October 1988

Feature Structures Based Tree Adjoining Grammars

Aravind K. Joshi
University of Pennsylvania, joshi@cis.upenn.edu

K. Vijay Shanker
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-79.

Feature Structures Based Tree Adjoining Grammars

Abstract

We have embedded Tree Adjoining Grammars (TAG) in a feature structure based unification system. The resulting system, Feature Structure based Tree Adjoining Grammars (FTAG), captures the principle of factoring dependencies and recursion, fundamental to TAG's. We show that FTAG has an enhanced descriptive capacity compared to TAG formalism. We consider some restricted versions of this system and some possible linguistic stipulations that can be made. We briefly describe a calculus to represent the structures used by this system, extending on the work of Rounds, and Kasper [Rounds et al. 1986, Kasper et al. 1986)involving the logical formulation of feature structures.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-79.

FEATURE STRUCTURES BASED TREE ADJOINING GRAMMARS

K. Vijay-Shanker
and A. K. Joshi

MS-CIS-88-79
LINC LAB 136

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

October 1988

Proceedings of the International Conference on Computational Linguistics (COLING 88), Budapest, Hungary, August 1988

Acknowledgements: This research was supported in part by DARPA grant NOOO14-85-K-0018, NSF grants MCS-8219196-CER, IRI84-10413-AO2 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Feature Structures Based Tree Adjoining Grammars[1]

K. Vijay-Shanker
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19711
U.S.A

A. K. Joshi
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
U.S.A

Abstract We have embedded Tree Adjoining Grammars (TAG) in a feature structure based unification system. The resulting system, Feature Structure based Tree Adjoining Grammars (FTAG), captures the principle of factoring dependencies and recursion, fundamental to TAG's. We show that FTAG has an enhanced descriptive capacity compared to TAG formalism. We consider some restricted versions of this system and some possible linguistic stipulations that can be made. We briefly describe a calculus to represent the structures used by this system, extending on the work of Rounds, and Kasper [Rounds et al. 1986, Kasper et al. 1986] involving the logical formulation of feature structures.

1 Introduction

Tree Adjoining Grammars (TAG) were first introduced by Joshi, Levy, and Takahashi [Joshi et al. 1975]. The first study of this system, from the point of view of its formal properties and linguistic applicability, was carried out by Joshi in [Joshi 1985]. TAG's have been used in providing linguistic analyses; a detailed study of the linguistic relevance was done by Kroch and Joshi in [Kroch et al. 1985].

In this paper, we show how TAG's can be embedded in a feature structure based framework. Feature structure based Tree Adjoining Grammars (FTAG) are introduced in Section 2, and is followed by a comparsion of the descriptive capacity of FTAG and TAG. A restricted version of FTAG is proposed and some possible linguistic stipulations are considered. In Section 3, we introduce a calculus, which is an extension of the logical calculus of Rounds and Kasper [Rounds et al. 1986, Kasper et al. 1986] allowing λ-abstraction and application, in order to describe the structures used in FTAG's. Finally, in Section 4, we summarise the work presented in this paper.

1.1 Introduction to Tree Adjoining Grammars

Tree Adjoining Grammars (TAG), unlike other grammatical systems used in computational linguistics, is a tree rewriting system. Unlike the string rewriting formalisms which writes 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 agreement, subcategorization, and filler-gap. There are two kinds of elementary trees: the *initial trees* and *auxiliary trees*. The initial trees roughly (Figure 1) correspond to simple sentences. Thus, the root of an initial tree is labelled by the symbol S. They are required to have a frontier made up of terminals.

The auxiliary trees (Figure 2) correspond roughly to minimal recursive constructions. Thus, if the root of an auxiliary tree is labelled by a nonterminal symbol, X, then there is a node (called the foot node) in the frontier of this tree which is labelled by X. The rest of the nodes in the frontier are labelled by terminal symbols.

[1]This work was partially supported by NSF grants MCS-82-19116-CER, DCR-84-10413,ARO grant DAA29-84-9-0027, and DARPA grant N0014-85-K0018

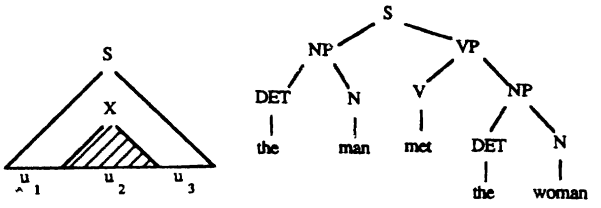

Figure 1: Initial Trees

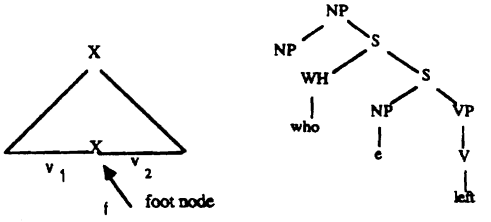

Figure 2: Auxiliary Trees

We will now define the operation of adjunction. Let γ be a tree with a node labelled by X. Let β be an auxiliary tree, whose root and foot node are also labelled by X. Then, adjoining β at the node labelled by X in γ will result in the tree illustrated in Figure 3. In Figure 3, we also

