FACTORING RECURSION AND DEPENDENCIES: AN ASPECT OF TREE ADJOINING GRAMMARS (TAG) AND A COMPARISON OF SOME FORMAL PROPERTIES OF TAGS, GPSGS, PLGS, AND LPGS *

Aravind K. Joshi
Department of Computer and Information Science
R. 268 Moore School
University of Pennsylvania
Philadelphia, PA 19104

1. INTRODUCTION

During the last few years there is vigorous activity in constructing highly constrained grammatical systems by eliminating the transformational component either totally or partially. There is increasing recognition of the fact that the entire range of dependencies that transformational grammars in their various incarnations have tried to account for can be satisfactorily captured by classes of rules that are non-transformational and at the same time highly constrianed in terms of the classes of grammars and languages that they define.

Two types of dependencies are especially important: subcategorization and filler-gap dependencies. Moreover, these dependencies can be unbounded. One of the motivations for transformations was to account for unbounded dependencies. The so-called non-transformational grammars account for the unbounded dependencies in different ways. In a tree-adjoining grammar (TAG), which has been introduced earlier in (Joshi, 1982), unboundedness is achieved by factoring the dependencies and recursion in a novel and, we believe, in a linguistically interesting manner. All dependencies are defined on a finite set of basic structures (trees) which are bounded. Unboundedness is then a corollary of a particular composition operation called adjoining. There are thus no unbounded dependencies in a sense.

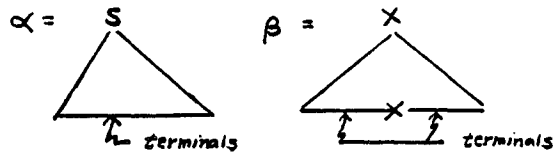
In this paper, we will first briefly describe TAG's, which have the following important properties: (1) we can represent the usual transformational relations more or less directly in TAG's, (2) the power of TAG's is only slightly more than that of context-free grammars (CFG's) in what appears to be just the right way, and (3) TAG's are powerful enough to characterize dependencies (e.g., subcategorization, as in verb subcategorization, and filler-gap dependencies, as in the case of moved constitutents in wh-questions) which might be at unbounded distance and nested or crossed. We will then compare some of the formal properties of TAG's, GPSG's, PLG's, and LFG's, in particular, concerning (1) the types of languages, reflecting different patterns of dependencies that can or cannot be generated by the different types of grammars, (2) the degree of free word ordering permitted by different grammars, and (3) parsing complexity of the different grammars.

*GPSG: Generalized phrase structure grammar, PLG: Phrase linking grammar, and LFG: Lexical functional grammar.

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

2. TREE ADJOINING GRAMMAR (TAG)

A tree adjoining grammar (TAG), G = (I,A) consists of two finite sets of elementary trees. The trees in I will be called the initial trees and the trees in A, the auxiliary trees. A tree α is an initial tree if the root node of is labeled S and the frontier nodes are all terminal symbols (the interior nodes are all non-terminals). A tree β is an auxiliary tree if the root node of β is labeled by a non-terminal, say, X, and the frontier nodes are all terminals except one which is also labeled X, the same label as that of the root. The node labeled by X on the frontier will be called the foot node of β. The internal nodes are non-terminals.



As defined above, the initial trees and the auxiliary trees are not constrained in any manner other than as indicated above. The idea, however, is that both the initial and the auxiliary trees will be minimal in some sense. An initial tree will correspond to a minimal sentential tree (i.e., for example, without recursing on any non-terminal) and an auxiliary tree, with the root node and the foot node labeled X, will correspond to a minimal structure that must be brought into the derivation, if one recurses on X.

* I wish to thank Bob Berwick, Tim Finin, Jean Gallier, Gerald Gazdar, Ron Kaplan, Tony Kroch, Bill Marsh, Mitch Marcus, Ellen Prince, Geoff Pullum, R. Shyamasundar, Bonnie Webber, Scott Weinstein, and Takashi Yokomori for their valuable comments

We will now define a composition operation called adjoining (or adjunction) which composes an auxiliary tree β with a tree γ. Let γ be tree with a node labeled X and let β be an auxiliary tree with the root labeled X also. Note that β must have, by definition, a node (and only one) labeled X on the frontier. Adjoining can now be defined as follows. If β is adjoining to γ at the node n then the resulting tree γ' is as shown in Fig.1.

