and Joshi 1988). In Section 3, we give examples to show why the introduction of feature structures and unification adds to the descriptive capabilities of TAG. In particular, we focus on the implementation of the so-called adjoining constraints (that determine locally which structures can be used for adjoining and whether adjunction is mandatory). We will show that not only can adjoining constraints be specified in a linguistically more appealing manner now, but also that in several cases redundant specifications of structural descriptions can be avoided.

In Section 5, we consider some possible implications of the new interpretation of the formalism proposed here. One particular question that arises is whether the operations of adjoining and even multi-component adjoining (as used in Multi-component Tree Adjoining Grammar) can be considered to be the same as the substitution operation where the characteristics of the adjoining and multi-component adjoining operations can be derived from the fundamental (linguistic) assumptions that concern the make-up of elementary objects of a grammar. Questions related to this issue, such as whether a distinction between initial and auxiliary structures (the two types of basic structures used in a TAG) needs to be made, are also raised. Further work along the lines suggested in this section depends on investigation of certain linguistic issues involved in the use of the TAG formalism that is beyond the scope of this work. Al-

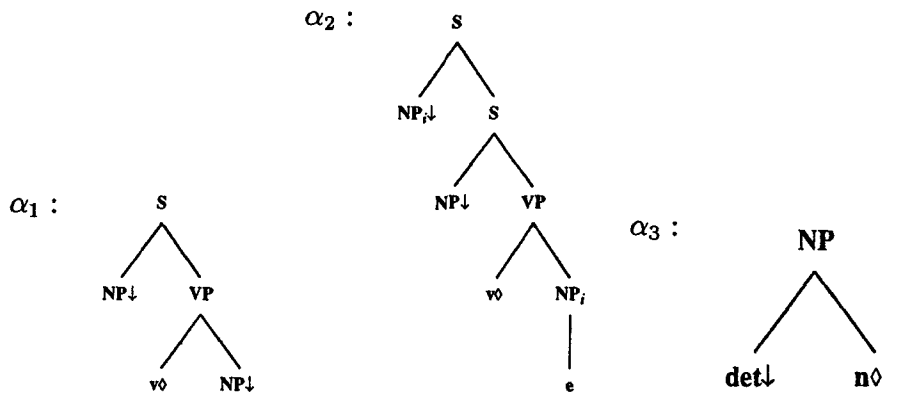


Figure 1
Initial trees.

though we provide no definitive answers to these questions, these topics are raised in this paper because they are brought out by the new interpretation of the TAG formalism that we propose.

In Section 4, we propose a logical formulation of FTAG grammars (along the lines of the logical formulation of Functional Unification Grammars given by Rounds and Manaster-Ramer [1987]) and then show how the denotation of a FTAG grammar can be obtained. The logical formulation is given, in part, to show the separation between the descriptions of well-formed structures (as specified in a FTAG grammar) and the models that satisfy these descriptions.

We would like to note that the work presented in this paper concerns a formalism and not linguistic issues. A deliberate attempt has been made to only discuss the TAG formalism in general terms rather than focusing on linguistic issues. By doing so, our intent is to pay closer attention to the formalism itself and uncover the aspects of the definition of TAG that are stipulations and those that fall out as a corollary of a formalism that tries to localize dependencies. The use of linguistic examples in this paper by no means indicates the suitability of any linguistic theories. The only assumption that we make is that a grammar will attempt to localize dependencies to the extent possible.

1.2 Introduction to Tree Adjoining Grammars

A Tree Adjoining Grammar (TAG) as *defined traditionally* is said to be specified by a finite set of **elementary** trees. Unlike the string rewriting formalisms that write recursion into the rules that generate the phrase structure, a TAG factors recursion and dependencies into a finite set of elementary trees. The elementary trees in a TAG correspond to *minimal* linguistic structures that localize the dependencies such as subcategorization, and filler-gap. There are two kinds of elementary trees: **initial trees** and **auxiliary trees**. Originally, initial trees (e.g., α_1 and α_2 in Figure 1) were defined to correspond to minimal sentential structures. Therefore, the root of an initial tree was required to be labeled by the symbol *S*. With the advent of lexicalized TAG and the use of the substitution operation, this assumption is no longer made (see α_3).

Auxiliary trees (β_1, β_2 in Figure 2) are usually defined to correspond to minimal recursive constructions. Thus, if the root of an auxiliary tree is labeled by a nonterminal

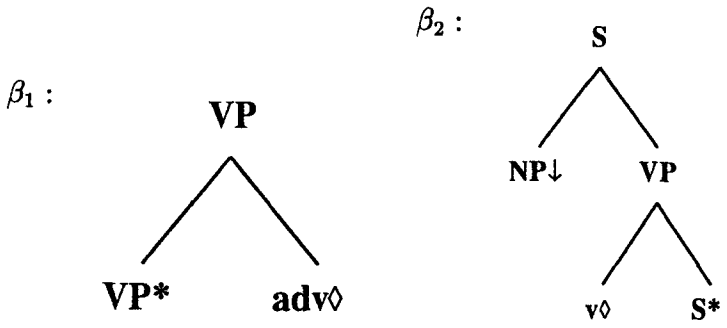


Figure 2
Auxiliary trees.

symbol, A, then there is a distinguished node, called the **foot node**, in the frontier of this tree that is also labeled by A. The foot nodes of auxiliary trees, β_1 and β_2, are indicated with an asterisk.

The *adjoining* operation is used to compose trees. An auxiliary tree, whose root and foot node are labeled A, can be adjoined at a node that is also labeled A. Adjoining may be described as follows: the subtree below the node of adjunction is excised; the auxiliary tree is inserted in its place; and the excised subtree is substituted at the foot node of the inserted auxiliary tree.

Figure 3 shows the result of adjoining β_1 at the VP node in α_1 (to yield γ_1) as well as the adjunction of β_2 in α_2 to yield γ_2. The latter example illustrates a key feature of TAG, i.e., localization of dependencies. The tree α_2 indicates the topicalization of the object, localizing the filler-gap dependency. Notice that although the dependent nodes (the two nodes labeled NP_i) are stretched apart in γ_2, the adjoining operation does not alter any dependency present in the original trees being composed.

1.2.1 Lexicalized Approach to TAG and Substitution. In the traditional approach to TAG, adjunction was the only operation used to compose trees. In the lexicalized approach to TAG as proposed by Schabes, Abeille, and Joshi (1988), the **substitution** operation is also used. In this approach, elementary trees are associated with lexical items. These lexical items (indicated by ◊) are said to be the **anchors** of the trees. These trees define the arguments required by the anchor. Figure 1 shows two initial trees α_1 and α_2 whose anchors are transitive verbs. The two trees specify the arguments required by the anchor (a transitive verb) and describe the structure for the simple declarative form and for the case where the object is topicalized. Note in both these trees, the argument (subject and object NP) nodes are not elaborated any further. This elaboration is done instead by substituting other initial trees at these nodes. The tree γ_3 (Figure 3) is the result of substituting α_3 at the subject NP node in α_1. In a lexicalized TAG, frontier nodes labeled by nonterminals (with the exception of foot nodes) are marked for substitution (specified by ↓).

1.2.2 Adjoining Constraints. So far, the only restriction we have placed on the set of auxiliary trees that can be adjoined at a node is that the label of that node must be the same as the label of the root (and the foot) node of the auxiliary tree. However, often it becomes necessary to allow only a subset of such auxiliary trees to be adjoined at

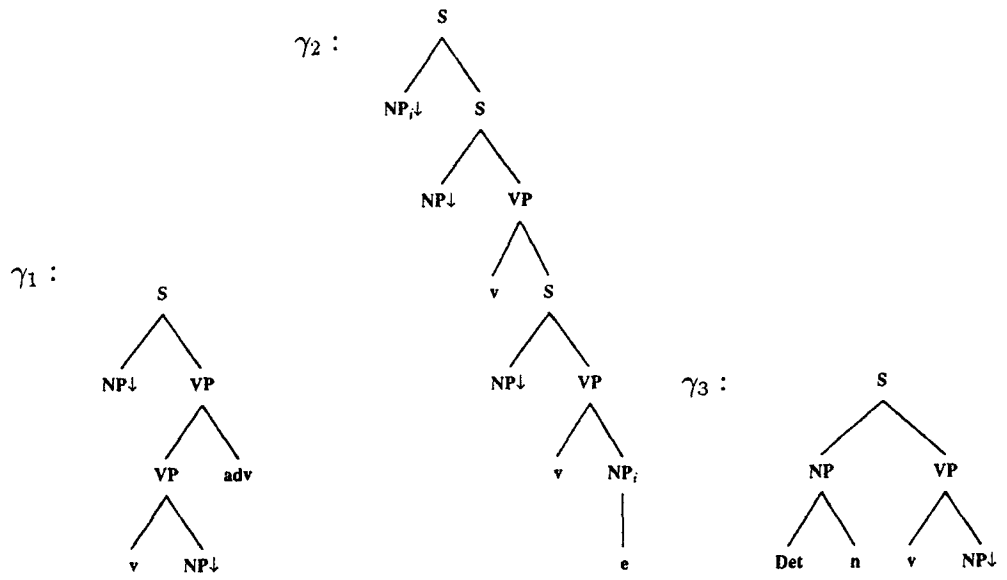


Figure 3
Some examples of adjoining and substitution.

a node. In a TAG, associated with each node is a list of auxiliary trees that can be adjoined at that node. This specification of a set of auxiliary trees with each node is called the **Selective Adjoining** (SA) constraints of the nodes. A node is said to have a **Null Adjoining** (NA) constraint if no auxiliary tree is allowed to be adjoined at that node. An NA constraint is specified by associating an empty set with a node. In current TAG literature NA constraints are therefore said to be a special case of SA constraints. In addition, for some nodes it is necessary to insist that adjunction is mandatory at a node. In such a case, we say that the node has an **Obligatory Adjoining** (OA) constraint.

A more detailed description of TAG, the use of adjoining constraints, their propagation during derivation, and their usefulness in providing linguistic analyses may be found in Kroch and Joshi (1985). At this point we would like to note that by the specification of such adjoining constraints are *stipulations* of the adjunction possibil-

ities at that node. On the other hand, we will see that in the version of FTAG we define here, decisions such as the choice of auxiliary trees that can be adjoined at a node or whether adjunction is mandatory at a node follows from the assertions (stated in terms of feature structures) about the linguistic features of individual nodes, rather than being specific to the adjoining operation. In fact, in this paper, we would like to highlight this issue while addressing the usefulness of this "unification-based approach" to TAG.

2. A Unification-Based Approach to Tree Adjoining Grammars (FTAG)

In the unification-based approach to grammars, the rules of a grammar are viewed as constraints that define well-formedness. At any point during derivation, the structures built reflect the information known at (or the constraints specified up to) that point. Further derivation leads to more constraints being specified. We begin this section by illustrating why the traditional definition of TAG is incompatible with this aspect of the unification-based approach to grammars.

2.1 Adjoining of Trees

Given α_1 (Figure 1), we can state that there is a relationship between the S node and the v node that is fixed by the fact that we have stated that α_1 is a *tree*. For instance, one of the assertions we can make is that (since we consider α_1 as a tree) following two immediate domination (ID) links from the S node leads us to the v node. Now consider the tree γ_1 (Figure 3) obtained by adjoining at the VP node (of α_1) that lies along the path from the S node to the v node. In γ_1, although the S and v node are still present, the v node is no longer the grandchild (two ID links) of the root node. This example illustrates that, in general, the adjoining operation on trees nullifies certain assertions that can be made about the component trees (that are composed). The reason that the traditional definition of TAG is not compatible with the unification approach is that it defines that the grammar manipulates (composes) fully specified structures (trees in this case) rather than partially specified structures. The composition operation of adjoining creates a new structure that does not maintain all of the properties that held in the original (fully specified) structures of which it is composed.

In the rest of the paper we will discuss an alternate definition of TAG and argue that our proposed definition is more compatible with the unification approach. Unlike the traditional definition of TAG, we do not consider the objects manipulated by a grammar to be trees. Rather, we will say that although the elementary objects do specify tree structures, they do so only in a *partial fashion*.

2.2 A New Interpretation of TAG Objects

We start by examining the nature of objects that are manipulated by a TAG. The only assumption we will make about these objects is that the elementary objects of the grammar give a sufficiently enlarged domain of locality that allows localization of statements of dependencies such as subcategorization, and filler-gap.

2.2.1 Quasi-Trees. Let us reconsider α_1 (shown in Figure 1), which is assumed to be one of the tree structures associated with a transitive verb. Let us consider which information captured in this tree is important for asserting the cooccurrence dependencies involved. We must represent the obligatory arguments required by a transitive verb. If we look at the relationship between the obligatory arguments and the anchor captured by the tree α_1, we notice that the sentence structure is formed by combining the subject NP node with a VP node. This information is often captured by a rule

(1): $S \rightarrow NP\ VP$. Also, notice that a VP that captures the combination of the lexical anchor with the other obligatory arguments must be formed. In the case of a transitive verb, such information can be captured by a rule (2): $VP \rightarrow v\ NP$. Thus we can see that the essential information captured by α_1 includes the simultaneous use of the two rules and can be described by stating the relationships between the six entities (three for each rule) involved in the rules.

These relationships can be stated by means of some assertions about the individual entities. At this point it is useful to use some names (identifiers) to refer to these entities.[1] Let these names be s_1, np_1, vp_1 (for the three symbols in rule (1)) and vp_2, v_2, np_2 (for the three symbols in rule (2)). The assertions given below (that can be captured by the structural representation, α_4, given in Figure 4) can be stated to minimally describe a structure anchored by a transitive verb.