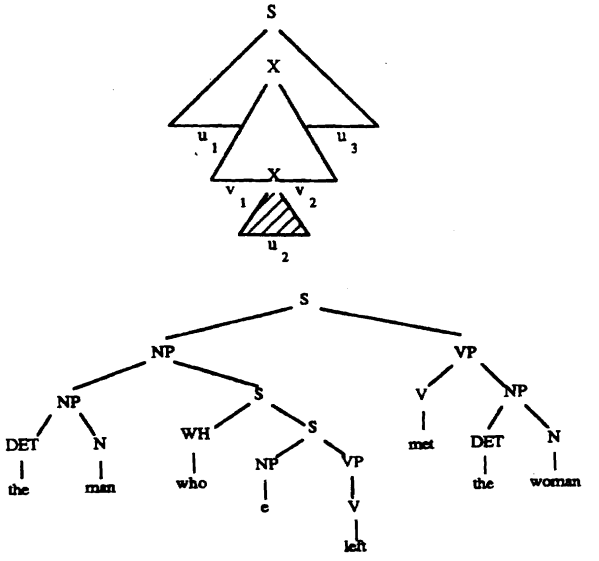

Figure 3: The operation of adjoining

show the result of adjoining the auxiliary tree β_1 at the subject NP node of the initial tree α_1.

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 the node must be the

same as the label of the root (and the foot) node of the auxiliary tree. Further restriction on this set of auxiliary trees is done by enumerating with each node the subset of auxiliary trees which can be adjoined at that node. This specification of a set of auxiliary trees, which can be adjoined at a node, is called the *Selective Adjoining* (SA) constraints. In the case where we specify the empty set, we say that the node has a *Null Adjoining* (NA) constraints. It is possible 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's and their linguistic relevance may be found in [Kroch et al. 1985].

1.2 Feature Structure Based Grammatical Systems

Several different approaches to natural language grammars have developed the notion of *feature structures* to describe linguistic objects. In order to capture certain linguistic phenomena such as agreement, subcategorization, etc., a number of recent grammatical systems have added, on top of a CFG skeleton, a feature based informational element. Example of such systems (see [Shieber 1985a]) include Generalized Phrase Structure Grammars (GPSG), Lexical functional Grammars (LFG), and Head-driven Phrase Structure Grammars (HPSG). A feature structure (as given below) is essentially a set of attribute-value pairs where values may be atomic symbols or another feature structure.

$$\begin{bmatrix} cat: & S \\ 1: & \boxed{\square}\begin{bmatrix} cat: & NP \\ agr: & \boxed{\square} \end{bmatrix} \\ 2: & \begin{bmatrix} cat: & VP \\ agr: & \boxed{\square} \\ subject: & \boxed{\square} \end{bmatrix} \end{bmatrix}$$

The notation of the co-indexing box ($\boxed{\square}$ in this example) is used to express the fact that the values of two subfeatures are the same. Feature structures with co-indexing boxes have also been called *reentrant* feature structures in the literature.

We can define a partial ordering, $\sqsubseteq$, on a set of feature structures using the notion of *subsumption* (carries *less information* or is *more general*). Unification of two feature structures (if it is defined) corresponds to the feature structure that has all the information contained in the original two feature structures and nothing more. We will not describe feature structures any further (see [Shieber 1985a] for more details on feature structures and an introduction to the unification based approach to grammars).

2 Feature Structure Based Tree Adjoining Grammars (FTAG)

The linguistic theory underlying TAG's is centered around the factorization of recursion and localization of dependencies into the elementary trees. The "dependent" items usually belong to the same elementary tree[2]. Thus, for example, the predicate and its arguments will be in the same tree, as will the filler and the gap. Our main goal in embedding TAG's in an unificational framework is to capture this localization of dependencies. Therefore, we would like to associate feature structures with the elementary trees (rather than break these trees into a CFG-like rule based systems, and then use some mechanism to ensure only the trees produced by the TAG itself are generated[3]). In the feature structures associated with the elementary trees, we can state the constraints among the dependent nodes directly. Hence, in an initial tree corresponding to a simple sentence, we can state that the main verb and the subject NP (which are part of the same initial tree) share the agreement feature. Thus, such checking, in many cases, can be precompiled (of course only after lexical insertion) and need not be done dynamically.