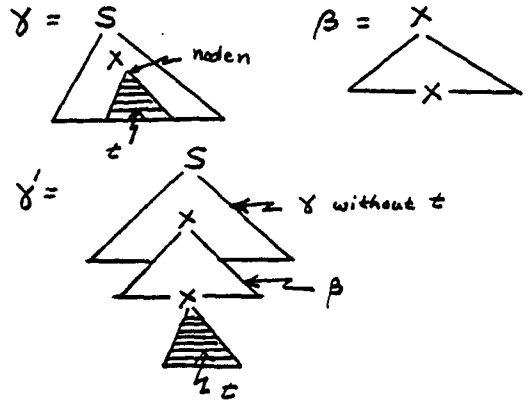
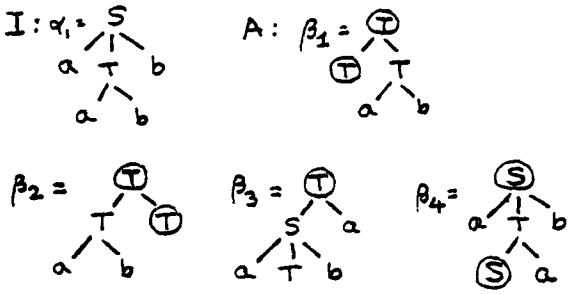


FIG. 1.

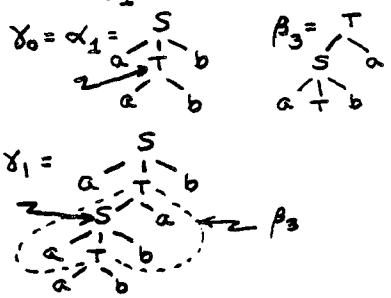
The tree t dominated by X in γ is excised, β is inserted at the node n in and the tree t is attached to the foot node (labeled X) of β, i.e., β is inserted or 'adjoined' to the node n in γ pushing t downwards. Note that adjoining is not a substitution operation in the usual sense.

Example 2.1: Let G = (I,A) be a TAG where

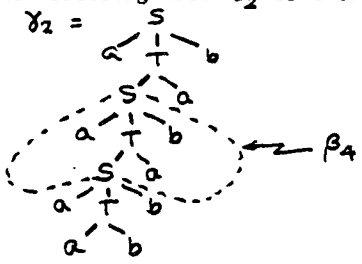


The root node and the foot node of each auxiliary tree is circled for convenience. Let us look at some derivations in G.

β_3 will be adjoined to γ_0 at the indicated node in γ_0. The resulting tree is then γ_1



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



Note that γ_0 is an initial tree, a sentential tree. The derived trees γ_1 and γ_2 are also sentential trees.

We will now define

T(G): The set of all trees derived in G starting from the initial trees in I. This set will be called the tree set of G.

L(G): The set of all terminal strings of the trees in T(G). This set will be called the string language (or language) of G.

The relationship between TAG's CFG's and the corresponding string languages can be summarized as follows (Joshi, Levy, and Takahashi, 1975).

Theorem 2.1: For every CFG, G', there is an equivalent TAG, G, both weakly and strongly.

Theorem 2.2: For every TAG, G, one of the following statements holds:

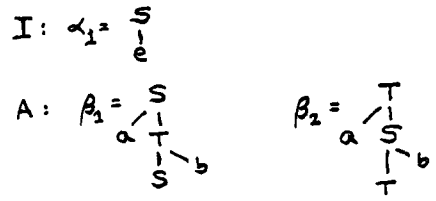
(a) there is a cfg, G', that is both weakly and strongly equivalent to G,

(b) there is a cfg, G', that is weakly equivalent to G but not strongly equivalent to G, or

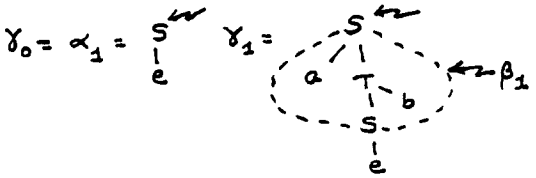
(3) there is no cfg, G', that is weakly equivalent to G.