1. The label of the entity referred by s_1 is S. It *immediately dominates* the entities referred to by np_1 and vp_1. np_1 and vp_1 correspond to the occurrences of NP and VP in the right-hand side of rule (1) and hence the immediate domination.
2. np_1 refers to the subject of v_2 and is labeled by the symbol NP. It is one of the obligatory arguments required by the anchor.
3. vp_1 is labeled by VP and is used to indicate the combination with the subject (i.e., np_1) to yield a sentence.
4. vp_2 (also labeled VP), corresponds to the occurrence of VP in the left-hand side of rule (2). It immediately dominates v_2 and np_2. vp_2 is used to indicate the result of the combination of the transitive verb with the obligatory object argument (given by np_2).
5. Since the combination of the anchor with the subcategorized arguments (given by np_1 and np_2) will yield a sentence, the s_1 dominates v_2 by a path of length at least two. Furthermore, the nodes named vp_1 and vp_2 lie on the path from the v_2 node to the s_1 node. Since vp_1 must dominate v_2, we can conclude that the node named vp_1 must *dominate* the node named vp_2 (indicated by a dashed link in α_6) and thus, in turn the v_2 node. *Immediate domination*, on the other hand, is represented in the usual fashion.

Here we define the domination relation to be reflexive (i.e., a node dominates itself) in addition to being transitive and antisymmetric. Therefore, we are not stating that the nodes named as vp_1 and vp_2 are necessarily different. Notice that the above assertions have been made independent of TAG or the commitment to use trees for the elementary objects. In TAG, given the decision to use trees, a (minimal) tree that satisfies these assertions will be used. It is due to this minimality requirement that the nodes named as vp_1 and vp_2 are assumed to be the same.

On the other hand, the only decision we have committed to is to use structures large enough to localize subcategorization. In this case, we have given some assertions that describe the structure for simple declarative sentences anchored by a transitive verb. Although compatible (though different) assertions have been made about the

1 We adopt this practice of naming nodes following D-theory. This choice to incorporate ideas from D-theory arose from an observation made by S. M. Shieber.

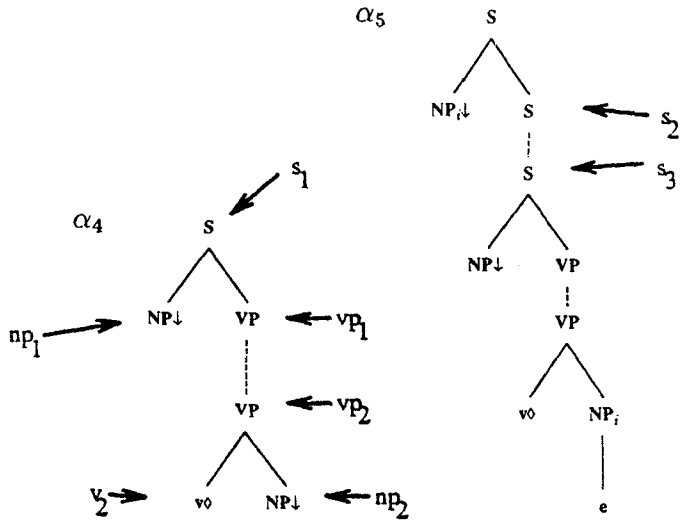


Figure 4
The domination relations.

nodes referred by vp_1 and vp_2, (from these assertions) we cannot conclude whether these nodes are different or are the same node. In fact, this is the reason that structures such as γ_1 (which represents the case where the two are different) in Figure 3 as well as α_1 (where vp_1 and vp_2 both refer to the same node), given in Figure 1, can both be derived. The structure given by α_4 (with the dashed link indicating possible separation) partially describes the phrase structure tree for both cases. Since vp_1 and vp_2 can both refer to the same node, to avoid confusion, henceforth we will call them **quasi-nodes**. Thus a node such as the *VP* node in α_1 (Figure 1) is represented by a pair of quasi-nodes in α_4. We will refer to these quasi-nodes as the top (for example, vp_1) quasi-node and the bottom (vp_2) quasi-node. Structures such as α_4 will be called **quasi-trees** to indicate that they are not trees but (partial) descriptions of trees.

A second example that also motivates the proposed interpretation of TAG where the elementary objects are taken to be descriptions of trees (quasi-trees with domination and immediate domination links) rather than trees involves the tree structure in the case of topicalization. The topicalization of the object of a transitive verb can be described by the quasi-tree α_5 (in Figure 4), which is the counterpart of α_2 (Figure 1) used traditionally in TAG. If the elementary structures in a TAG are supposed to depict the localization of dependencies such as those arising due to subcategorization and movement, then we claim structures like α_5 are indeed the appropriate structures to consider. For instance, no treatment of topicalization can justify the identification of the nodes referred to by s_2 and s_3. Thus, a pair of quasi-nodes is appropriate for their representation. As in the case of vp_1 and vp_2 quasi-nodes in α_4, one can only claim that s_2 quasi-node dominates s_3 quasi-node (again, by domination, we also allow for the possibility that s_2 and s_3 can refer to the same node). It may be interesting to contrast this lack of information in α_5 (whether or not they refer to the same node) with the use of functional uncertainty in LFG (Kaplan and Maxwell 1988) to account for long-distance dependency.

In order to consistently maintain the distinction between descriptions of trees with trees, while discussing the proposed interpretation of TAG we will use the terms

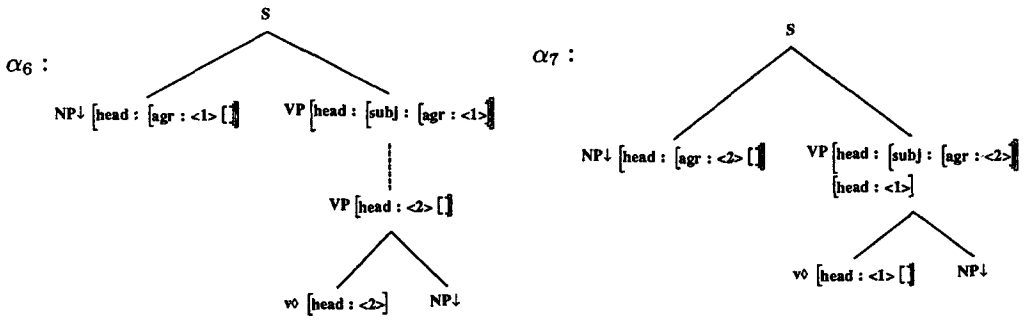


Figure 5
Associating feature structures with quasi-nodes.

initial quasi-tree, **auxiliary quasi-tree**, **quasi-root**, and **quasi-foot** in place of initial tree, auxiliary tree, root node, and foot node, respectively.

2.3 Associating Feature Structures with Quasi-Nodes

Let us now consider α_4 given in Figure 4 and the pair of *VP* quasi-nodes. In the version of FTAG formalism we define here, the feature structure that we associate with quasi-nodes simply reflects the assertions that we make about them. For instance, suppose a constraint $VP.head.subj.agr = NP.head.agr$ was used in conjunction with the rule $S \rightarrow NP\ VP$; and the constraint $VP.head = v.head$ was used with the rule $VP \rightarrow v\ NP$. These two rules (and associated constraint equations) when used together produce α_6, shown in Figure 5. Notice that the feature structure associated with a top quasi-node can be considered as constraints on it (and hence a constraint on the nature of tree that is rooted at this quasi-node) that are made on the basis of its ancestors and siblings. Similarly, the feature structure associated with a bottom quasi-node reflects the nature of tree that is rooted at this quasi-node (that is its descendants).

Instead of explicitly using a pair of quasi-nodes and drawing the domination (dashed) link between them, we can also depict it in a more traditional manner found in TAG literature (see α_7 in Figure 5). In such a case a node, such as the *VP* node in α_7, will have two feature structures (the ones associated with the two quasi-nodes) associated with it. This matches the previous definition of feature structure–based Tree Adjoining Grammars where these two feature structures were called the top and bottom feature structures associated with a node. In fact, this correspondence was independently observed by Henderson (1990) and was used in the translation of an FTAG to a Structure Unification Grammar. When convenient, we will use "a node with two associated feature structures" instead of "a pair of quasi-nodes (with one feature structure associated with each quasi-node)."

If the objects manipulated by a TAG are considered as quasi-trees, a natural question arises when one considers what would be a node in a tree as a pair of quasi-nodes. For our current purposes, this aspect is not relevant. For instance, the auxiliary quasi-trees, $\beta_3, \beta_4, \beta_5, \beta_6$ in Figure 6, are equally acceptable (well-formed structures) in the formalism. No matter which one is used, for an auxiliary quasi-tree, we have to state the quasi-root node and the quasi-foot node. As shown in Figure 6, for the auxiliary quasi-trees, $\beta_3, \beta_4, \beta_5, \beta_6$, they are given by the pairs of names vp_3, vp_4; vp_5, vp_6; vp_7, vp_8; and vp_9, vp_{10}, respectively.

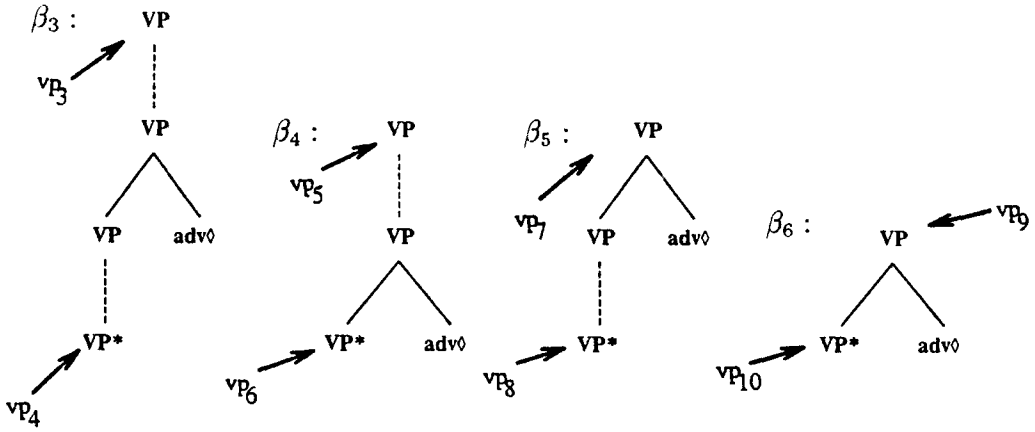


Figure 6
Some possible auxiliary quasi-trees.

2.4 The Adjoining Operation

We will now define the adjoining operation on quasi-trees and see that (unlike previously) this operation has the property that in the resulting structure all the structural relations specified in the objects being composed are preserved. We will see that this, in turn, allows for a straightforward embedding of TAG in the unification-based framework. Recall that by considering a pair of quasi-nodes we allow for possible separation. We now define the operation of adjoining as the operation that achieves this separation. Consider a quasi-tree, as shown in Figure 7, with a pair of top and bottom quasi-nodes referred by the names, say η_t and η_b respectively. In this figure, we have deliberately chosen to indicate that feature structures (f_t and f_b) as labeling quasi-nodes (η_t and η_b respectively). Consider a auxiliary quasi-tree, β, with the quasi-root node and quasi-foot node referred by the names $root_\beta$ and $foot_\beta$ as shown in Figure 7. Adjoining β at the pair of quasi-nodes (η_t, η_b) in α is now defined by stating that the domination relation between the pair of quasi-nodes is specified to be the same domination relation that exists between the quasi-root node of β and the quasi-foot node of β. Thus, the pair of quasi-nodes (referred by the names η_t and η_b) get separated when the quasi-nodes referred by η_t (η_b) and $root_\beta$ ($foot_\beta$) are identified. Adjunction is thus defined as a pair of simultaneous substitutions.[2] Note that the adjoining operation is defined only if the point of adjunction is specified as a pair of quasi-nodes.

We stated that the feature structure associated with a quasi-node is an encoding of the assertions (in terms of feature-value pairs) that are made about it. Let f_t, f_b, f_{root}, and f_{foot} be the feature structures that satisfy the constraints stated about the quasi-nodes referred by $\eta_t, \eta_b, root_\beta$, and $foot_\beta$ respectively. Then, because adjunction is defined as identifying the quasi-nodes referred by η_t and $root_\beta$ (as well as those referred by η_b and $foot_\beta$), f_t and f_{root} (f_b and f_{foot}) satisfy the constraints expressed about the same quasi-node. Thus they have to be unified (as shown in Figure 8). Note that we have motivated f_t as a reflection on the constraints about the tree below the corresponding

2 In this way, we have defined adjoining to make it compatible with the traditional definition of adjoining. Recall that in Section 1.2 adjoining in TAG was defined as substituting the auxiliary tree at the node of adjunction with the excised subtree being substituted at the foot node. In Section 5, we look more closely at the definition of adjoining, in particular whether the definition can be made to follow from linguistic principles or stipulations, i.e., whether adjoining is a derived operation and not a fundamental one.

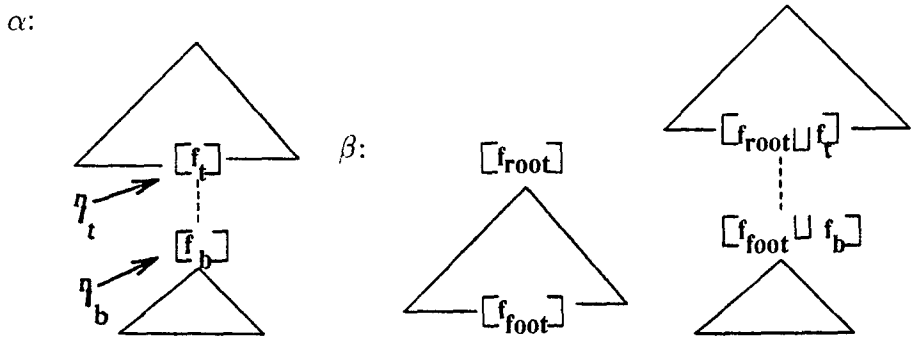


Figure 7
The adjoining operation in FTAG.

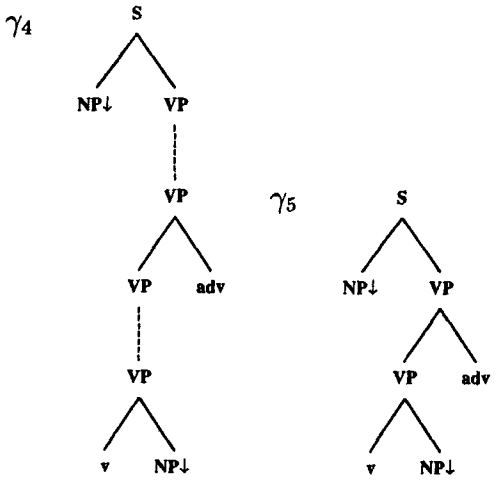


Figure 8
Examples of the adjoining operation.

quasi-node, one that possibly arises due to the relationship of this quasi-node with its ancestors and siblings in γ. The feature structure f_{root} also reflects the nature of the tree below the quasi-root node. Since these two quasi-nodes are now required to be the same, the unification of f_t with f_{root} gives a feature structure that reflects all constraints when the quasi-nodes are identified.