2.1 General Schema

In unification grammars, a feature structure is associated with a node in a derivation tree in order to describe that node and its realtion to features of other nodes in the derivation tree. In a TAG, any node in an elementary tree is related to the other nodes in that tree in two ways. Feature structures written in FTAG using the standard matrix notation, describing a node, η, can be made on the basis of:

1. the relation of η to its supertree, i.e., the view of the node from the top. Let us call this feature structure as t_η.
2. the relation to its descendants, i.e., the view from below. This feature structure is called b_η.

Note that both the t_η and b_η feature structure hold of the node η. In a derivation tree of a CFG based unification system, we associate one feature structure with a node (the unification of these two structures) since both the statements, t and b, together hold for the node, and no further nodes are introduced between the node's supertree and subtree. This property is not true in a TAG. On adjunction, at a node there is no longer a single node; rather an auxiliary tree replaces the node. We believe that this approach of associating two statements with a node in the auxiliary tree is in the spirit of TAG's because of the OA constraints in TAG's. A node with OA constraints *cannot* be viewed as a single node and must be considered as something that has to be replaced by an auxiliary tree. t and b are restrictions about the auxiliary tree that must be adjoined at this node. Note that if the node does not have OA constraint then we should expect t and b to be compatible. For example, in the final sentential tree, this node will be viewed as a single entity.

Thus, in general, with every internal node, η, (i.e., where adjunction could take place), we associate two structures, t_η and b_η. With each terminal node, we would associate only one structure[4].

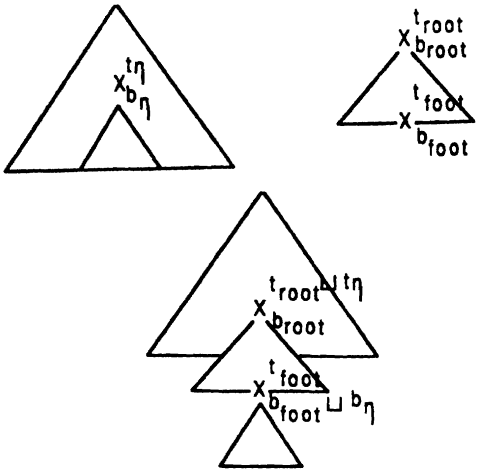

Figure 4: Feature structures and adjunction

[2] It is sometimes possible for "dependent" items to belong to an elementary tree and the immediate auxiliary tree that is adjoined in it.

[3] Such a scheme would be an alternate way of embedding TAG's in an unificational framework. However, it does not capture the linguistic intuitions underlying TAG's, and loses the attractive feature of localizing dependencies.

[4] It is possible to allow adjunctions at nodes corresponding to pre-lexical items. For example, we may wish to obtain verb-clusters by adjunctions at nodes which are labelled as verbs. In such a case, we will have to associate two feature structures with pre-lexical nodes too.

Let us now consider the case when adjoining takes place as shown in the figure 4. The notation we use is to write alongside each node, the t and b statements, with the t statement written above the b statement. Let us say that t_{root},b_{root} and t_{foot},b_{foot} are the t and b statements of the root and foot nodes of the auxiliary tree used for adjunction at the node η. Based on what t and b stand for, it is obvious that on adjunction the statements t_η and t_{root} hold of the node corresponding to the root of the auxiliary tree. Similarly, the statements b_η and b_{foot} hold of the node corresponding to the foot of the auxiliary tree. Thus, on adjunction, we unify t_η with t_{root}, and b_η with b_{foot}. In fact, this adjunction is permissible only if t_{root} and t_η are compatible as are b_{foot} and b_η. If we do not adjoin at the node, η, then we unify t_η with b_η. At the end of a derivation, the tree generated must not have any nodes with OA constraints. We check that by unifying the t and b feature structures of every node. More details of the definition of FTAG may be found in [Vijayashanker 1987].

We now give an example of an initial tree and an auxiliary tree. We would like to note that, just as in a TAG, the elementary trees which are the domain of co-occurence restrictions is available as a single unit during each step of the derivation. Thus, most of these co-occurence constraints can be checked even before the tree is used in a derivation, and this checking need not be linked to the derivation process.

2.2 Unification and Constraints

Since we expect that there are linguistic reasons determining why some auxiliary tree can be adjoined at a tree and why some cannot, or why some nodes have OA constraint, we would like to express these constraints in the feature structures associated with nodes. Further, as described in Section 2.1, adjunctions will be allowed only if the appropriate feature structures can be unified. Thus, we expect to implement the adjoining constraints of TAG's simply by making declarative statements made in the feature structures associated with the nodes to ensure that only the appropriate trees get adjoined at a node.