Parts (a) and (c) appear in (Joshi, Levy, and Takahashi, 1975). Part (b) is implicit in that paper, but it is important to state it explicitly as we have done here. For the TAG, G, in Example 2.1, it can be shown that there is a CFG, G', such that G' is both weakly and strongly equivalent to G. Examples 2.2 and 2.3 below illustrate parts (b) and (c) respectively.

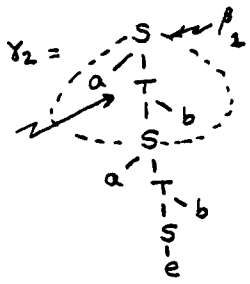
Example 2.2: Let G = (I,A) be a TAG where



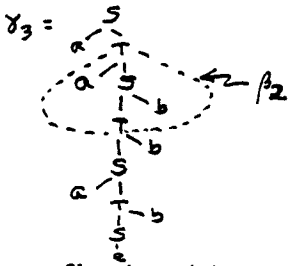
Some derivations in G.



γ1 = γ0 with β1 adjoined at the indicated node S in γ0



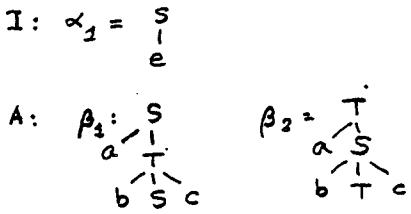
γ2 = γ1 with β1 adjoined at S as indicated in γ1.



γ3 = γ2 with β2 adjoined at T as indicated in γ2.

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

Example 2.3: Let G = (I,A) be a TAG where



It can be shown that L(G) = L1 = { w e c^n/ $n \geq 0$}, w is a string of a's and b's such that (1) the number of a's = the number of b's and (2) for any initial substring of w, the number of a's $\geq$ the number of b's.}

L1 can be characterized as follows. We start with the language L = { $(ba)^n e\, c^n$/ $n \geq 0$ }. L1 is then obtained by taking strings in L and moving (dislocating) some a's to the left. It can be shown that L1 is a strictly context-sensitive language (csl), thus there can be no CFG that is weakly equivalent to G.

TAG's have more power than CFG's, however, the extra power is quite limited. The language L1 has equal number of a's ,b's nad c's; however, the a's and b's are mixed in a certain way. The Language L2 ={$a^n b^n e\, c^n$/ n 0} is similar to L1, except that all a's come before all b's. TAG's are not powerful to generate L2. The so-called copy lnguage L3 = {w e w /w $\in$ {a,b}* } also cannot be generated by a TAG.

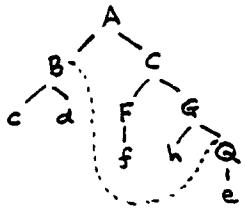
The fact that TAG's cannot generate L2 and L3 is important, because it shows that TAG's are only slightly more powerful than CFG's. The way TAG's acquire this power is linguistically significant. With some modifications of TAG's or rather the operation of adjoining, which is linguistically motivated, it is possible to generate L2 and L3, but only in some special ways. (This modification consists of allowing for the possibility for checking left-right tree context(in terms of a proper analysis) as well as top-bottom tree context (in terms of domination) around the node at which adjunction is made. This is the notion of local constraints in (Joshi and Levy,1981)). Thus L2 and L3 in some ways characterize the limiting cases of context-sensitivity that can be achieved by TAG's and TAG's with local constraints.

In (Joshi,Levy, and Takahashi,1975) it is also shown that

$$\text{CFL's} \subsetneq \text{TAL's} \subsetneq \text{IL's} \subsetneq \text{CSL's}.$$

where IL's denotes indexed languages.