Figure 8 shows the result of adjoining at the paired *VP* quasi-nodes in α_4 by the auxiliary quasi-trees β_3 (resulting in γ_4) and β_6 (γ_5).

2.5 The Substitution Operation

The substitution operation used in TAG is the same as that used in context-free grammars (CFGs), where one considers a CFG as a tree-rewriting formalism rather than a string-rewriting formalism. In this case, given two trees, the substitution operation can be defined as the tree obtained by identifying the root node of one tree with the target

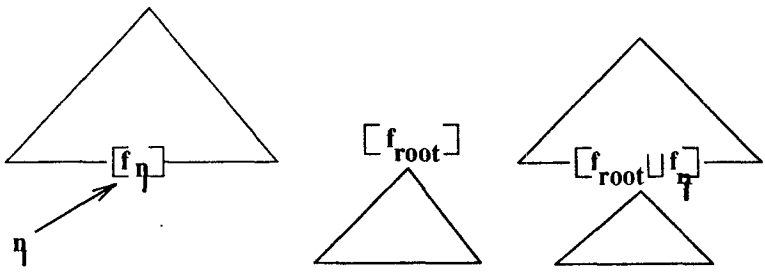


Figure 9
The substitution operation.

(of the substitution operation) node appearing in the frontier of the other tree. Due to this identification, the feature structures associated with the two nodes in question are unified.

A similar definition will be used to define the substitution operation here. Let η refer to a quasi-node in the frontier (see Figure 9). The substitution of γ at the quasi-node η is defined as the quasi-tree obtained by identifying the quasi-nodes η and the quasi-root of γ. Thus the feature structures associated with these quasi-nodes get unified as shown in Figure 9.

2.6 Some Observations

We can make the following observations at this stage. The dashed link between a pair of quasi-nodes indicates that it is possible for the two to be the same. However it is possible to insist that such a pair of quasi-nodes are distinct. This is possible, by stating incompatible assertions about them. On the other hand, as was noted by Marcus, Hindle, and Fleck (1983), without explicitly stating so, we cannot make assertions about a pair of quasi-nodes that will indicate that they are the same. These observations will be further elaborated in Section 3 to capture obligatory adjoining (*OA*) constraints.

Another point that can be noted is that the adjoining operation and its use of auxiliary trees can itself be motivated from the definition of quasi-trees. Notice that we have introduced the concept of quasi-trees simply from the motivation of considering structures with enlarged domains of locality in order to localize dependencies such as subcategorization and filler-gap. In defining quasi-trees we stated that pairs of quasi-node can be separated (i.e., they need not be the same node). If a pair of associated quasi-nodes are to be separated by the use of a composition operation, it is easy to see that it can only be done by an operation like adjunction, and the kind of structure that can fit between them must have the general form of an auxiliary tree. Of course, with the use of the new notation, the insistence that the root and foot nodes (of auxiliary trees) be labeled by the same nonterminal symbol (as well as for the target of adjunction) is only a *stipulation* (and not required by the formalism). Let us consider the labeling of nodes (quasi-nodes) by atomic symbols (such as *S*, *NP*). In contrasting the traditional definition of TAG with the definition given here, suppose we make a correspondence between a node in a tree (using the traditional definition) with a pair of quasi-nodes in a quasi-tree. It must be the case that such a pair of quasi-nodes are labeled by the same atomic symbol (since they correspond to a single node according to the traditional definition of TAG). Proceeding with the assumption that pairs of quasi-nodes are labeled by the same symbol we note from the definition of adjoining given in Section 2.4, it follows that for any auxiliary tree to be adjoined at this node,

the quasi-root of this auxiliary tree and the quasi-foot must also be labeled the same. Thus the above-mentioned stipulation is a statement that recursion is factored out of elementary trees. In fact, as we will see, if instead of nonterminal symbols we consider category structures (as specified in GPSG) as labeling nodes then almost all pairs of quasi-nodes in the trees we will consider here will be labeled differently (by compatible or incompatible categories). In fact, it is the relationship between the two labels that will determine the subset of auxiliary trees that can be adjoined at a node. Further discussion on this matter can be found in Section 3.

In the definition here, since we do not start by assuming that trees are composed, there is no need to make such an assumption that a pair of quasi-nodes separated by the domination (dashed) link must be labeled the same, unless if it follows from some linguistic principle/intuition being expressed using the TAG formalism. At this point we would like to note that enforcing such a stipulation has significant consequences on the definition of the formalism. Some of these consequences are noted in Section 5, where we contrast multi-component adjoining with adjoining.

2.7 Objects Derived by a Grammar

We have stated that an FTAG grammar *manipulates* (partial) descriptions of trees (i.e., quasi-trees). We will now state that a grammar *derives* trees (with nodes labeled by feature structures).

The composition operations of adjoining and substitution compose quasi-trees to build more complex (and more specific) quasi-trees. Each quasi-tree obtained during the derivation process specifies a set of trees. The set of trees *derived* can be obtained by taking the *circumscriptive* reading of the domination relation indicated in the quasi-trees obtained. The domination link between a pair of quasi-nodes represents the situation that they may or may not refer to the same object. In the absence of further information (for instance at the end of the derivation process) we shall consider that the pair of quasi-nodes refer to the same (single) node. Thus, given a quasi-tree, its minimal reading leads to the derived tree that is obtained by explicitly equating the related top and bottom quasi-nodes for each pair of quasi-nodes (since by the domination relation specified here any quasi-node dominates itself). Thus, in a derived tree (such as α_2, in Figure 1, obtained by taking the circumscriptive reading of the domination relationship specified by the quasi-tree α_5 given in Figure 4) only one feature structure is associated with each node.

The discussion given above justifies the unification (or coindexing) of the top and bottom feature structures of a node at the end of the derivation process as specified in the previous definition of FTAG. Of course, due to the associativity of the unification operation, the coindexing of the top and bottom feature structures for all nodes does not have to be delayed until the end of the derivation process. Such unifications for a node can be done whenever one decides that there will be no more adjunctions at that particular node.

In the traditional definition of TAGs, a derived tree cannot have nodes with *OA* constraints, even though intermediate trees can have nodes with *OA* constraints. This requirement on derived trees is analogous to the use of ANY in FUG. In our current definition a tree is derived (in the above-mentioned manner) only if the corresponding quasi-tree has compatible feature structures associated with each pair of quasi-nodes. If this were not the case, i.e., some pair of quasi-nodes had incompatible feature structures associated with them, then taking the circumscriptive reading of the domination relation will not be possible. Such quasi-trees do describe a set of trees, but the one obtained by equating the pairs of top and bottom quasi-nodes is not one of them. Obvi-

ously this should be the case, since incompatible assertions about a pair of quasi-nodes indicates that they do refer to different nodes (and hence specify *OA* constraints).

2.8 Using One (Rather than Two) Feature Structure

A question arises whether (as in standard CFG-based unification grammars) one could associate just one (rather than two) feature structure per node, i.e., whether it is necessary to consider pairs of quasi-nodes. In fact, Harbusch (1990) defined such a treatment of TAG where only one feature structure is associated with each node.

One could argue that it may be inefficient (for instance, when implementing the formalism as defined here) to start with the pairs of quasi-nodes and then try to merge them eventually when possible. Strategies to improve processing may be considered particularly if we believe that, on an average, a relatively small proportion of potential sites will become actual targets of adjunctions during a derivation of a sentence. Then (to improve performance) we could specify that *by default* the associated pair of top and bottom quasi-nodes are to be identified. That is, we will not consider a node as a pair of quasi-nodes unless there is reason to believe it is necessary (if adjunction has to be performed). So we can even state that there is just one feature structure per node, which has to be the one obtained by unifying the feature structures associated with the top and bottom quasi-nodes. Now if adjunction takes place at a node in some tree that has been derived, then the "unification" that has been performed has to be undone to recover the top (relating it with its ancestors and siblings) and bottom (based on the structure it dominates) feature structures. This undoing can be quite complex, especially if the pair of quasi-nodes in question is a part of a derived object rather than an elementary structure specified by the grammar. The above description essentially captures the definition of the formalism presented by Harbusch (1990).

Another point can be made about the scheme presented above. Consider a node whose top and bottom feature structures are incompatible and hence nonunifiable. If we were to insist that only one feature structure were to be associated with every node then we can only unify the compatible parts of the top and bottom feature structures and somehow (perhaps with the use of a device like ANY) retain (effectively) the OA constraint machinery.

3. Feature Structures and Adjoining Constraints

In the traditional definition of a TAG, the adjoining possibilities at a node is determined by the association of adjoining constraints with each node. In this section we consider how such constraints may be captured by the use of feature structures and then contrast the two methods of determining the adjoining possibilities. Since we attempt to contrast the adjoining possibilities, in this section we will make correspondences between nodes in trees (used in the traditional definition of TAG) with pairs of quasi-nodes that are linked by the domination (dashed) link. That is, we talk of such pairs of quasi-nodes as the target of adjunction. Also, if we have a pair of quasi-nodes given by η_1 and η_2 where η_1 quasi-node dominates η_2, we will say that the η_2 is the bottom quasi-node paired with η_1 or that η_1 quasi-node is the top quasi-node paired with η_2.

3.1 *OA* Constraint

In the definition of TAG, given in Section 1.2, it was stated that if a node has an OA constraint, then adjoining is mandatory at that node. In terms of quasi-nodes this means that the corresponding pair of quasi-nodes must be separated. Therefore, the use of an OA constraint at a node may be interpreted as stating that the related pair

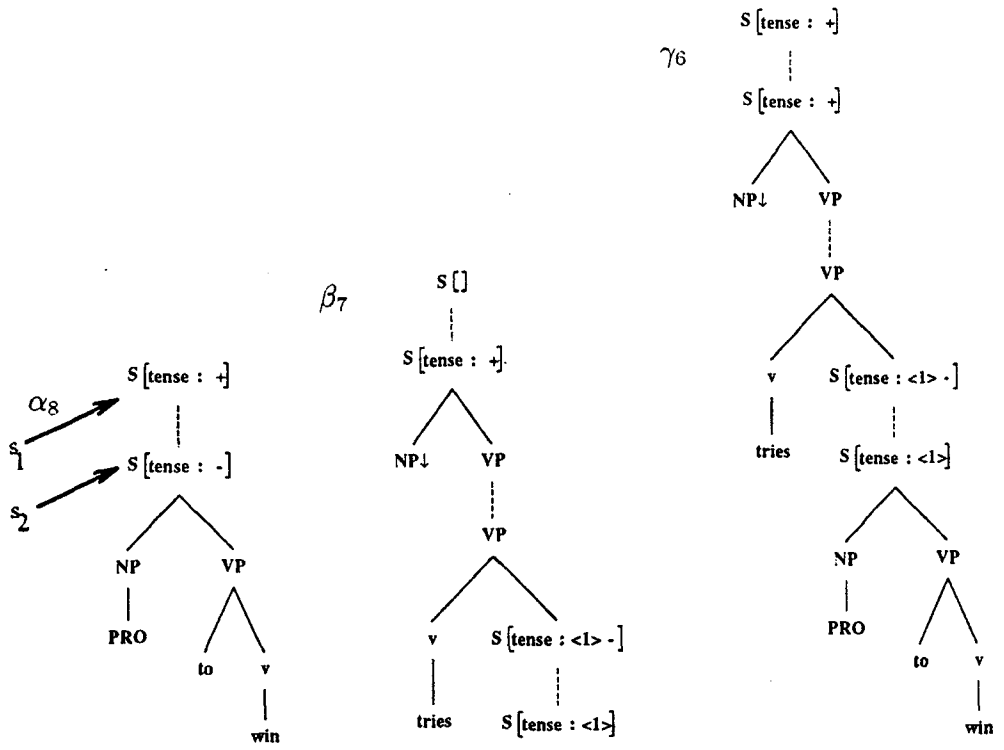


Figure 10
"OA" constraints.

of quasi-nodes are indeed distinct, i.e., there must be some feature that distinguishes them. Hence the linguistic basis for making the claim that the node has an OA constraint must be stated in such a way that the feature structures on the two quasi-nodes are incompatible. As an example, consider α_8 given in Figure 10. The feature structure of the quasi-root of α_8 has a value of + for the *tense* attribute to specify that any tree rooted at this quasi-node must satisfy the constraint that it describes a tensed sentence. On the other hand, the feature structure of the paired bottom quasi-node has a value of − for the *tense* attribute since it only reflects the descendants. Since these two feature structures are incompatible, this pair of quasi-nodes has an "OA constraint" (since it is not possible to stop the derivation process and identify the top with the bottom quasi-node). However, γ_6 that results from the adjoining of β_7 does not have any pair of quasi-nodes with an "OA constraint."