The adjoining constraints are implemented in FTAG as follows. Notice, from Figure 4, t_η and t_{root}, and b_η and b_{foot} must be compatible for adjunction to occur. We hope to specify some feature-values in these t, b statements to specify the local constraints so that

1. if some auxiliary tree should not adjoined at a node (because of its SA constraint) then some unification involved (t_η with t_{root}, or b_{foot} with b_η) in our attempt to adjoin this auxiliary tree will fail, and

2. if a node has OA constraint, we should ensure that an appropriate auxiliary tree *does* get adjoined at that node. This is ensured if t_η is incompatible with b_η.

The example, given in Figure 7, illustrates the implementation of both the OA and SA constraint. The view of the root node of α from below suggests that b statement for this node makes the assertion that the value of the *tense* attribute is $-$ (or untensed). However, the t statement should assert *tense* : + (since every complete sentence must be tensed)[5]. Thus, an auxiliary tree whose root node will correspond to a tensed sentence and whose foot node will dominate an untensed sentence can be adjoined at this node. Therefore, only those auxiliary trees whose main verb subcategorizes for an untensed sentence (or an infinitival clause) can be adjoined at the root node of this initial tree. This shows why only auxiliary tree such as β can be adjoined, whereas an auxiliary tree corresponding to *John thinks S* can not be adjoined since the verb *thinks* subcategories for a tensed sentence. The example also serves to illustrate the implementation of OA constraint at the root of α, since the t and b feature structures for this node are not unifiable.

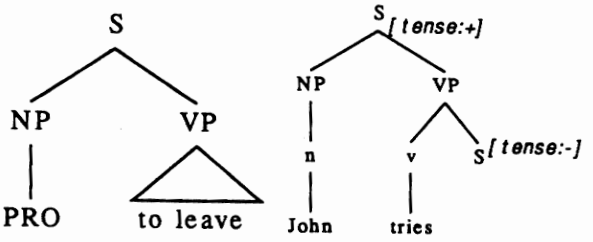

Figure 6: Illustration of implementation of SA and OA constraints

2.2.1 Comments on the Implementation of Constraints in FTAG

In the TAG formalism, local constraints are specified by enumeration. However, specification by enumeration is not a linguistically attractive solution. In FTAG we associate with each node two feature structures which are declarations of linguistic facts about the node. The fact that only appropriate trees get adjoined is a corollary of the fact that only trees consistent with these declarations are acceptable trees in FTAG. As a result, in a FTAG, constraints are dynamically instantiated and are not pre-specified as in a TAG. This can be advantageous and useful for economy of grammar specification. For example, consider the derivation of the sentence

What do you think Mary thought John saw

In the TAG formalism, we are forced to replicate some auxiliary trees. Consider the auxiliary tree β_1 in the TAG fragment in Figure 7. Since the intermediate phrase *what Mary thought John saw* is not a complete sentence, we will have to use OA constraints at the root of the auxiliary tree β_1. However, this root node should not have OA constraints when it is used in some other context; as in the case of the derivation of

Mary thought John saw Peter

We will need another auxiliary tree, β_2, with exactly the same tree structure as β_1 except that the root of β_2 will not have an OA constraint. Further, the root nodes in α_1 and α_2 have SA constraints that allow for adjunction only by β_1 and β_2 respectively. As seen in the Figure 8, corresponding to the FTAG fragment, we can make use of the fact that constraints are dynamically instantiated and give only one specification of β_1. When used in the derivation of

What do you think Mary thought John saw

t_{root} inherits the feature *inverted* : + which it otherwise does not have, and b_{root} inherits the feature *inverted* : $-$. Thus, the node which corresponds to root of β_1, by the dynamic instantiation of the feature structure, gets an OA constraint. Note that there will not be any OA constraint in nodes of the final tree corresponding to

What do you think Mary thought John saw.

Also, the root of the auxiliary tree, corresponding to *Mary thought S*, does not get OA constraint, when this tree is used in the derivation of the sentence

Mary thought John saw Peter.

[5] t statement is more complicated than just "view from the top". t statement is a statement about the node while viewing the node from the top, and hence is a statement concerning the entire subtree below this node (i.e., including the part due to an auxiliary tree adjoined at the node), and how it constrains the derivation of the nodes which are its siblings and ancestors. b remains the same as before, and is the statement about this node and the subtree below it, without considering the adjunction at this node.