3. We will now consider TAG's with links. The elementary trees (initial and auxiliary trees) are the appropriate domains for characterizing certain dependencies. The domain of the dependency is defined by the elementary tree itself. However, the dependency can be characterized explicitly by introducing a special relationship between certain specified pairs of nodes of an elementary tree. This relationship is pictorially exhibited by an arc (a dotted line) from one node to the other. For example, in the tree below, the nodes labeled B and Q are linked,



We will require the following conditions to hold for a link in an elementary tree. If a node n1 is linked to a node n2 then (1) n2 c-commands n1 and (2) n1 dominates a null string (or a terminal symbol in the non-linguistic formal grammar examples).

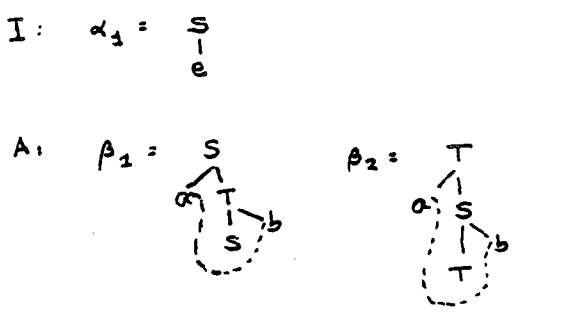
The notion of a link introduced here is closely related to that of Peters and Ritchie (1982).

A TAG with links is a TAG where some of the elementary trees may have links as defined above. Henceforth, we may often refer to a TAG with links as just a TAG. Links are defined on the elementary trees. However, the important idea is that the composition operation of adjoining will preserve the links. Links defined on the elementary trees may become stretched as the derivation proceeds.

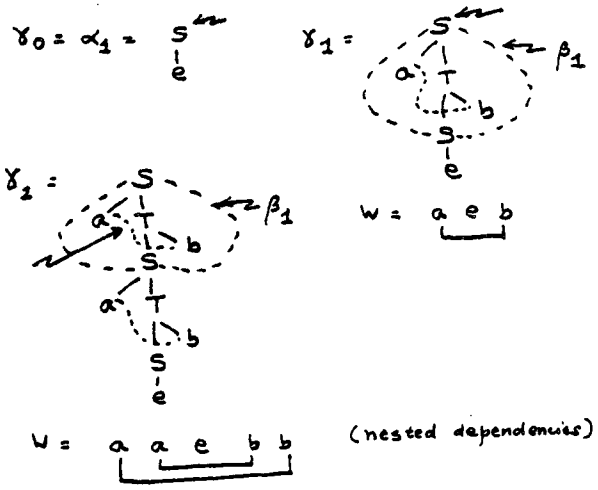
In a TAG the dependencies are defined on the elementary trees(which are bounded) and these dependencies are then preserved by the adjoining(recursive) operation. This is how recursion and dependencies are factored in a TAG. This is in contrast to transformational grammars (TG) where recursion is defined in the base and the transformations essentially carry out the checking of the dependencies. The PLG's and LFG's share this aspect of TG,i.e., recursion builds up a set of structures, some of which are filtered out by transformations in a TG, by the constraints on linking in a PLG, and by the constraints introduced via functional structures in LFG. In a GPSG on the other hand, recursion and the checking of the dependencies go hand in hand in a sense. In a TAG, dependencies are defined initially on bounded structures and recursion simply preserves them.

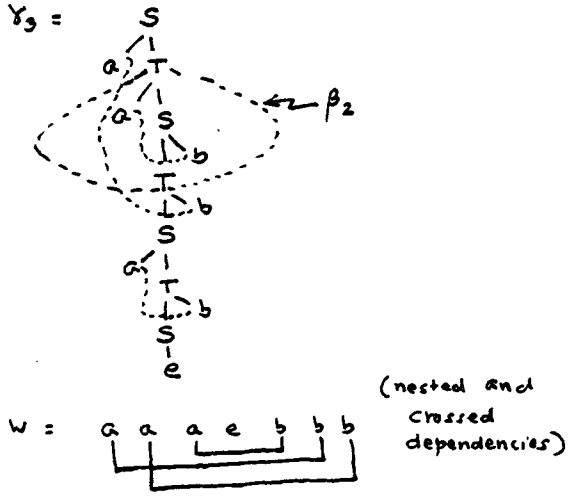
In the APPENDIX we have given some examples to show how certain sentences could be deirved in a TAG.