3.2 *SA* Constraints

Recall that an SA constraint of a node lists a subset of auxiliary trees that can be adjoined at this node. The definition of adjunction used here is stated in terms of a pair of substitutions (and thus adjunction involves two unifications). In terms of quasi-trees, we allow the "SA" constraints to be determined as a consequence of the unifications required by identifications of quasi-nodes. If an auxiliary quasi-tree cannot be adjoined at a pair of quasi-nodes, then it must be the case that there is an incompatibility among the relevant pairs of feature structures that we unify when we attempt adjunction. When we attempt adjunction the feature structure of the top quasi-node (in the pair

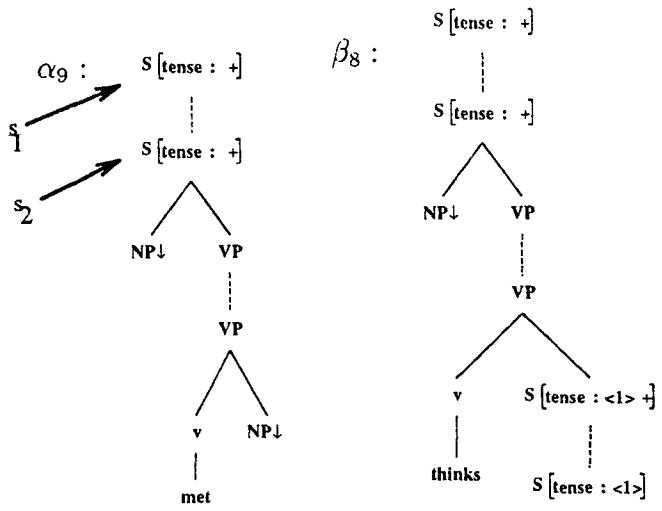


Figure 11
"SA" constraints.

where adjunction is attempted) and the feature structure of the quasi-root (of the auxiliary quasi-tree) are unified, as are the feature structure associated with the bottom quasi-node (in the pair where adjunction is attempted) and the feature structure of the quasi-foot (of the quasi-tree being adjoined). If at least one of these unifications fails then adjunction is not possible.

Consider β_8 given in Figure 11. This quasi-tree cannot be adjoined at the pair (s_1, s_2) in α_8 (Figure 10) but can be adjoined at the pair (s_1, s_2) of α_9. On the other hand, we saw that β_7 can be adjoined at the pair (s_1, s_2) of α_8. Thus we can say that the pair (s_1, s_2) of α_8 has an "SA constraint" that includes β_7 but not β_8.

3.3 *NA* Constraints

Recall that a node with an NA constraint cannot be the target of an adjunction. Traditionally, this is specified by stating that the set of auxiliary trees that can be adjoined at such a node is the empty set. For this reason, it is often stated that NA constraints are special form of SA constraints.

There are two possible ways of interpreting "NA constraints" in the quasi-tree framework. Firstly, a pair of quasi-nodes with an "NA constraint" may be interpreted as a *stipulation* that insists that no quasi-tree can be adjoined at this pair; a statement made regardless of the nature of the auxiliary quasi-trees in the grammar. This may for instance be made if we wish to allow only certain derivation sequences. One could argue that the reason for insisting that foot nodes of *complement*[3] auxiliary trees have NA constraints, as is the case in most TAG accounts, is to avoid certain derivation sequences (Kroch and Joshi 1985).

On the other hand, we may also interpret the association of "NA constraint" with a pair of quasi-nodes as a statement that none of the auxiliary quasi-trees in the grammar matches the requirements of the type of auxiliary quasi-trees that can be

3 A complement tree (for example, the tree β_2 in Figure 2) is one where the foot node corresponds to one of the arguments required by the anchor of the tree.

adjoined at this pair (as determined by the associated feature structures). Unlike the previous case, adjunction is not barred per se. Instead, attempting to adjoin at such a pair will never yield well-formed structures. This is because of the nature of such a pair and of the auxiliary quasi-trees in the given grammar. In the TAG formalism, both these interpretations are captured by the same *operational* mechanism.

The first kind of NA constraint is easily stated. According to this interpretation, for each pair of quasi-nodes with an "NA constraint," the two quasi-nodes are indeed the same node (since we are stating that there is no possible separation). Since the two quasi-nodes are to be identified, the feature structure associated with the resulting quasi-node must reflect both the relationship of the quasi-node with its ancestor (which we assume stands for the top feature structure) as well as its relationship with its descendants (the bottom feature structure).

Earlier we had stated that the target of an adjunction operation must be a pair of quasi-nodes that have not been identified (i.e., merged). Suppose that a pair of quasi-nodes (η_1, η_2) were merged. Let the quasi-root and quasi-foot of some auxiliary quasi-trees β be given by r and f. Adjoining β at the pair given by η_1 and η_2 (after they have been identified) will result in the identification of η_1 with r and η_2 with f and thus r with f. If we stipulate that in all auxiliary quasi-trees, the quasi-root and quasi-foot do not refer to the same node (i.e., the quasi-root *properly* dominates the quasi-foot), then no adjunction can occur at a pair of quasi-nodes that have been identified. Thus the identification of a pair of quasi-nodes captures "NA constraints" of the first kind.

As far as the second kind of "NA constraints" is concerned, we note that it is only a specific case of "SA constraints." Therefore, given a pair of quasi-nodes, if the associated feature structures are such that no auxiliary quasi-tree can be adjoined at this pair then it has an "NA constraint" (of the second kind). However, because of the nature of feature structures (in that they capture only partial information), it is hard to detect if a pair of quasi-nodes has such an "NA constraint." In Section 3.4, we will consider such an example.

3.4 Comparing the Implementation of Adjoining Constraints

In the TAG formalism, selective adjoining constraints are specified by enumeration, and hence are stipulations stating which trees can be adjoined at a node. Hence, specifying adjoining constraints in such a way is not a linguistically appealing solution. Obviously, such stipulations are needed because the information content of the labels of nodes in a TAG is often insufficient to determine the trees that can be adjoined at various nodes. In the case of FTAG, labeling of quasi-nodes by symbols such as NP, S is only a part of information contained in the feature structures associated with them. We associate with a pair of quasi-nodes feature structures that describe the features of the top and bottom quasi-nodes. The fact that only appropriate quasi-trees get adjoined is a corollary of the fact that only those consistent with these declarations are acceptable. Additionally, in a FTAG, "adjoining constraints" can be dynamically instantiated and are not pre-specified as in a TAG.

We will now point out some differences between the implementation of adjoining constraints in TAG and FTAG that arise because of different methods adopted in adjoining constraint specification. Of course, if the constraints are prespecified as in TAG, then little work has to be done (say by a parser) to verify whether an auxiliary tree can be adjoined at a node during the derivation process. This is not the case in FTAG, because of dynamic instantiation of "constraints" in FTAG. For example, instead of β_7 (Figure 10), suppose we consider β_9 shown in Figure 12. The result of adjoining β_9 at the pair (s_1, s_2) of α_9 is γ_7. There is a pair of quasi-nodes, (s_3, s_4), in γ_7 with values of $-$ and $+$ for the *tense* attribute (thus giving rise to "OA constraints").

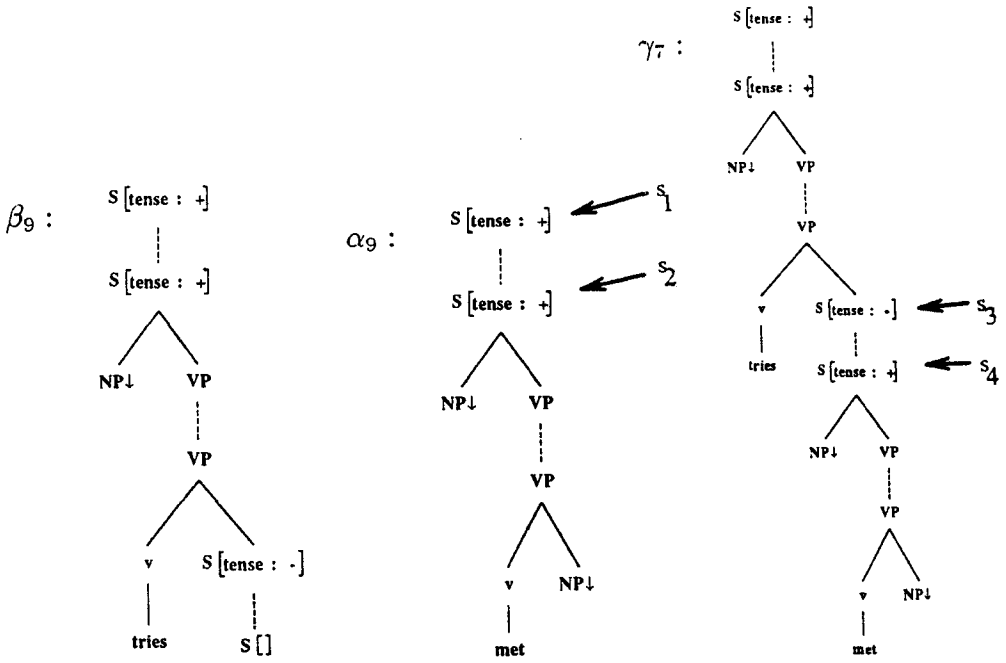


Figure 12
Comparison of adjunction constraints—Example 1.

In a TAG grammar, the SA constraints at the root of tree corresponding to α_9 would be given to disallow this adjunction. In the case of FTAG, as shown in Figure 12, this adjunction is allowed, because the associated unifications did not fail. Now suppose (as one might expect) the auxiliary quasi-trees in the grammar were such that none of them had their quasi-root with a feature structure compatible with *tense*: − and quasi-foot with a feature structure compatible with *tense*: +. In this case, although the adjunction of β_9 was permitted, no tree can ever be derived from the result of adjunction. In fact, until we try all possible adjunctions at the node η in γ_7, we cannot realize that adjunction of β_9 at the root of α_9 can result in a final acceptable tree. Thus, the pair (s_3, s_4) has an NA constraint of the second kind.

Now we will consider an example where specification of constraints in TAG suffers in comparison with the implementation of "constraints" in FTAG. Consider the following well-formed sentences

(1) Who did John see?

(2) Who did Peter think John saw?

(3) I wonder who John saw.

(4) I wonder who Peter thought John saw.

(5) Peter thought John saw Mary.

and the following, which are not well-formed sentences.

(6) Who John saw?

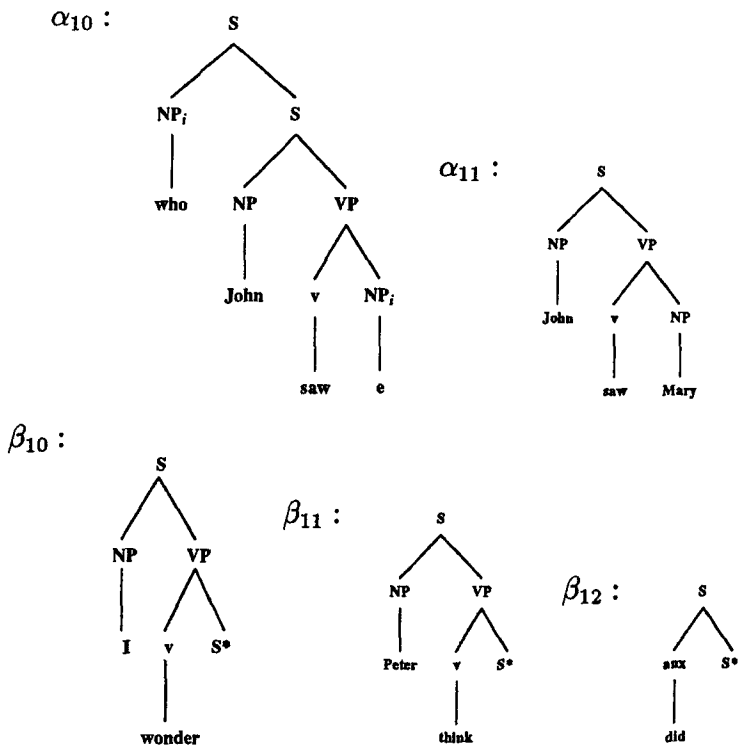


Figure 13
Comparison of adjunction constraints—Example 2.

(7) I wonder who did John see?

(8) Who Peter thought John saw.

We will first consider a TAG account (in traditional style). The trees (without considering adjoining constraints) given in Figure 13 have been suggested in literature to account for the well-formed sentences above. We have drawn these trees accounting for substitution at the *NP* nodes.

From the well-formedness of (1) and ill-formedness of (6) it follows that the node η of α_{10} must have an OA constraint with β_{12} in its SA constraint. On the other hand, from the well-formedness of (3) and ill-formedness of (6) it follows that the root of α_{10} must have an OA constraint with β_{10} in its SA constraint. However, the requirement of an OA constraint on these two nodes in α_{10} is mutually exclusive. Because of this, a TAG grammar that accounts for the sentences above must have two trees, that have exactly the same tree structure but only differ in the adjoining constraints attached at the nodes.

Now, from the well-formedness of (5), which can be derived by adjoining β_{11} at the root of α_{11}, we can conclude that there need not be an OA constraint on the root of β_{11}. However, suppose we adjoin β_{11} at the node η in α_{10} such that the frontier matches with (8). From the ill-formedness of (8) and the well-formedness of (2) we realize that there must be an OA constraint on the root of β_{11} with β_{12} in its OA constraint. Thus,

again we will need two trees (corresponding to β_{11}), with identical tree structure but differing in the adjoining constraints.