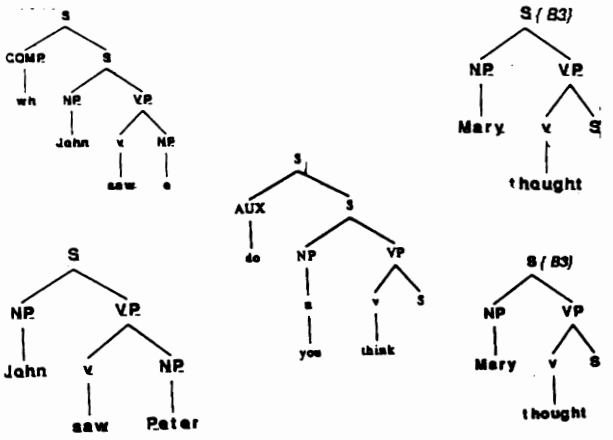

Figure 7: A TAG fragment

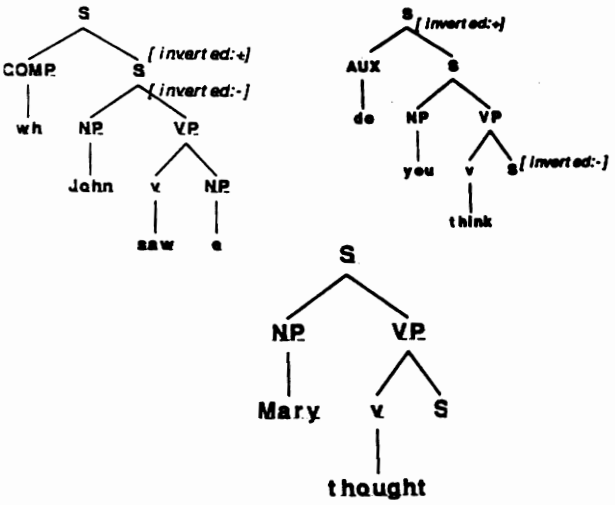

Figure 8: An FTAG fragment

2.3 Some Possible Linguistic Stipulations in FTAG

In this section, we will discuss some possible stipulations for a FTAG grammar. However, at this stage, we do not want to consider these stipulations as a part of the formalism of FTAG. First, some of the linguistic issues pertaining to these stipulations have not yet been settled. Secondly, our primary concern is to specify the FTAG formalism. Further, if the formalism has to incorporate these stipulations, it can be done so, without altering the mechanism significantly.

The current linguistic theory underlying TAG's assumes that every foot node has a NA constraint. The justification of this stipulation is similar to the projection principle in Chomsky's transformation theory. It is appealing to state that the adjunction operation does not alter the grammatical relations defined by the intermediate tree structures. For example, consider the following derivation of the sentence

Mary thought John saw Bill hit Jill.

If the derivation results in the intermediate tree corresponding to *Mary thought Bill hit Jill*, then we would expect to obtain the relation of Mary thinking that "Bill hit Jill". This relation is altered by the adjunction at the node corresponding to the foot node of the auxiliary tree corresponding to *Mary thought S*.

If we wish to implement this stipulation, one solution is to insist that only one F-V statement is made with the foot node, i.e., the t_{foot} and b_{foot} are combined. The definition of adjunction can be suitably altered.

The second stipulation involves the complexity of the feature structure associated with the nodes. So far, we have not placed any restrictions on the growth of these feature structures. One of the possible stipulations that are being considered from the point of view of linguistic relevance is to put a bound on the information content in these feature structures. This results in a bound on the size of feature structures and hence on the number of possible feature structures that can be associated with a node. An FTAG grammar, which incorporates this stipulation, will be equivalent to a TAG from the point of view of generative capacity but one with an enhanced descriptive capacity.

Unbounded feature structures have been used to capture the subcategorization phenomenon by having feature structures that act like stacks (and hence unbounded in size). However, in TAG's, the elementary trees give the subcategorization domain. As noted earlier, the elements subcategorized by the main verb in an elementary tree are part of the same elementary tree. Thus, with the feature structures associated with the elementary trees we can just point to the subcategorized elements and do not need any further devices. Note, that any stack based mechanism that might be needed for subcategorization is provided by the TAG formalism itself, in which the tree sets generated by TAG's have context free paths (unlike CFG's which have regular paths). This additional power provided by the TAG formalism has been used to an advantage in giving an account of West Germanic verb-raising [Santorini 1986].