Example 2.4: Let G = (I,A) be a TAG with links where



Some derivations in G:





β_1 and β_2 each have one link. γ_2 and γ_3 show how the linking is preserved in adjoining. In γ_3 one of the links is stretched. It should be clear now, how, in general, the links will be preserved during the derivation. We note in this example that in γ_2 the dependencies between the a's and the b's as reflected in the terminal string are properly nested, while in γ_3 two of them are properly nested, and the third one is cross-serial and it is crossed with respect to the nested ones. The two elementary trees β_1 and β_2 have only one link each. The nestings and crossings in γ_2 and γ_3 are the result of adjoining. There are two points to note here: (1) TAG's with links can characterize certain cross-serial dependencies as well as, of course, nested dependencies. (2) The cross-serial dependencies as well as the nested dependencies arise as a result of adjoining. But this is not the only way they can arise. It is possible to have two links in an elementary tree which represent crossed or nested dependencies, which will then be preserved during the derivation.

It is clear from Example 2.4 that the string language of TAG with links is not affected by the links. Thus if G is a TAG with links. Then L(G)=L(G') where G' is a TAG which is obtained from G by removing all the links in the elementary trees of G. The links do not affect the weak generative capacity. However, they make certain aspects of the structural description explicit, which is implicit in the TAG without the links.

TAG's (or TAL's) also have the following three important properties:

(1) Limited cross-serial dependencies: Although TAG's permit cross-serial dependencies, these are restricted. The restriction is that if there are two sets of crossing dependencies, then they must be either disjoint or one of them must be properly nested inside the other. Hence, languages such as the double copy language, L4 = {w e w e w / w $\in$ {a,b}*} or L5 = {$a^n b^n c^n d^n e^n$/ n $\geq$ 1} cannot be generated by TAG's. For details, see (Joshi,1983).

(2)Constant growth property: In a TAG,G,at each step of the derivation, we have a sentential tree with the terminal string which is a string in L(G). As we adjoin an auxiliary tree, we augment the length of the terminal string by the length of the terminal string of β (not counting the single non-terminal symbol in the frontier of β).Thus for any string, w, of L(G), we have

$$|w| = |w_k| + a_1|w_1| + a_2|w_2| + \cdots + a_m|w_m|$$

$$a_i \geq 0 \ , \ 1 \leq i \leq m$$

where $w_k$is the terminal string of some initial tree and w_i, $1 \leq i \leq m$, the terminal string of the i-th auxiliary tree, assuming there are m auxiliary trees. Thus w is a linear combination of the length of the terminal string of some initial tree and the lengths of the terminal strings of the auxiliary trees. The constant growth property severely restricts the class of languages generated by TAG's. Hence,languages such as L6 = { a^{2^n} / n $\geq$ 1} or L8 ={a^{n^2} /n $\geq$ 1} cannot be generated by TAG's.

(3)Polynomial parsing:TAL's can be parsed in time $O(n^4)$(Joshi and Yokomori, 1983). Whether or not an $O(n^3)$ algorithm exists for TAL's is not known at present.

3. A COMPARISION OF GPSG's,TAG's,PFG's,and LFG's WITH RESPECT TO SOME OF THEIR FORMAL PROPERTIES

TABLE 1 lists (i) a set of languages reflecting different patterns of dependencies that can or cannot be generated by the different types of grammars, and (ii) the three properties just mentioned above.

As regards the degree of free word order permitted by each grammar, the languages 1,2,3,4,5, and 6 in TABLE 1 give some idea of the degree of freedom. The language in 3 in TABLE 1 is the extreme case where the a's, b's,and c's can be any order, as long as the number of a's =the number of b's=the number of c's. GPSG's and TAG's cannot generate this language (although for TAG's a proof is not in hand yet). LFG's can generate this language.

In a TAG for each elementary tree, we can add more elementary trees, systematically generated from the given tree to provide additional freedom of word order (in a somewhat similar fashion as in (Pullum,1982)). Since the adjoining operation in a TAG gives some additional power to a TAG beyond that of a CFG, this device of augmenting the set of elementary trees should give more freedom, for example, by allowing some limited scrambling of an item outside of the constituent it belongs to. Even then a TAG does not seem to be capable of generating the language in 3 in TABLE 1. Thus there is extra freedom but it is quite limited.