We will see that such replication of tree structure is not necessary. Now consider the FTAG fragment (inspired by similar treatment in Abeille [1991]) given in Figure 14. If the feature structures of s_1 and s_2 quasi-nodes of α_{12} are unified then the other pair

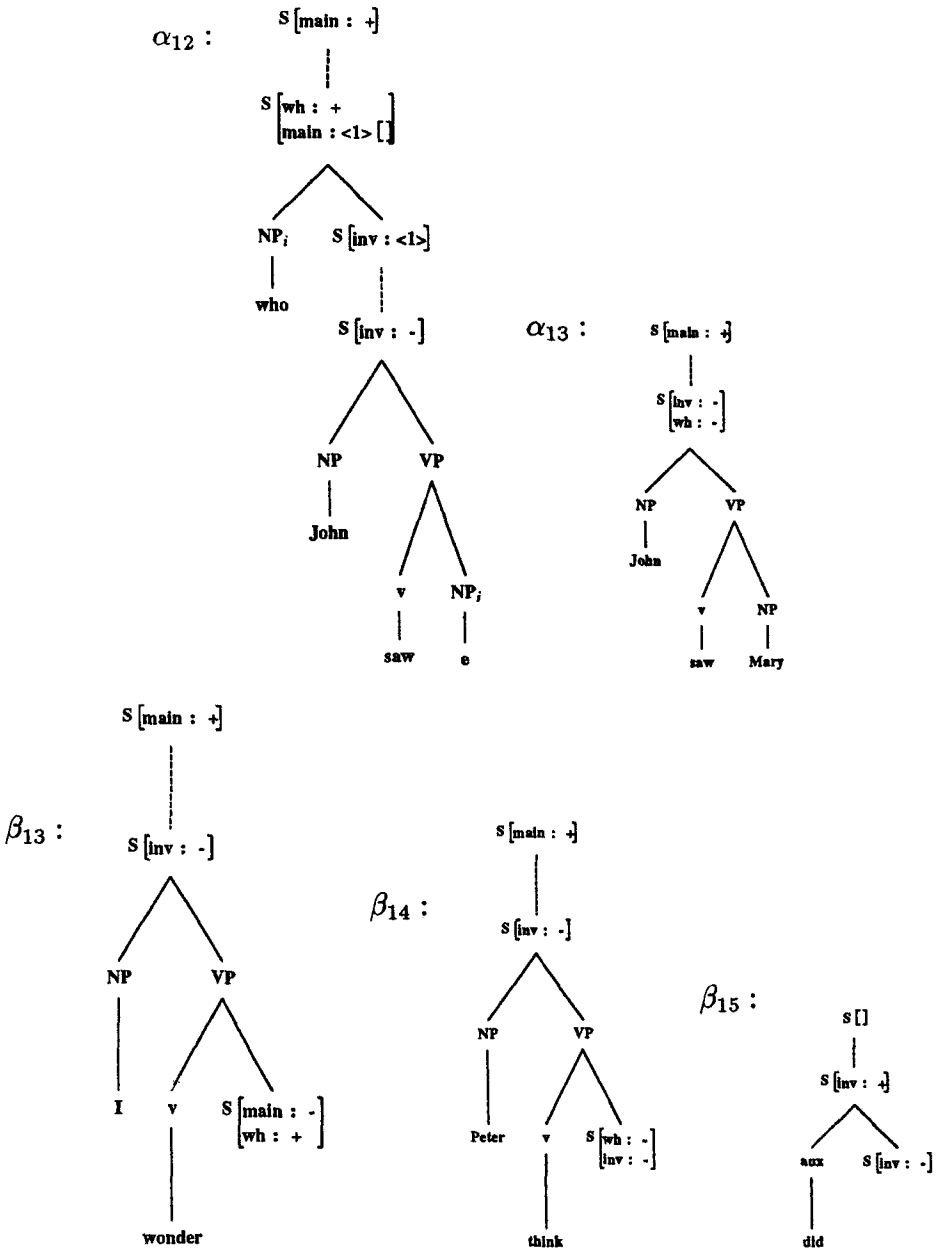


Figure 14
Comparison of adjunction constraints—Example 2.

of quasi-nodes labeled S will obtain an "OA constraint" and vice versa as required (hence (6) can not be derived). In fact, if β_{13} were adjoined at the root of α_{12} (and thus showing (3) is well formed) then it will no longer be possible to derive (7). Likewise, by adjoining β_{15} at the pair of s_3 and s_4 quasi-nodes in α_{12}, we can derive (1) but will no longer be able to derive (7).

Proceeding in this manner we can show the well-formedness of (1)–(5) and the ill-formedness of (6)–(8). Thus we have shown that if appropriate assertions can be stated about the individual nodes then a more succinct grammar can be given: one that does not require replication of tree structures, due to the fact that adjoining constraints are not pre-specified as in a TAG.

4. A Logical Formulation

A central theme in our definition of FTAG has been the view that the objects manipulated by a grammar are descriptions of trees (rather than trees). This separation of descriptions of trees from the trees (models) derived has been crucial in embedding TAG in the unification framework. The question of which language to use to describe trees (together with its semantics) arises. We have used quasi-trees (as the descriptions themselves) in order to focus on TAG, and have not introduced some general formal framework for describing trees. The discussion below does not constitute a suggestion about how such general descriptions may be given, but is one way to specify an FTAG that will be convenient for our purposes here.

In this section, we describe a logical formulation of the unification-based approach to TAGs. The purpose of providing a logical formulation of FTAG is so that we can find the denotation of an FTAG grammar (the set of structures generated) as well as contrast it with context-free grammar-based unification grammars. To define the denotation of an FTAG grammar, we will first describe how an FTAG grammar can be represented. This representation uses the logical formulation of feature structures as given by Kasper and Rounds (1986) and Johnson (1988) and is similar in approach to the logical formulation of Functional Unification Grammar (FUG) given by Rounds and Manaster-Ramer (1987).

In the framework of Rounds and Manaster-Ramer (1987), an FUG (or any context-free grammar with associated unification equations as in, say PATR-II) can be represented by means of a set of equations, using the formulae of Kasper–Rounds to represent feature structures. For example, a context-free grammar rule $S \rightarrow NP\ VP$ can be represented as $s ::= CAT : S \wedge 1 : np \wedge 2 : vp$. Here s, np, and vp are *type variables*. The attributes 1 and 2 are used to indicate the first and second children respectively. Using standard techniques to derive fixed points from a set of recursive rules, the denotation of type variables are obtained. The denotation of the type variables gives the set of structures derived from the corresponding nonterminals.

Now suppose we wish to express reentrancy in feature structures by using variables; it is clear that we have to use *individual variables* and not type variables. As in Johnson (1988), we use *individual* variables and equalities to express reentrancy. The syntax we adopt to describe attribute-value structures is as follows. Firstly, the set of terms is defined as

t::=	a	where a is an atomic value
	x	where x is an individual variable
	$l(t_1)$	where l is a label (or attribute) and t_1 is a term.

The set of formulae is defined as

$$\begin{array}{lll} \phi ::= & t_1 \approx t_2 & \text{where } t_1, t_2 \text{ are terms} \\ & \phi_1 \wedge \phi_2 & \text{where } \phi_1, \phi_2 \text{ are formulae} \\ & \phi_1 \vee \phi_2 & \text{where } \phi_1, \phi_2 \text{ are formulae.} \end{array}$$

For example, $(l(x) = y) \wedge (h(x) = z) \wedge (g(y) = z) \wedge (z = a)$ describes (among others) the following feature structure.

$$\begin{bmatrix} l & : & g & : & \boxed{1} \\ h & : & \boxed{1} a & & \end{bmatrix}$$

Note that individual variables (that stand for individual feature structures) are being used to capture reentrancy, whereas typed variables play a role analogous to the role of nonterminals in grammars (such as CFGs) and stand for a set of feature structures. For the purpose of describing an FTAG, we need individual variables to specify reentrancy (as well as to refer to quasi-nodes) and "typed" variables to denote the set of structures derived from elementary quasi-trees. To distinguish between these two kinds of variables, in our framework, we will use monadic predicate instead of typed variables.

4.1 Expressing an FTAG

Firstly, we note that quasi-initial trees are analogous to nonterminals in CFGs. Thus, as indicated above, quasi-initial trees will be represented by monadic predicates. If α is a quasi-initial tree, then we will use a predicate symbol $\overline{\alpha}$ to represent this quasi-tree. If a structure $\mathcal{A}$ is derivable in the grammar starting from α then we would like to have $\mathcal{A}$ to belong to the set denoted by $\overline{\alpha}$. For example, any structure described by α_{14} can be assumed to satisfy the requirements on the variable x in

$$cat(x) \approx S \wedge Dom(x, y) \wedge cat(y) \approx S \wedge 1(y) \approx z \wedge count(y) \approx zero \wedge cat(z) \approx c.$$

This description is intended to not only describe the features of nodes, but also the structure of the subtrees rooted at each node (with attributes $1, 2, \ldots$ used to specify the first, second, ... child of a node). In the formula given above, x represents the quasi-root node. Therefore, we will define $\overline{\alpha}_{14}$ by

$$\overline{\alpha}_{14}(x) \Longleftrightarrow cat(x) \approx S \wedge Dom(x, y) \wedge cat(y) \approx S \wedge 1(y) \approx z \wedge count(y) \approx zero \wedge cat(z) \approx c.$$

In this case, $Dom(x, y)$ is used to indicate that the quasi-root (x) dominates the associated bottom quasi-node (given by y).

Now if we view the definition of α_{14} independent of the rest of the grammar, then $Dom(x, y)$ represents domination in any arbitrary manner. However, the rest of the grammar specifies the constraints on the domination relation by defining the actual possibilities for the domination. This is because a pair of quasi-nodes (say as given by x and y in α_{14}) is intended to mean that either they are the same objects or are different nodes that are related by proper domination. In our definition, the separation can take place only by adjunction. So given a grammar, we can specify that the domination relationship is actually defined by

$$Dom(x, y) \Longleftrightarrow x \approx y \vee \overline{\beta}_1(x, y) \vee \ldots \vee \overline{\beta}_n(x, y)$$

where $\{\beta_1, \ldots, \beta_n\}$ are the quasi-auxiliary trees in the grammar. Here we assume that $\overline{\beta}$ captures the (domination) relationship between its quasi-root and quasi-foot nodes

of the quasi-auxiliary tree β. Since the actual definition of the domination between a pair of quasi-nodes is determined by the quasi-trees of the grammar, it is appropriate to consider fixed-point semantics to define the denotation of a grammar.

Before we discuss the fixed point we will complete our discussion about how we can specify a grammar. Let us define another monadic predicate *Inittree* by

$$Inittree(x) \Longleftrightarrow \overline{\alpha}_1(x) \vee \ldots \vee \overline{\alpha}_m(x)$$

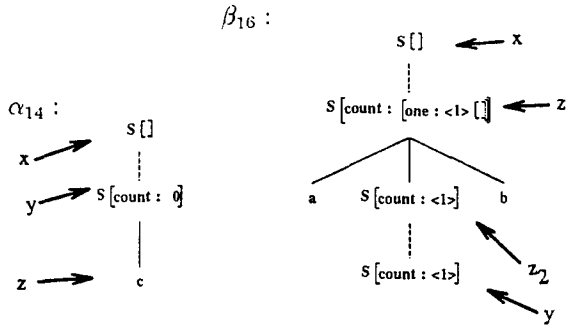
where $\{\alpha_1, \ldots, \alpha_m\}$ is the set of initial trees. If we further wish to stipulate that a structure is derived in a FTAG if it is derived from some quasi-initial tree and is rooted in S we can define

$$Grammar(x) \Longleftrightarrow Inittree(x) \wedge cat(x) = S.$$

Note that for a quasi-node (referred to as x) where substitution can take place, we can specify $Inittree(x)$ to specify the substitution.

We will now illustrate the representation of an FTAG grammar, shown pictorially in Figure 15. This grammar contains α_{14} and β_{16}. Apart from the *cat* information, the only other attribute used in the feature structures are *count* (counts the number of adjoining operations used in deriving a tree), *one* (used in counting), and attributes $1, 2, 3$ (which are used for specifying the children of a node).

To compare our representational scheme for FTAG with that for FUG given by Rounds and Manaster-Ramer (1987), we have used predicate symbols instead of type variables. The use of monadic predicates alone is sufficient to represent FUG (or actually a CFG-based unification grammar) since only "substitution" is used. Binary



$$
\begin{array}{lcl}
Grammar(x) & \Longleftrightarrow & Inittree(x) \wedge cat(x) \approx S \\
Inittree(x) & \Longleftrightarrow & \overline{\alpha}_{14}(x) \\
\overline{\alpha}_{14}(x) & \Longleftrightarrow & cat(x) \approx S \wedge Dom(x, y) \wedge cat(y) \approx S \wedge \\
 & & 1(y) \approx z \wedge count(y) \approx zero \wedge cat(z) \approx c \\
Dom(x, y) & \Longleftrightarrow & \overline{\beta}_{16}(x, y) \vee (x \approx y) \\
\overline{\beta}_{16}(x, y) & \Longleftrightarrow & cat(x) \approx S \wedge Dom(x, z) \wedge cat(z) \approx S \wedge one(count(z)) \approx count(z_2) \\
 & & \wedge 1(z) \approx z_1 \wedge 2(z) \approx z_2 \wedge 3(z) \approx z_3 \wedge cat(z_1) \approx a \\
 & & \wedge Dom(z_2, y) \wedge cat(z_2) \approx S \wedge cat(z_3) \approx b \wedge cat(y) \approx S \\
 & & \wedge count(z_2) \approx count(y)
\end{array}
$$

Figure 15
Example: An FTAG grammar and its representation.

predicates are used to capture adjunction (which is defined as a pair of substitutions) in FTAG.

4.2 Fixed-Point Semantics (Denotation of an FTAG Grammar)

As mentioned before, the set of terms is defined recursively as

t::=	a	where a is an atomic value
	x	where x is an individual variable
	$l(t_1)$	where l is a label and t_1 is a term.

However the set of formulae is now defined by

ϕ::=	$t_1 \approx t_2$	where t_1, t_2 are terms
	$P(t_1, \ldots, t_n)$	where $t_1, \ldots, t_n$ are terms and P is a n-ary predicate symbol
	$\phi_1 \wedge \phi_2$	where ϕ_1, ϕ_2 are formulae
	$\phi_1 \vee \phi_2$	where ϕ_1, ϕ_2 are formulae.

From the discussion given in the previous section any FTAG can be stated as

$$P_1(t_{1,1}, \ldots, t_{m,1}) \Longleftrightarrow \phi_1$$
$$\vdots$$
$$P_n(t_{1,n}, \ldots, t_{m,n}) \Longleftrightarrow \phi_n$$

where $\phi_1, \ldots, \phi_n$ are formulae and $t_{1,1}, \ldots t_{m,1}, t_{1,n} \ldots, t_{m,n}$ are terms such that for $1 \leq i, j \leq n$, if $i \neq j$ then the symbol $P_i \neq P_j$. Of course for describing an FTAG, monadic and binary predicates are enough.

The structures that terms denote are the finite state automata (actually equivalence classes containing such automata; for details, we refer to Moshier [1988] for a discussion about these structures) as defined by Kasper and Rounds (1986) and used in defining the satisfiability of formulae in their logic. We can give a fixed point semantics of a grammar in the standard way.

Definition

Let ρ be a function that maps each variable to an automaton. We define a *Value* function as a partial function that returns the denotation of a term (an automaton) relative to an environment (mapping variables to automata).