3 A Calculus to Represent FTAG Grammars

We will now consider a calculus to represent FTAG's by extending on the logical formulation of feature structures given by Rounds and Kasper [Rou Kasper et al. 1986]. Feature structures in this logic (henceforth called R-K logic) are represented as formulae. The set of well-formed formulae in this logic is recursively defined as follows.

$$
\begin{aligned}
e ::= \; & NIL \\
& TOP \\
& a \\
& l : e_1 \\
& e_1 \wedge e_2 \\
& e_1 \vee e_2 \\
& \{p_1, \ldots, p_n\}
\end{aligned}
$$

where a is an atomic value, e_1, e_2 are well-formed formulae. NIL and TOP convey "no information" and "inconsistent information" respectively. Each p_i represents a path of the form $l_{i,1} : l_{i,2} : \ldots : l_{i,n_i}$ respectively. This formula is interpreted as $p_1 = \ldots = p_n$, and is used to express reentrancy.

Our representation of feature structures similar to the R-K logic's representation of feature structures and differs only in the clause for reentrancy. Given that we want to represent the grammar itself in our calculus, we can not represent reentrancy by a finite set of paths. For example, suppose we wish to state that agreement features of a verb matches with that of its subject (note in a TAG the verb and its subject are in the same elementary tree), the two paths to be identified can not be stated until we obtain the final derived tree. To avoid this problem, we use a set of equations to specify the reentrancy. The set of equations have the form given by $x_i = e_i$ for $1 \leq i \leq n$, where $x_1, \ldots, x_n$ are variables, $e_1, \ldots, e_n$ are formulae which could involve these variables.

For example, the reentrant feature structure used in Section 1.2, is represented by the set of equations

$$x = cat : S \wedge 1 : y \wedge 2 : (cat : VP \wedge agr : z \wedge subject : y)$$
$$y = cat : NP \wedge agr : z$$

We represent a set of equations, $x_i = e_i$ for $1 \leq i \leq n$ as

$$rec < x_1, \ldots, x_n > < e_1, \ldots, e_n >$$

Let us now consider the representation of trees in FTAG and the feature structures that are associated with the nodes. The elementary feature structure associated with each elementary tree encodes certain relationships between the nodes. Included among these relationships are the sibling and ancestor/descendent relationships; in short, the actual structure of the tree. Thus, associated with each node is a feature structure which encodes the subtree below it. We use the attributes $i \in \mathcal{N}$ to denote the i^{th} child of a node.

To understand the representation of the adjunction process, consider the trees given in Figure 4, and in particular, the node η. The feature structure associated with the node where adjunction takes place should reflect the feature structure after adjunction and as well as without adjunction (if the constraint is not obligatory). Further, the feature structure (corresponding to the tree structure below it) to be associated with the foot node is not known but gets specified upon adjunction. Thus, the bottom feature structure associated with the foot node, which is b_{foot} before adjunction, is instantiated on adjunction by unifying it with a feature structure for the tree that will finally appear below this node. Prior to adjunction, since this feature structure is not known, we will treat it as a variable (that gets instantiated on adjunction). This treatment can be obtained if we think of the auxiliary tree as corresponding to functions over feature structures (by λ-abstracting the variable corresponding to the feature structure for the tree that will appear below the foot node). Adjunction correponds to applying this function to the feature structure corresponding to the subtree below the node where takes place.

We will formalize representation of FTAG as follows. If we do not consider adjoining at the node η, the formula for γ will be of the form

$$(\ldots t_\eta \wedge b_\eta \wedge \ldots)$$

Suppose the formula for the auxiliary tree β is of the form

$$(t_{root} \wedge \ldots b_{foot})$$

the tree obtained after adjunction at the node η will then be represented by the formula

$$(\ldots t_\eta \wedge (t_{root} \wedge \ldots b_{foot}) \wedge b_\eta \wedge \ldots)$$

We would like to specify one formula with the tree γ, and use appropriate operation corresponding to adjunction by β or the case where we do not adjoin at η. Imagining adjunction as function application where we consider auxiliary trees as functions, the representation of β is a function, say f_β, of the form

$$\lambda f.(t_{root} \wedge \ldots (b_{foot} \wedge f))$$

To allow the adjunction of β at the node η, we have to represent γ by

$$(\ldots t_\eta \wedge f_\beta(b_\eta) \wedge \ldots)$$

Then, corresponding to adjunction, we use function application to obtain the required formula. But note that if we do not adjoin at η, we would like to represent γ by the formula

$$(\ldots t_\eta \wedge b_\eta \wedge \ldots)$$

which can be obtained by representing γ by

$$(\ldots t_\eta \wedge I(b_\eta) \wedge \ldots)$$

where I is the identity function. Similarly, we may have to attempt adjunction at η by any auxiliary tree (SA constraints are handled by success or failure of unification). Thus, if $\beta_1, \ldots, \beta_n$ form the set of auxiliary tree, we have a function, F, given by

$$F = \lambda f.(f_{\beta_1}(f) \vee \ldots \vee f_{\beta_n}(f) \vee I(f)) = \lambda f.(f_{\beta_1}(f) \vee \ldots \vee f_{\beta_n}(f) \vee f)$$

and represent γ by

$$(\ldots t_\eta \wedge F(b_\eta) \wedge \ldots)$$

In this way, we can represent the elementary trees (and hence the grammar) in an extended version of R-K logic (to which we add λ-abstraction and application).