TABLE 1

	GPSG (and CFG)	TAG (with or without local constraints)	PLG	LFG
1. Language obtained by starting with $L=\{(ba)^n c^n / n \geq 1\}$ and then dislocating some a's to the left.	no	yes	yes	yes
2. Same as 1 above except that the dislocated a's are to the left of all b's.	no	yes	yes	yes
3. L={w / w is string of equal number of a's,b's and c's but mixed in any order}	no	no(?)	yes	yes
4. $L=\{x c^n y / n \geq 1$, x,y are strings of a's and b's such that the number of a's in x and y = the number of b's in x and y= n}	no	no	yes	yes
5. Same as above except that the length of x = length of y.	no	yes	no(?)	yes(?)
6. $L=\{w c^n / n \geq 1$, w is string of a's and b's and the number of a's in w = the number of b's in w = n}	no	yes	yes(?)	yes(?)
7. $L=\{a^n b^n c^n / n \geq 1\}$	no	yes	no	yes
8. $L=\{a^n b^n c^n d^n / n \geq 1\}$	no	yes	no	yes
9. $L=\{a^n b^n c^n d^n e^n / n \geq 1\}$	no	no	no	yes
10. L= {w w/ w is string of a's and b's}(copy language)	no	yes	yes(?)	yes
11. L={w w w/ w is string of a's and b's}(double copy language)	no	no	?	yes
12. $L=\{a^n c^m b^n d^m / m \geq 1, n \geq 1\}$	no	no	no(?)	?
13. $L=\{a^n b^n c^p / n \geq 1, p \neq n\}$	no	yes	?	yes(?)
14. $L=\{a^{2^n} / n \geq 1\}$	no	no	no(?)	yes
15. $L=\{a^{n^2} / n \geq 1\}$	no	no	no(?)	yes
16. Limited cross-serial dependencies.	no	yes	?	no(?)
17. Constant growth property	yes	yes	yes(?)	no
18. Polynomial parsing	yes	yes	?	no(?)

Notation: ?: answer unknown to the author. yes(?): conjectured yes
no(?): conjectured no.

REFERENCES

[1] Gazdar,G.,"Phrase structure grammars" in The Nature of Syntactic Representations(eds. P. Jacobson and G.K. Pullum),D. Reidel, Dordrecht, (to appear).

[2] Joshi, A.K. and Levy, L.S.,"Phrase structure trees bear more fruit than you would have thought", AJCL, 1982.

[3] Joshi, A.K., Levy, L.S., and Takahashi, M.,"Tree adjunct grammars", Journal of the Computer and System Sciences,1975.

[4] Joshi, A.K.,"How much context-sensitivity is required to provide adequate structural descriptions ?", in Natural language processing: Psycholinguistic, Theoretical, and Computational Perseptives, (eds. Dowty, D., Karttunen, L., and Zwicky, A.), Cambridge University Press, (to appear).

[5] Joshi, A.K. and Yokomori, T.,"Parsing of tree adjoining grammars", Tech. Rep. Department of Computer and Information Science, University of Pennsylvania,1983.

[6] Joshi, A.K. and Kroch, T., "Linguistic significance of TAG's" (tentative title), forthcoming.

[7] Kaplan R. and Bresnan J.W., "Lexical functional grammar-a formal system for grammatical representation", in The Mental Representation of Grammatical Relations(ed. Bresnan, J.), MIT Press, 1983.

[8] Peters, S. and Ritchie, R.W., "Phrase linking grammars",Tech. Rep. University of Texas at Austin, Department of Linguistics, 1982.

[9] Pullum, G.K.,"Free word order and phrase structure rules", in Proceeding of NELS 12(eds. Pustejovsky, J. and Sells, P.), Amherst, MA, 1982.