- $Value_\rho(x) = \rho(x)$ where x is a variable.
- $Value_\rho(a) = \mathcal{A}_a$ where a is an atom, where $\mathcal{A}_a$ is the atomic structure that corresponds to the atom a.
- $Value_\rho(l(t)) = \mathcal{A}/l$, if $\mathcal{A}/l$ is defined, where l is an attribute, t is a term and $Value_\rho(t) = \mathcal{A}$. If $Value_\rho(t)$ is not defined or $Value_\rho(t) = \mathcal{A}$ but $\mathcal{A}/l$ is not defined then $Value_\rho(l(t))$ is not defined.

Let ρ be an environment function and I be an interpretation mapping predicate symbols to their denotations, i.e., if P is a n-ary predicate symbol then I maps P to some set of n-tuples of automata. Given an interpretation function I and an environment ρ we define $\models$ in the following way.

Definition
$(I,\rho) \models \phi_1 \wedge \phi_2$ iff $(I,\rho) \models \phi_1$ and $(I,\rho) \models \phi_2$
$(I,\rho) \models \phi_1 \vee \phi_2$ iff $(I,\rho) \models \phi_1$ or $(I,\rho) \models \phi_2$
$(I,\rho) \models t_1 \approx t_2$ iff $Value_\rho(t_1)$ and $Value_\rho(t_2)$ are defined and $Value_\rho(t_1) = Value_\rho(t_2)$
$(I,\rho) \models P(t_1,\ldots,t_n)$ iff $Value_\rho(t_i)$ is defined $(1 \leq i \leq n)$ and
$\langle Value_\rho(t_1),\ldots,Value_\rho(t_n)\rangle \in I(P)$.

We now define a transformation function mapping interpretations in the following way. For some $m \geq 1$, let $P_i(t_{i,1},\ldots,t_{i,n_i}) \Longleftrightarrow \phi_i$ $(1 \leq i \leq m)$ be the grammar specification. We define the transformation function, T_G, such that given an interpretation, I, T_G returns an interpretation $T_G(I)$ given by

Definition
For all substitutions, ρ, where $Value_\rho(t_{i,j})$ is defined for $1 \leq j \leq n_i$,

$$\langle Value_\rho(t_{i,1}),\ldots,Value_\rho(t_{i,n_i})\rangle \in T_G(I)(P_i) \quad \textit{iff} \quad (I,\rho) \models \phi_i.$$

Ordering relations
We use the ordering relationship, $\sqsubseteq$, as defined by Rounds and Kasper (1986) i.e., $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ iff there is a homomorphism mapping the states of $\mathcal{A}_1$ to the states of $\mathcal{A}_2$ that preserves the transition and output functions. We extend this ordering relation to an ordering on n-tuples and state that $\langle \mathcal{A}_1,\ldots,\mathcal{A}_n\rangle \sqsubseteq \langle \mathcal{B}_1,\ldots,\mathcal{B}_n\rangle$ iff for $1 \leq i \leq n$ $\mathcal{A}_i \sqsubseteq \mathcal{B}_i$.

Among pairs of sets of n-tuples of automata, say $\mathcal{D}_1, \mathcal{D}_2$, we use the same ordering as that used by Rounds and Manaster-Ramer (1987) and state that $\mathcal{D}_1 \sqsubseteq \mathcal{D}_2$ iff $\mathcal{D}_1 \subseteq \mathcal{D}_2$. The least element among the sets of n-tuples of automata is the empty set. The ordering among interpretation functions is defined as $I_1 \sqsubseteq I_2$ iff for all predicate symbols P, $I_1(P) \sqsubseteq I_2(P)$, i.e., $I_1(P) \subseteq I_2(P)$.

Lemma 4.1.
If $I_1 \sqsubseteq I_2$, then for all environments, ρ, and formulae, ϕ, if $(I_1,\rho) \models \phi$ then $(I_2,\rho) \models \phi$.

This can be easily shown by using induction on the structure of the formula ϕ.

Theorem 4.1.
The transformation function is monotonic.

Let $I_1 \sqsubseteq I_2$. We have to show for all P that $T_G(I_1)(P) \subseteq T_G(I_2)(P)$. Let $P(t_1,\ldots,t_n) \Longleftrightarrow \phi$ be a part of the grammar specification and let $\langle \mathcal{A}_1,\ldots,\mathcal{A}_n\rangle \in T_G(I_1)(P)$. Thus, for any environment ρ such that $(I_1,\rho) \models \phi$ and for $1 \leq i \leq n$ we have $Value_\rho(t_i) = \mathcal{A}_i$. By the above lemma, we also have $I_2,\rho \models \phi$ and hence $\langle \mathcal{A}_1,\ldots,\mathcal{A}_n\rangle \in T_G(I_2)(P)$. Thus, $T_G(I_1)(P) \subseteq T_G(I_2)(P)$ and $T_G(I_1) \sqsubseteq T_G(I_2)$.

We will call an interpretation, I, *finite* if for all predicate symbols, P, $I(P)$ is a finite set.

Lemma 4.2.
For all environments, ρ, and interpretations, I, if $(I,\rho) \models \phi$ then there is a finite interpretation I_0 such that $I_0 \sqsubseteq I$ and $(I,\rho) \models \phi$.

This can be shown by a straightforward induction on the structure of ϕ, and by constructing I_0 in the obvious manner.

Theorem 4.2.
The transformation function is continuous.

This can be easily established using Lemma 4.1 and Lemma 4.2.

Since T_G is continuous, the least fixed point of T_G can be obtained as

$$\bigsqcup_{i \geq 0} T^i(I_\perp)$$

where $I_\perp$ is the least interpretation function and is given by $I_\perp(P)$ the empty set for all predicate symbols P. Let I_G be the fixed point of T_G. Then the set of structures derived by a grammar G is given by $I_G(Grammar)$, where *Grammar* is the distinguished predicate symbol as defined earlier.

4.3 Some Remarks

The logical formulation of FTAG given above is similar to the formulation of FUG and the associated semantics given by Rounds and Manaster-Ramer. This logical formulation of FUG essentially captures CFG-based unification grammars where substitution (and associated unifications) is the operation used for composition. This can be seen from their semantic treatment where type variables are repeatedly substituted for. Rather than using type variables for "nonterminals," in our formulation predicate symbols represent the nonterminals. Although "substitution" at frontier nodes can be effectively captured by Rounds–Manaster-Ramer calculus, we found it less cumbersome to express adjunction operation and FTAG in the above DCG-like style. The domination relation and adjunction operation are easily captured by using binary predicates and their substitutions. Despite these syntactic differences, the presentation of the semantics is essentially the same traditional fixed-point semantics. Not only do we capture the substitution operation, as was done in the Rounds–Manaster-Ramer calculus, but we are also able to contrast FUG (and CFG-based unification grammars) with FTAG by capturing adjoining as a pair of substitutions.

5. Some Consequences of the New Interpretation

So far we have concerned ourselves with an interpretation of TAG that is compatible with the constraint-based approach to grammars. We will now briefly discuss some possible implications that this new interpretation may have on design or development of TAG grammars. The point of this section is simply to raise certain possibilities and questions. Providing definitive answers and solutions involves exploring linguistic issues that are beyond the scope of this work.

5.1 Adjoining, Multi-Component Adjoining, and Substitution

We defined the adjoining operation as an operation that fits a structure in the gap between a pair of associated quasi-nodes. Although the nature of the adjoining operation itself has not been examined in much detail in this paper (apart from defining it in terms of quasi-nodes in a manner such that it is similar to the traditional definition), questions that arise from this work are: how different is the adjoining operation from the more commonly used substitution operation; and whether the definition of adjoining itself (as stated here) follows from some more fundamental linguistic assumptions. To motivate our arguments, we start by considering an example using the so-called multi-component adjoining.

Consider the derivation of:

(1) Which picture did you buy a copy of?

(2) Which picture did you buy a photograph of a copy of?

This form of long-distance dependency cannot be localized in a TAG if we wish to localize the predicate-argument dependencies as well (for details, see Kroch [1987]). On the other hand, an analysis has been given using a version of multi-component adjoining. Multi-Component Tree Adjoining Grammar (MCTAG) differs from (the traditional definition of) TAG in that the elementary objects of the grammar are sets of trees rather than trees, and multi-component adjoining involves the composition of these elementary sets of trees[4] (rather than elementary trees). See Joshi (1987) for more details on Multi-Component Tree Adjoining Grammars (MCTAG).

The multi-component sets, given in Figure 16, may be used to give an account for sentences (1) and (2). Obtained by adjoining the two components of β_{17} in α_{15}, γ_8 can be used for sentence (1) (Figure 17).

The need for introducing multi-component sets and multi-component adjoining (in this case, at least) arises because of the decision in traditional TAGs to compose trees (rather than descriptions of trees, i.e., quasi-trees). In particular, the domination relations allow us to give partial descriptions of trees such as α_{16} (in Figure 18) that captures the same information as in the multi-component set β_{17} (in Figure 16). Note that α_{16} can be described by using the same principles that relate α_{13} and α_{12} (see Figure 14). If, for a moment, we consider α_{15} to be an auxiliary quasi-tree (rather than an initial quasi-tree) and use it for "adjoining" (treating the n_2 quasi-node as the quasi-foot) then we obtain the same structure as γ_8 (Figure 17).

Two issues can be raised with respect to this example. Firstly we can question whether such uses of multi-component adjoining (and where the foot node of one component dominates the root of the other components in the eventual structure[5]) can be considered to be adjoining in the quasi-tree framework; and secondly whether these operations can be thought of as essentially the substitution operation when viewed in this framework (that uses quasi-trees rather than trees). However, α_{15} would normally be called an initial quasi-tree, and we would have considered substitution at the n_2 quasi-node rather than treating α_{15} as an auxiliary quasi-tree and the n_2 quasi-node as the quasi-foot. Nevertheless, this "adjunction" of α_{15} seems to be really playing the role of substitution (with a sub-quasi tree though).

Addressing the first issue, in the case of the multi-component adjoining example used here, we believe the need for multi-component adjoining arises from the fact that objects being composed were defined to be trees. Even in the previous version of FTAG, it was assumed that the objects being composed were trees despite the fact that two feature structures were associated with each node. These top and bottom feature structures associated with a node were supposed to account for a view of that node from two different perspectives (from the top and from below). However,

4 There are three different definitions of multi-component adjoining that have been proposed. The version considered here is the simplest kind: one where a set of trees are simultaneously adjoined into a single tree. This version leads to a system weakly equivalent to TAG. The other definitions include the case where sets of trees are adjoined simultaneously into nodes in trees that belong to another set and finally where a set of auxiliary trees are adjoined simultaneously without any restriction on the adjoining sites.

5 Although not a part of the definition of multi-component adjoining, in all analyses we are aware of, it is the case that the foot node of one component dominates the root of the other component in the eventual structure.

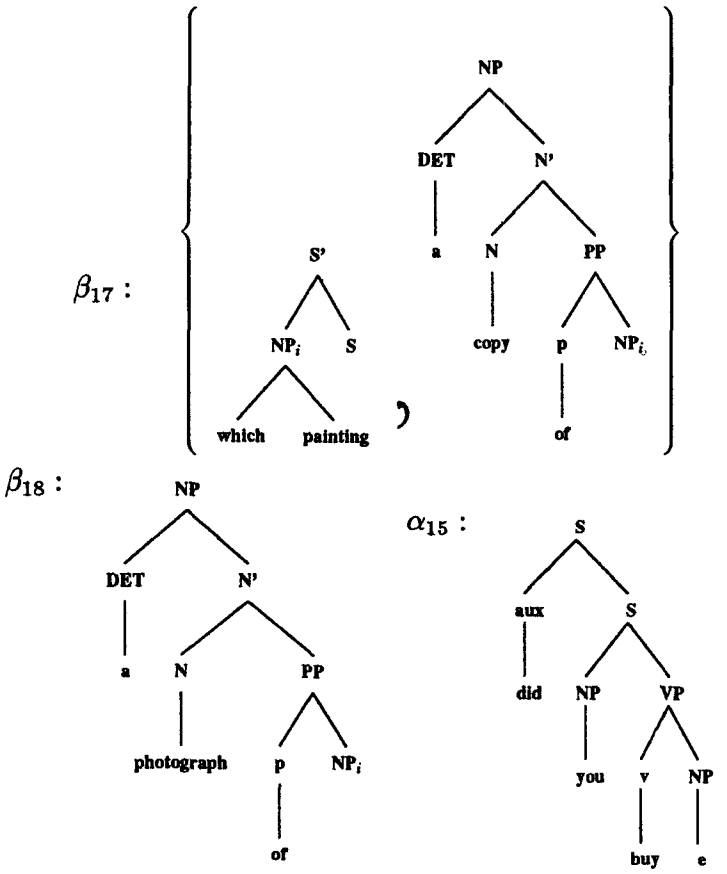


Figure 16
A multi-component tree adjoining grammar.

because one was dealing with a *single node*, it was taken for granted that the two feature structures associated with the *single* node would assign the node the same label $(S, NP, \ldots)$, no matter which perspective (viewing a node from above or from below) one took. That is, we could not consider the possibility of a node whose top and bottom parts were labeled by S and NP. Therefore, instead of using one quasi-tree (α_{16}), a multi-component set, β_{17}, composed of two trees is used. Assuming the possibility of stating domination between quasi-nodes with different labels (as in α_{16}), we can similarly extend the definition of "auxiliary quasi-trees" to allow for the quasi-root and quasi-foot nodes being labeled differently. This is the assumption we made when we "adjoined" the quasi-tree α_{15} to capture the effect of multi-component adjoining.