3.1 Representing Tree Adjoining Grammars

We will now turn our attention to the actual representation of an FTAG grammar, having considered how the individual elementary trees are represented. According to our discussion in the previous section, the auxiliary trees are represented as functions of the form $\lambda x.e$ where e is a term in FSTR which involves the variable x. If $\beta_1, \ldots, \beta_n$ are the auxiliary trees of a FTAG, G, then we have equations of the form

$$f_1 = \lambda x.e_1$$
$$\vdots$$
$$f_n = \lambda x.e_n$$

$e_1, \ldots, e_n$ are encodings of auxiliary trees $\beta_1, \ldots, \beta_n$ as discussed above. These expressions obey the syntax which is defined recursively as follows.

$$\begin{aligned} e &::= NIL \\ &::= TOP \\ &::= x \\ &::= l : e_1 \\ &::= e_1 \wedge e_2 \\ &::= e_1 \vee e_2 \\ &::= f(e) \end{aligned}$$

where x is a variable over feature structures and f is a function variable.

In addition, as discussed above, we have another equation given by

$$f_0 = \lambda x.f_1(x) \vee \ldots \vee f_n(x)$$

The initial trees are represented by a set of equations of the form

$$x_1 = e'_1$$
$$\vdots$$
$$x_m = e'_m$$

where $e'_1, \ldots, e'_m$ are expressions which describe the initial trees $\alpha_1, \ldots, \alpha_m$ Note that in the expressions $e_1, \ldots, e_n, e'_1, \ldots, e'_m$, wherever adjunction is possible, we use the function variable f_0 as described above. The grammar is characterized by the structures derivable from any one of the initial trees. Therefore, we add

$$x_0 = x_1 \vee \ldots \vee x_m$$

Assuming that we specify reentrancy using the variables $y_1, \ldots, y_k$ and equations $y_i = e''_i$ for $1 \leq i \leq k$, an FTAG grammar is thus represented by the set of equations of the form

$$first(rec(x_0, x_1, \ldots, x_m, y_1, \ldots, y_k, f_0, f_1, \ldots, f_n)$$
$$(e_0, e'_1, \ldots, e'_m, e''_1, \ldots, e''_k, g_1, \ldots, g_n))$$

3.2 Semantics of FTAG

So far, we have only considered only the syntax of the calculus used for representing feature structures and FTAG grammars. In this section, we consider the mathematical modelling of the calculus. This can be used to show that the set of equations describing a grammar will always have a solution, which we can consider as the denotation of the grammar.

The model that we present here is based on the work by Rounds and Kasper [Rounds et al. 1986] and in particular their notion of satisfiability of formulae. Let F be the space of partial functions (with the partial ordering $\sqsubseteq$, the standard ordering on partial functions) defined by $F = (L \rightarrow F) + A$ where A is set of atoms and L is set of labels. This space has been characterized by Pereira and Sheiber [Pereira et al. 1984]. Any expression e (which is not a function) can be thought as upward closed subset of F (the set of partial functions which satisfy the description e). Note that if a partial function satisfies a description then so will any function above it. We let U(F) stand for the collection of upward closed subsets of F. Expressions are interpreted relative to an environment (since we have variables as expressions, we need to consider environments which map variables to a member of U(F)). Functions get interpreted as continuous functions in the space $U(F) \rightarrow U(F)$, with the environment mapping function variables to functions on U(F). Note that the ordering on U(F) is the inverse of set inclusion, since more functions satisfy the description of a more general feature structure.

Because of space limitations, we cannot go into the details of the interpretations function. Roughly, the interpretation is as follows. We interpret the expression a as the set containing just the atom "a"; the expression $l : e$ is interpreted as the set of functions which map l to an element in the set denoted by e; conjunction and disjunction are treated as intersection and union respectively except that we have to ensure that any value assigned to a variable in one of the conjuncts is the same as the value assigned to the same variable in the other conjunct.

Since the grammar is given by a set of equations, the denotation is given by the least solution. This is obtained by considering the function corresponding to the set of equations in the standard way, and obtaining its least fixpoint. Details of these issues may be found in [Vijayashanker 19

In [Vijayashanker 1987], we have shown that any set of equations has a solution. Thus, we can give semantics for recursive set of equations which may be used to describe cyclic feature structure. For example, we give the solution for equations such as

$$x = f : x \wedge g : a$$

As shown in [Vijayashanker 1987], we can obtain the least fixed-point by assuming the least value for x (which is the entire set of partial functions, or the interpretation of *NIL*) and obtaining better and better approximations. The least upper bound of these approximations (which will give the least fixed-point) corresponds to the required cyclic structure, as desired.