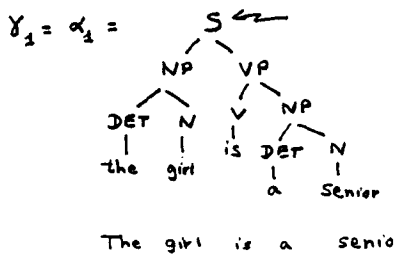
APPENDIX

We will give here some examples to show how certain sentences could be derived in a TAG. For further details about this TAG and its linguistic relevance, see (Joshi,1983 and Joshi and Kroch, forthcoming). Only the relevant trees of the TAG, G=(I,A) are shown below. The following points are worth noting: (1)In a TAG the derivation starts with an initial tree. The appropriate lexical insertions are made for the initial tree and the corresponding constraints as specified by the lexicon can be checked (e.g., agreement and subcategorization). Then as the derivation proceeds, as each auxiliary tree is brought into the derivation, the appropriate lexical items are inserted and the constraints checked. Thus in a TAG, lexical insertion goes hand in hand with the derivation. (2) Each one of the two finite sets, I and A can be quite large, but these sets need not be explicitely listed. The trees in I roughly correspond to all the 'minimal' sentences corresponding to different subcategorization frames together with the 'transforms' of these sentences. We could , of course, provide rules for obtaining the trees in I from a given subset of I. These rules achieve the effect of conventional transformational rules, however, these rules can be formulated not as the usual transformational rules but directly as tree rewriting rules, since both the domains and the co-domains of the rules are finite. Introduction of links can be considered as a part of this rewriting. In any case, these rules will be abbreviatory in the sense that they will generate only finite sets of trees. Their adoption will be only a matter of convenience and does not affect the TAG in any essential manner. The set of auxiliary trees is also finite. Again these trees could themselves be 'derived' from the corresponding trees in I by introducing appropriate tree rewriting rules. Again these rules will be abbreviatory only as discussed above. It is in this sense that the trees in I and A capture the usual transformational relations more or less directly.

Some derivations:

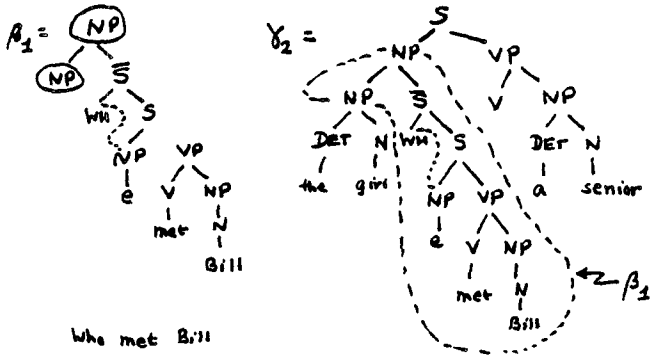
(1)The girl who met Bill is a senior.

We start with the initial tree α_1 with the appropriate lexical insertions.



The girl is a senior

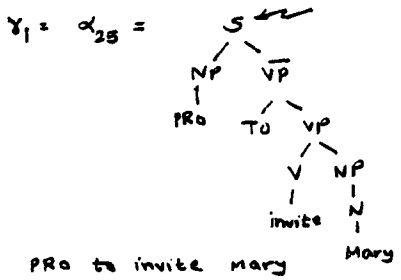
Adjoining β_1 (with the appropriate lexical insertions) to α_1 at the indicated node in γ_1, we obtain γ_2.



Who met Bill

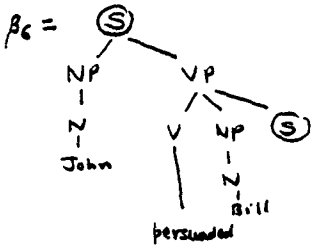
The girl who met Bill is a senior

(2)John persuaded Bill to invite Mary.

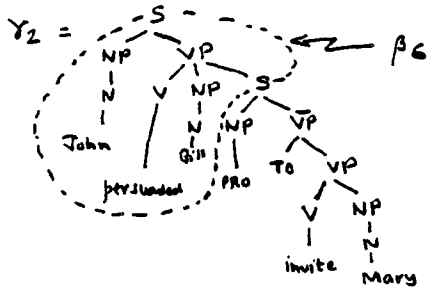


PRO to invite Mary

Adjoining β_6 to γ_1 at the indicated node in γ_1, we obtain γ_2.