Assuming that α_{15} is an auxiliary structure points out the similarities between multi-component adjoining and adjoining. However, it is more natural to assume it is an initial quasi-tree and use substitution at the object NP node (rather than call α_{15} an auxiliary quasi-tree and n_2 quasi-node, without any justification, a quasi-foot). A similar situation arises when we consider the so-called *complement* auxiliary trees (see Kroch [1987]). β_{19}, an elementary quasi-tree anchored by a verb such as "think," would be defined to be a complement auxiliary quasi-tree because the quasi-foot is present due to the subcategorization requirements of the anchor. In general, in the lexicalized approach to TAG, it is assumed that such an argument node is expanded

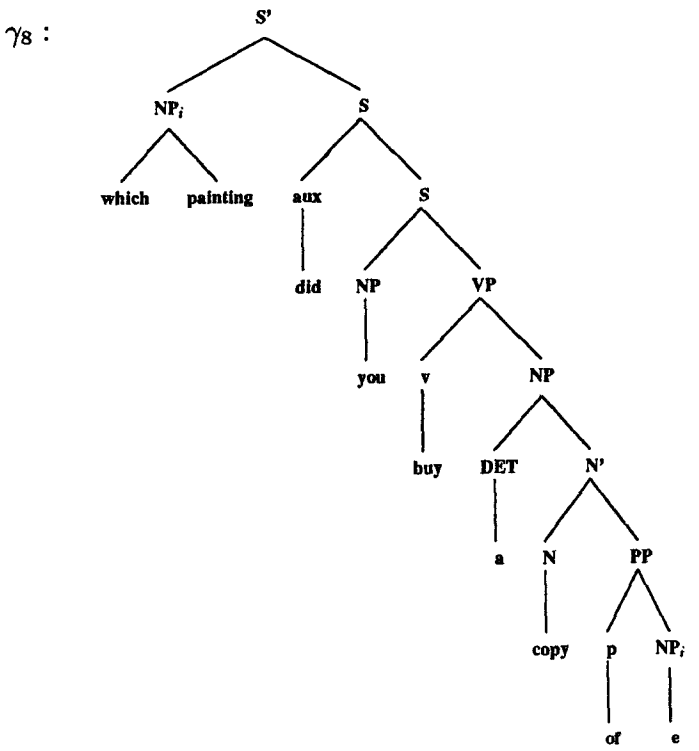


Figure 17
Derivations in MCTAG.

by substitution. This is consistent with Figure 19 where we could call β_{19} an initial quasi-tree and substitute α_{17} at the supposed quasi-foot (s_2) to derive a structure for *Peter thinks John saw Mary*. However, a quasi-tree such as β_{19} must be treated as an auxiliary structure in order that we could use it for adjoining so that it can be adjoined in α_{18} (see Figure 18) at the pair (s_3, s_4) to derive a structure for *who did Peter think John saw*.

The question about the basis of deciding when one should call an elementary structure auxiliary or initial remains. It is hard to justify that s_2 quasi-node of β_{19} is the quasi-foot on the basis of factoring of recursion (the original reason for introducing auxiliary structures). However, while developing a grammar, the s_2 node in β_{19} is not expanded further because we wish to factor recursion, but because it is required by the subcategorization of the anchor and such nodes are expanded as a result of a derivation step. Among the quasi-nodes that appear in the frontier of β_{19}, the s_2 quasi-node is called the quasi-foot because extraction cannot occur from a tree that can appear below the subject *NP* quasi-node, whereas it can in the case of s_2 quasi-node. However, on this basis, one could also call α_{15} an auxiliary quasi-tree and state that the n_2 quasi-node is the quasi-foot.

Structures such as β_{19} and α_{18} (of Figure 20) raise the question of whether there is an essential difference between initial and (complement) auxiliary quasi-trees, and whether adjoining is only a special form of substitution. It appears that in the case of the two examples above, we came to the situation of calling certain structures auxiliary

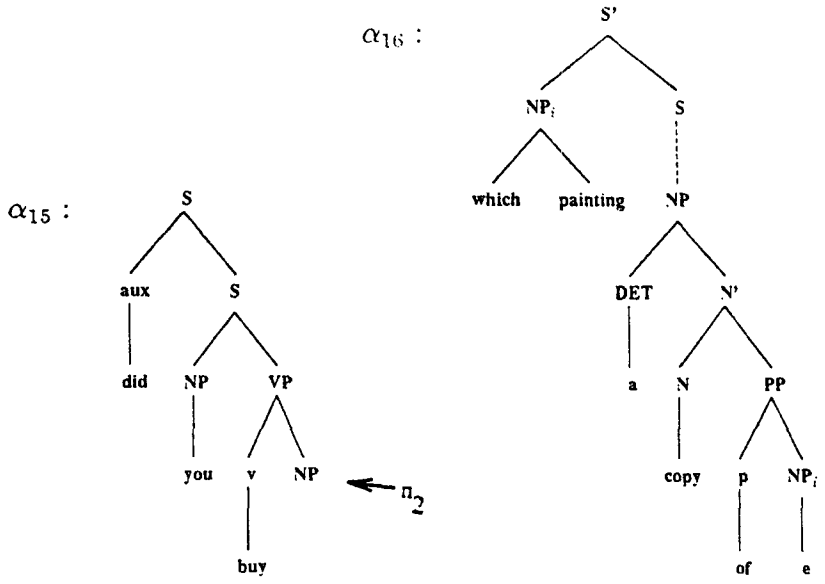


Figure 18
Multi-component adjoining seen as adjoining.

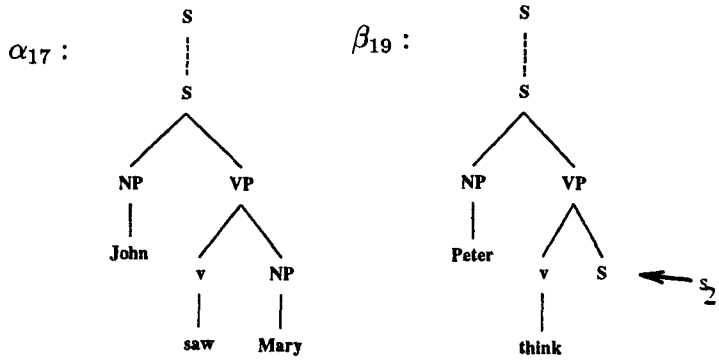


Figure 19
Initial or auxiliary?

structures solely for the purpose of using the adjoining operation. If we wish to claim that there is no essential difference between initial and auxiliary structures (at least of the complement auxiliary tree variety), then we must account for the apparent difference between substitution and adjoining operations. We argue now that it may not be necessary to make this distinction if we take a closer look at the adjoining and substitution operations.

Recall that the substitution operation was defined by the identification of two quasi-nodes. So far this has been illustrated by identifying a quasi-node that appears in the frontier of a quasi-tree with the quasi-root of another quasi-tree. However, now consider β_{19} (see Figure 21) and "substitution" at s_2 by the subtree rooted at s_4 (i.e.,

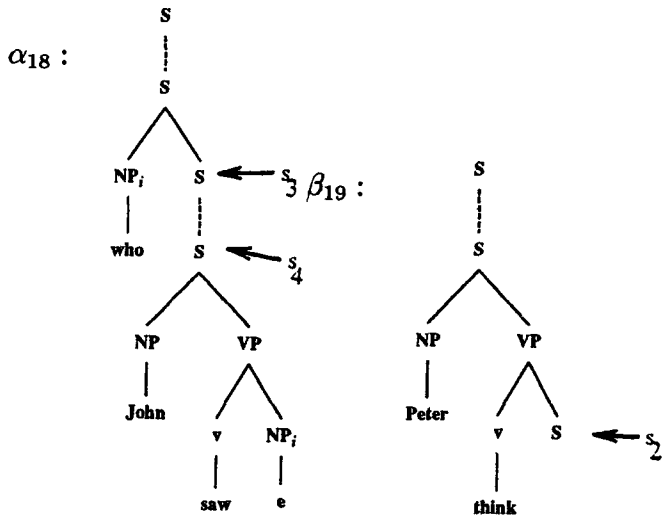


Figure 20
Adjoining versus substitution.

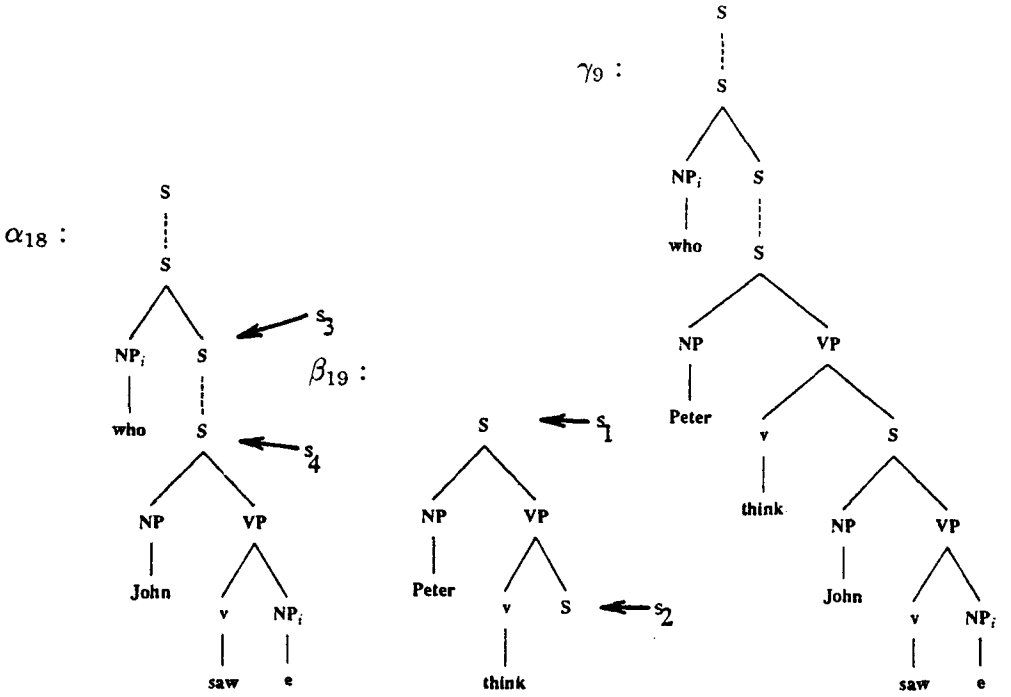


Figure 21
Adjoining seen as substitution.

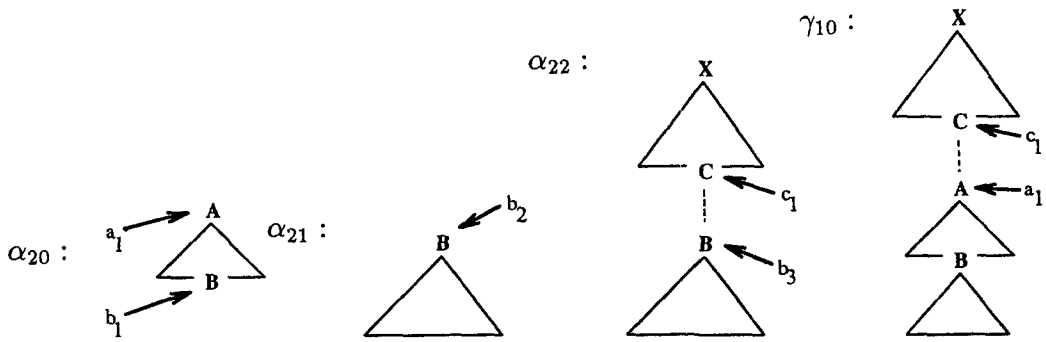


Figure 22
Adjoining.

identify the nodes referred by s_2 and s_4). If we insist that the resulting structure must describe a tree, then we must have either s_1 dominate s_3 or s_3 dominate s_1. Now suppose there are some fundamental linguistic principles (perhaps those principles that govern the makeup of elementary structures and hence also the characteristics of the domination link between paired quasi-nodes) that determine that it is the case that s_3 must dominate s_1 and not vice versa. In this case we obtain γ_9 (as shown in Figure 21), a structure obtained by "adjoining" β_{19}. In fact, that s_3 must dominate s_1 must be derivable from any reasonable linguistic theory that is used to produce the elementary structures concerned (for otherwise a wrong sequence of words would be predicted). One possible explanation of why s_3 dominates s_1 could be given by importing a device like the functional uncertainty machinery (Kaplan and Maxwell 1988) used in LFG. The treatment used in LFG, when imported here, would suggest that zero or more structures of the form given by β_{19} would fit in the gap specified by the domination link between s_3 and s_4. Thus when the identification of s_2 and s_4 takes place, s_3 must dominate s_1 and again zero or more structures of the form of β_{19} could fit between s_3 and s_1 now (see Joshi and Vijay-Shanker [1989]) for a discussion of the treatment of long-distance dependency in TAG and LFG). Another way to explain the domination of s_3 over s_1 could be done by using the notions of maximal government domains discussed by Kroch (1989) and using it now to define the characteristics of the domination links such as that between s_3 and s_4. Note that once the nature of the adjoining operation has been derived, one can pre-compile out the linguistic principles and machinery used to express it. Thus even if one uses, say, the functional uncertainty machinery or maximal government domains, these additional devices (used during the developmental stages) of the grammar need not be used again during the derivation process once we have derived the adjoining operation. This is analogous to the situation with elementary structures. Some linguistic theory will be involved in defining the elementary structures of a TAG. However, once the grammar has been developed, these principles are no longer directly involved during the derivation phase. This is because the principles have been pre-compiled into the elementary structures built.