4 Conclusions and Future Work

We have shown a method of embedding TAG's in a feature structure based framework. This system takes advantage of the extended domain of locality of TAG's and allows lingusitic statements about cooccurence of features of dependent items to be stated within elementary trees. We have shown that we can make a clearer statement of adjoining constraints in FTAG's than in TAG's. The specification of local constraints in a TAG is by enumeration, which is not satisfactory from the linguistic point of view. We show that in FTAG, we can avoid such specifications, instead the declarative statements made about nodes are sufficient to ensure that only the appropriate trees get adjoined at a node. Furthermore, we also illustrate how duplication of information can be avoided in FTAG's in comparison with TAG's. It can be shown that analyses that require extensions of TAG's using multi-component adjoining (simultaneous adjunction of a set of trees in distinct nodes of an elementary tree) as defined in [Joshi 1987, Kroch 1987], can be easily stated in FTAG's.

It is possible to parse an FTAG grammar using the Earley-style parser given by [Schabes et al. 1988]. This Earley-style parser can extended in the same way that Sheiber extended the Earley parser for PATR-II [Shieber 1985b]. The reason this extension of the TAG parser to one for FTAG is possible follows from the fact that the treatment of having the t and b feature structures for every node in FTAG is compatible with the characterization, adopted in the parsing algorithm in [Schabes et al. 1988], of a node in terms of two substrings.

In [Vijayashanker 1987], we have proposed a restricted version of FTAG In a manner similar to GPSG, we place a bound on the information content of feature structures associated with the nodes of trees used in the grammar. The resulting system, RFTAG, generates the same language as TAG's, and yet retains an increased descriptive and generative capacity due to the extended domain of locality of TAG's.

Finally, in this paper, we have briefly discussed a calculus to represent FTAG grammars. This calculus is an extention of the Rounds-Kasper logic for feature structures. The extentions deal with λ-abstraction over feature structures and function application, which is used to characterize auxiliary trees and the adjunction operation. [Vijayashanker 1987] gives a detailed description of this calculus and its semantics.

References

Joshi, A. K. 1985. How Much Context-Sensitivity is Necessary for Characterizing Structural Descriptions — Tree Adjoining Grammars. In: D. Dowty, L. Karttunen, and A. Zwicky, Eds., *Natural Language Processing — Theoretical, Computational and Psychological Perspective*. Cambridge University Press, New York, NY.

Joshi, A. K. 1987. An Introduction to Tree Adjoining Grammars. In: A. Manaster-Ramer, Ed., *Mathematics of Language*. John Benjamins, Amsterdam.

Joshi, A. K., Levy, L. S., and Takahashi, M. 1975. Tree Adjunct Grammars. *J. Comput. Syst. Sci.*, 10(1).

Kasper, R. and Rounds, W. C. 1986. A Logical Semantics for Feature Structures. In: 24th *meeting Assoc. Comput. Ling.*

Kroch, A. 1987. Subjacency in a Tree Adjoining Grammar. In: A. Manaster-Ramer, Ed., *Mathematics of Language*. John Benjamins, Amsterdam.

Kroch, A. and Joshi, A. K. 1985. *Linguistic Relevance of Tree Adjoining Grammars*. Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Pereira, F. C. N. and Shieber, S. 1984. The Semantics of Grammar Formalisms Seen as Computer Languages. In: 10th *International Conference on Computational Linguistics*.

Rounds, W. C. and Kasper, R. 1986. A complete Logical Calculus for Record Structures Representing Linguistic Information. In: *IEEE Symposium on Logic and Computer Science*.

Santorini, B. 1986. *The West Germanic Verb-Raising Construction: A Tree Adjoining Grammar Analysis*. Master's thesis, University of Pennsylvania, Philadelphia, PA.

Schabes, Y. and Joshi, A. K. 1988. An Earley-Type Parsing Algorithm for Tree Adjoining Grammars. In: 26th *meeting Assoc. Comput. Ling.*

Shieber, S. M. *An Introduction to Unification-Based Approaches to Grammar*. Presented as a Tutorial Session 23rd meeting Assoc Comput. Ling., 1985.

Shieber, S. M. 1985. Using Restriction to Extend Parsing Algorithms for Complex-feature-based Formalisms. In: 23rd *meeting Assoc. Comput. Ling.*address and pages82–93.

Vijayashanker, K. 1987. *A Study of Tee Adjoining Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, Pa.