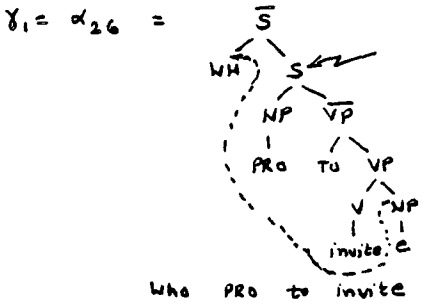


John persuaded Bill S



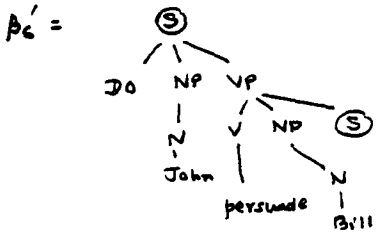
John persuaded Bill to invite Mary

(3)Who did John persuade Bill to invite ?

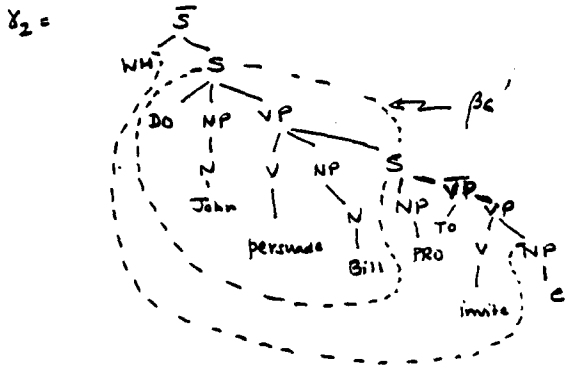


Who PRO to invite

Adjoining β_6' to γ_1 at the indicated node in γ_1, we obtain γ_2.



did John persuade Bill S

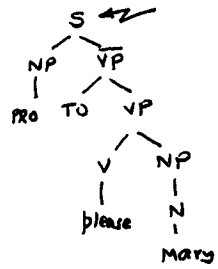


Who did John persuade Bill to invite ?

Note the link in γ_1 is 'preserved' in γ_2 , it is 'stretched' resulting in the so-called unbounded dependency.

(4)John tried to please Mary.

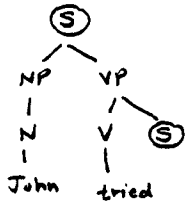
$\gamma_1 = \alpha_{25}$ =



PRO to please Mary

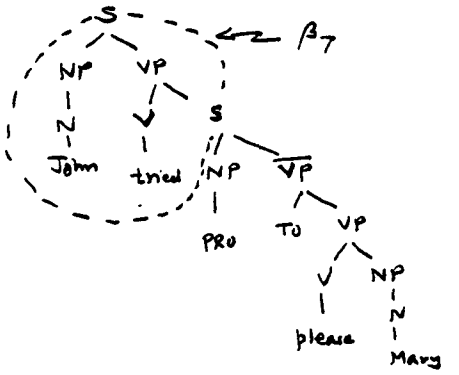
Adjoining β_7 to γ_1 at the indicated node in γ_1 we obtain γ_2.

β_7 =



John tried S

γ_2 =

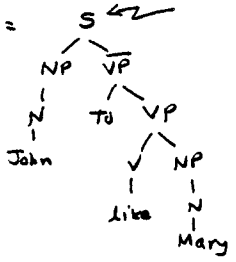


John tried PRO to please Mary

On the other hand

(5)John seems to like Mary. could be derived as follows. We will start with α'_{25}.

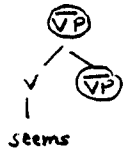
$\gamma_1 = \alpha'_{25}$ =



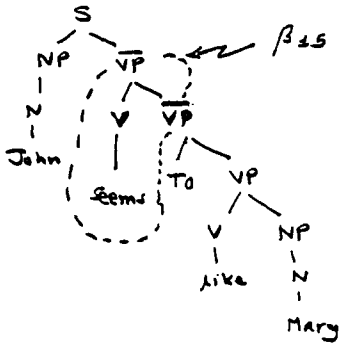
John to like Mary

Adjoining β_{15} to γ_1 at the indicated node in γ_1 , we obtain γ_2 .

β_{15} =



γ_2 =



John seems to like Mary