Figure 22 describes the general situation that may be used to contrast substitution, adjoining and multi-component adjoining. As usual, the identification of the b_1 and b_2 quasi-nodes defines the substitution of the α_{21} at the b_1 quasi-node of α_{20}. Now suppose instead of considering a (quasi) root such as the one named b_2 we consider a pair of

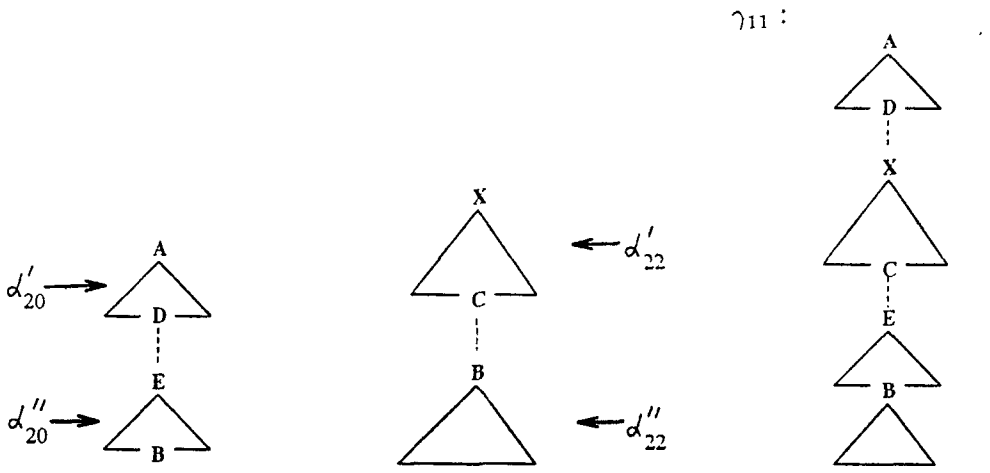


Figure 23
Multi-component adjoining.

quasi-nodes, such as c_1 and b_3, that are interior quasi-nodes. Now suppose we unify the b_1 and b_3 quasi-nodes. Since we will assume that the resulting structure must be a description of a tree, we must have the a_1 quasi-node dominate c_1 quasi-node or vice versa. If the c_1 quasi-node dominates a_1 (as in γ_{10}), we have a structure that appears like the one obtained by adjoining. Suppose there is some principle that predicts this situation to occur when substitution takes place; then we can conclude that adjoining is not a fundamental operation in itself but rather a derived operation. Trying to capture the above-mentioned principle would involve specifying the characteristics of the domination link between pairs of quasi-nodes such as that specified by c_1 and b_3 and the makeup of elementary structures of a grammar.

Let us now consider the other case. Suppose we substitute at the b_1 node with the quasi-tree rooted by b_3; there is no reason to assume that c_1 must dominate a_1. Consider the case when a_1 dominates c_1. In this case, the structure α_{20} must be spliced into two (α'_{20} and α''_{20}) as indicated in Figure 23. There are several possibilities. First, α'_{20} may appear above all of α_{22} as indicated by γ_{11}. This appears to correspond to the version of multi-component adjoining where different components of a set ($\{\alpha'_{20}, \alpha''_{20}\}$) are adjoined simultaneously into another multi-component set, ($\{\alpha'_{22}, \alpha''_{22}\}$). Other possibilities include α'_{20} and α'_{22} splintered into some number of pieces (depending on the domination links found in them) and interleaved in a more complex fashion.

To summarize, when we substitute at b_1 by identifying it with a quasi-root of another structure, we have the standard substitution. On the other hand, when we substitute at b_1 by identifying b_1 and b_3, if c_1 dominates a_1 then the resulting structure appears to be the one formed after adjunction. When a_1 dominates c_1 the situation seems to be comparable with that of multi-component adjoining, where α_{20} and α_{22} are multi-component sets made up of α'_{20}, α''_{20} and α'_{22}, α''_{22}, respectively. Such multi-component adjoining has been used previously in providing linguistic analyses. Since both cases occur (a_1 dominates c_1 or vice versa), we believe it only further justifies our claim that in situations where we consider substitutions as above, whether we have c_1 dominating a_1 (adjoining) or not (multi-component adjoining) depends on the linguistic principles being instantiated during the development of elementary structures (and

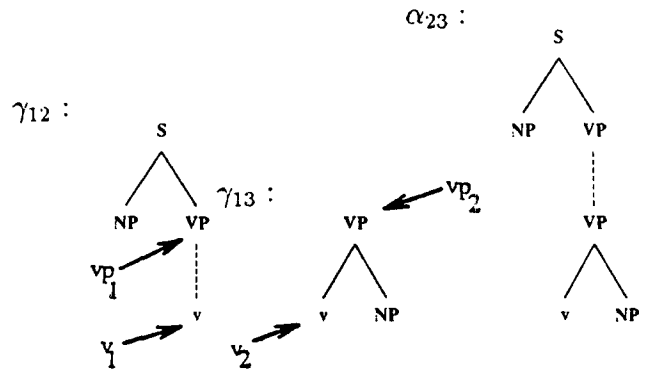


Figure 24
Representation of an elementary structure.

hence also determining the nature of domination links). Thus, this raises the question that although adjoining is used in defining the TAG formalism, could it too (like the elementary structures) be precompiled from some more fundamental principles?

5.2 Describing the Elementary Objects of a Grammar

In this section we show that the new interpretation of the TAG formalism allows the possibility of representing a grammar in a more compact fashion. This is illustrated by means of an example.

The structure named γ_{12} (Figure 24) pictorially represents the normal (or default) tree structure that can be associated with any verb, whereas γ_{13} will be used specifically in the case of a simple transitive verb. The default structure associated with a simple transitive verb can be obtained by considering the description illustrated pictorially by γ_{13} and inheriting the description (γ_{12}) that is common for all verbs. Now since the v_1 and v_2 nodes have to be identified, we have the following.

- The domination link between vp_1 and v_1 quasi-nodes indicates a path length greater than or equal to 0. However, in this case since the labels of these quasi-nodes are different, they cannot refer to the same node. Thus, in this case we have a path length that is greater than 0.
- vp_2 quasi-node immediately dominates the v_2 quasi-node (i.e., path length=1).
- Since v_1 and v_2 quasi-nodes are identified and since we insist on a tree structure, we have vp_1 and vp_2 quasi-nodes in the domination relation. In fact vp_1 quasi-node must dominate vp_2 quasi-node in the resulting structure by a path of length 0 or more (from the two observations above).

Thus we get the structure given by α_{23} as desired. Rogers and Vijay-Shanker (1992) describe a proof system that can be used to perform the type of reasoning involved in constructing the structure α_{23} as described above.

In the manner described above we can build the default structure for every subcategorization frame. Such structures will be specified in any lexicalized TAG; the difference (in the envisaged specification method) is that we no longer precompile out

all possibilities (thus repeating the structure γ_{12} in all structures associated with every type of verb). To complete the description of the rest of the elementary quasi-trees one would have to use transformations, meta-rules, or lexical rules to specify the structures for passivization, wh-movement, topicalization, etc. Work along this direction is being carried out (Vijay-Shanker and Schabes 1992).

6. Conclusions

In this paper, we have embedded TAG in the unification framework in a manner consistent with the constraint-based approach used in this framework. Starting from first principles and taking the localization of dependencies within the elementary structures of a TAG grammar as the only basic principle, we have argued that the objects manipulated by such a grammar are not trees but descriptions of well-formed syntactic structures. From D-Theory, we have adopted the use of domination relation and use of identifiers to refer to nodes while describing such structures. Quasi-trees were introduced to depict pictorially partial descriptions of trees. The pairing of quasi-nodes (with domination link between them) was then used to explain the association of two feature structures with individual nodes in previous definition of Feature structure–based Tree Adjoining Grammars (FTAG). In fact, we also show that the formalism defined in Harbusch (1990) (where only one feature structure is associated with every node) turns out to be similar to the use of FTAG with an additional decision to merge every pair of quasi-nodes by default. We argue that such defaults lead to nonmonotonic behavior.

One can now view FTAG as a generalization of TAG in that arbitrary categories (as used in GPSG) can label nodes, instead of just atomic symbols (nonterminals) as in TAG. In fact, by not insisting that a pair of quasi-nodes be labeled by the same category in FTAG, as was done in TAG, we argue that the "adjoining constraints" follow from the definition of adjunction and the labeling of quasi-nodes, thus making unnecessary the stipulations of SA and OA constraints. In addition, contrary to the assumptions made in current literature on TAG, we show that there are two possible interpretations of NA constraints, only one of which is a special case of SA constraint. We note that as the information associated with quasi-nodes grows during derivation, "adjoining constraints" get instantiated dynamically in an FTAG. We make use of this property in order to give examples to show how FTAG can give more succinct descriptions than TAG.

We have given a logical formulation of FTAG. This builds on a similar treatment of FUG given by Rounds and Manaster-Ramer. We view this logical formulation as a description of those trees and associated feature structures that are built by CFG-based unification grammars. Unlike a CFG-based formalism that allows only for substitution operation, for an FTAG one has to depict adjunction in addition to substitution. Our treatment captures both these cases. We end by giving a presentation of the semantics that can be used to give the denotation of a grammar, i.e., in our case, the structures derived by a grammar.

We have emphasized throughout the paper that we are only interested in the definition of the FTAG formalism. In particular, we have not been concerned with linguistic analyses. However, we have raised a few questions about the formalism that we believe can only be answered on linguistic grounds. In the context of the new interpretation, some of these include whether the linguistic uses of multi-component adjoining can be simulated as the adjoining operation; whether there is an essential need to divide the elementary structures of the grammar as initial and auxiliary structures; and whether the adjoining operation itself can be defined as a substitution

operation, the apparent differences between these operations being derived on the basis of some more fundamental linguistic principles used in the design of the elementary structures of the grammar. Even if the answer is in the affirmative, we believe there is considerable advantage to be gained by deriving this operation in order that we can manipulate directly the elementary structures that localize various forms of the dependencies. As observed earlier, with the derivation of this operation (like the derivation of the elementary structures of the grammar), we can disregard (i.e., not reason with) the principles used (to derive them) during the derivation of more complex structures. Finally we have also shown that the new interpretation of the TAG formalism proposed here allows for the possibility of a more compact representation of a TAG grammar.

Acknowledgments

This work was partially supported by NSF grant IRI-9016591.

I am extremely grateful to A. Abeille, A. K. Joshi, A. Kroch, K. F. McCoy, Y. Schabes, S. M. Shieber, and D. J. Weir. Their suggestions and comments at various stages have played a substantial role in the development of this work. I am thankful to the reviewers for many useful suggestions. Many of the figures in this paper have been drawn by XTAG (Schabes and Paroubek 1992), a workbench for Tree-Adjoining Grammars. I would like to thank Yves Schabes for making this available to me.

References

Abeille, A.; Bishop, K.; Cote, Sharon; and Schabes, Y. (1990). "A lexicalized tree adjoining grammar for English." Technical report MS-CIS-90-24, Department of Computer and Information Science, University of Pennsylvania.

Abeille, A. (1991). *Une grammaire lexicalisée d'Arbres adjoints pour le Français: Application à l'analyse automatique.* Doctoral dissertation, Université Paris 7, Paris, France.

Gazdar, G.; Klein, E.; Pullum, S.; and Sag, I. (1985). *Generalized Phrase Structure Grammar.* Blackwell.

Harbusch, K. (1990). "Constraining tree adjoining grammars by unification." In *Proceedings, 13th International Conference on Computational Linguistics.*

Henderson, J. (1990). "Structure Unification Grammar: A Unifying Framework for Investigating Natural Language." Technical Report MS-CIS-90-94, Department of Computer and Information Science, University of Pennsylvania.

Johnson, M. (1988). *Attribute Value Logic and the Theory of Grammar.* University of Chicago Press.

Joshi, A. K. (1987). "An introduction to tree adjoining grammar." In *Mathematics of Language,* edited by A. Manaster-Ramer, 87–115. Benjamins Publishing Company.

Joshi, A. K. (1985). "How much context-sensitivity is necessary for characterizing structural descriptions—Tree Adjoining Grammars." In *Natural Language Processing—Theoretical, Computational and Psychological Perspective,* edited by D. Dowty, L. Karttunen, and A. Zwicky, 206–250. Cambridge University Press.

Joshi, A. K., and Vijay-Shanker, K. (1989). "Treatment of long-distance dependencies in LFG and TAG." In *Proceedings, 27th Annual Meeting of the Association for Computational Linguistics,* 220–227.

Joshi, A. K.; Levy, L. S.; and Takahashi, M. (1975). "Tree adjunct grammars." *Journal of Computer and System Sciences,* 10(1), 136–163.

Kaplan, R. M., and Maxwell, J. T. (1988). "An algorithm for functional uncertainty." In *Proceedings, 12th International Conference on Computational Linguistics,* 297–302.

Kasper, R., and Rounds, W. C. (1986). "A logical semantics for feature structures." In *Proceedings, 24th Annual Meeting of the Association for Computational Linguistics,* 257–266.

Kroch, A. S. (1989). "Asymmetries in long distance extraction in a tag grammar." In *Alternative Conceptions of Phrase Structure,* edited by M. Baltin and A. Kroch, 66–98. University of Chicago Press.

Kroch, A. S. (1987). "Unbounded dependencies and subjacency in a tree adjoining grammar." In *Mathematics of Language,* edited by A. Manaster-Ramer, 143–172. Benjamins Publishing Company.

Kroch, A., and Joshi, A. K. (1985). "Linguistic relevance of tree adjoining grammars." Technical report MS-CI-85-18, Department of Computer and Information Science, University of Pennsylvania.

Marcus, M.; Hindle, D.; and Fleck, M. M.

(1983). "D-theory—talking about talking about trees." In *Proceedings, 21st Annual Meeting of the Association for Computational Linguistics*, 129–136.

Moshier, M. A. (1988). *Extensions to unification grammar for the description of programming languages*. Doctoral dissertation, University of Michigan, Ann Arbor, MI.

Rogers, J., and Vijay-Shanker, K. (1992). "Reasoning with descriptions of trees." In *Proceedings, 30th Annual Meeting of the Association for Computational Linguistics*, 72–80.

Rounds, W., and Manaster-Ramer, A. (1987). "A logical version of functional grammar." In *Proceedings, 25th Annual Meeting of the Association for Computational Linguistics*, 89–96.

Schabes, Y., and Paroubek, P. (1992). "Xtag—a graphical workbench for developing tree adjoining grammars." In *Proceedings, Third Conference on Applied Natural Language Processing*, 216–223.

Schabes, Y.; Abeille, A.; and Joshi, A. K. (1988). "New parsing strategies for tree adjoining grammars." In *Proceedings, 12th International Conference on Computational Linguistics*, 578–583.

Vijay-Shanker, K. (1987). *A study of tree adjoining grammars*. Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

Vijay-Shanker, K., and Joshi, A. K. (1988). "Feature structure based tree adjoining grammars." In *Proceedings, 12th International Conference on Computational Linguistics*, 714–719.

Vijay-Shanker, K., and Schabes, Y. (1992). "Structure sharing in a lexicalized tree adjoining grammar." In *Proceedings, 14th International Conference on Computational Linguistics*, 205–211.