



1-1-1990

The Convergence of Mildly Context-Sensitive Grammar Formalisms

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

K. Vijay Shanker
University of Delaware

David Weir
Northwestern University

Follow this and additional works at: https://repository.upenn.edu/cis_reports

 Part of the [Computer Sciences Commons](#)

Recommended Citation

Aravind K. Joshi, K. Vijay Shanker, and David Weir, "The Convergence of Mildly Context-Sensitive Grammar Formalisms", . January 1990.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-01.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/539
For more information, please contact repository@pobox.upenn.edu.

The Convergence of Mildly Context-Sensitive Grammar Formalisms

Abstract

Investigations of classes of grammars that are *nontransformational* and at the same time highly constrained are of interest both linguistically and mathematically. Context-free grammars (CFG) obviously form such a class. CFGs are not adequate (both weakly and strongly) to characterize some aspects of language structure. Thus how much more power beyond CFG is necessary to describe these phenomena is an important question. Based on certain properties of tree adjoining grammars (TAG) an approximate characterization of class of grammars, *mildly context-sensitive grammars* (MCSG), has been proposed earlier. In this paper, we have described the relationship between several different grammar formalisms, all of which belong to MCSG. In particular, we have shown that head grammars (HG), combinatory categorial grammars (CCG), and linear indexed grammars (LIG) and TAG are all weakly equivalent. These formalisms are all distinct from each other at least in the following aspects: (a) the formal objects and operations in each formalism, (b) the domain of locality over which dependencies are specified, (c) the degree to which recursion and the domain of dependencies are factored, and (d) the linguistic insights that are captured in the formal objects and operations in each formalism. A deeper understanding of this convergence is obtained by comparing these formalisms at the level of the derivation structures in each formalism. We have described a formalism, the linear context-free rewriting system (LCFR), as a first attempt to capture the closeness of the derivation structures of these formalisms. LCFRs thus make the notion of MCSGs more precise. We have shown that LCFRs are equivalent to multicomponent tree adjoining grammars (MCTAGs), and also briefly discussed some variants of TAGs, lexicalized TAGs, feature structure based TAGs, and TAGs in which local domination and linear precedence are factored TAG(LD/LP).

Disciplines

Computer Sciences

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-01.

**The Convergence Of Mildly
Context-Sensitive
Grammar Formalisms**

**MS-CIS-90-01
LINC LAB 161**

**Aravind K. Joshi
K. Vijay Shanker
David Weir**

**Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389**

January 1990

Acknowledgements:

**This research was supported in part by DARPA grant
N00014-85-K-0018, NSF grants MCS-8219196-CER,
IRI84-10413-A02, US Army grants DAA29-84-K-0061,
DAA29-84-9-0027 and Advanced Technology Center
(PA) grant #309.**

THE CONVERGENCE OF MILDLY CONTEXT-SENSITIVE GRAMMAR FORMALISMS*

Aravind K. Joshi[†] K. Vijay-Shanker[‡] David Weir[§]

November 29, 1989

*This paper is a revised version of a paper presented by the first author at the conference on "The Processing of Linguistic Structure" sponsored by the Systems Development Foundation held at Santa Cruz, CA, January 1987. We want to thank Jean Gallier, Tony Kroch, Mitch Marcus, Remo Pareschi, Yves Schabes, Mark Steedman, Ramesh Subrahmanyam, and Bonnie Webber for valuable discussion about several aspects of this paper. This work was partially supported by NSF grant IRI84-10413 A02, US Army grant DAA6-29-84K-0061, and DARPA grant N0014-85-K0018, and Advanced Technology Center (PA) grant #309.

[†]Department of Computer and Information Science, Room 555, Moore School, University of Pennsylvania, Philadelphia, PA 19104

[‡]Department of Computer Science, University of Delaware, 103 Smith Hall, Newark, DE 19716

[§]Department of Electrical Engineering and Computer Science, The Technological Institute, Northwestern University, Evanston, IL 60208

Abstract

Investigations of classes of grammars that are *nontransformational* and at the same time highly constrained are of interest both linguistically and mathematically. Context-free grammars (CFG) obviously form such a class. CFGs are not adequate (both weakly and strongly) to characterize some aspects of language structure. Thus how much more power beyond CFG is necessary to describe these phenomena is an important question. Based on certain properties of tree adjoining grammars (TAG) an approximate characterization of class of grammars, *mildly context-sensitive grammars* (MCSG), has been proposed earlier. In this paper, we have described the relationship between several different grammar formalisms, all of which belong to MCSG. In particular, we have shown that head grammars (HG), combinatory categorial grammars (CCG), and linear indexed grammars (LIG) and TAG are all weakly equivalent. These formalisms are all distinct from each other at least in the following aspects: (a) the formal objects and operations in each formalism, (b) the domain of locality over which dependencies are specified, (c) the degree to which recursion and the domain of dependencies are factored, and (d) the linguistic insights that are captured in the formal objects and operations in each formalism. A deeper understanding of this convergence is obtained by comparing these formalisms at the level of the derivation structures in each formalism. We have described a formalism, the linear context-free rewriting system (LCFR), as a first attempt to capture the closeness of the derivation structures of these formalisms. LCFRs thus make the notion of MCSGs more precise. We have shown that LCFRs are equivalent to multicomponent tree adjoining grammars (MCTAGs), and also briefly discussed some variants of TAGs, lexicalized TAGs, feature structure based TAGs, and TAGs in which local domination and linear precedence are factored TAG(LD/LP).

1 INTRODUCTION

Since the late 1970's, there has been vigorous activity in constructing highly constrained grammatical systems by eliminating the transformational component either totally or partially. This was caused by the increasing recognition of the fact that the entire range of dependencies that the transformational grammars in their various incarnations have tried to account for can be captured satisfactorily by classes of grammars that are *nontransformational*, and at the same time are highly constrained in terms of the classes of grammars and languages they define. Peters and Ritchie (1969) showed that context-sensitive grammars (CSG) if used for analysis (and not for generation), thus providing more descriptive power than context-free grammars (CFG), have the same weak generative capacity as CFG. This result was generalized by Joshi and Levy (1978) to Boolean combinations of contextual predicates and domination predicates.

In the early 1980's, Gazdar (1982) proposed a grammatical formalism (which later became the GPSG formalism, Gazdar, Klein, Pullum, and Sag (1985)) whose weak generative capacity is the same as the CFG, but which is adequate to describe various syntactic phenomena previously described in transformational terms. Gazdar was careful to note that his results did not mean that syntactic phenomena which required formal power beyond CFG did not exist, his claim was that, as far as the range of phenomena known at that time, CFGs seemed to be quite adequate. In the late 1980's, some clear examples of natural language phenomena were discovered that required formal power beyond CFG e.g., Shieber (1984) and Culy (1984), for an argument from weak generative capacity, and Bresnan, Kaplan, Peters, and Zaenen (1983), for an argument from strong generative capacity. Hence, the question of how much power beyond CFG is necessary to describe these phenomena became important.

An extension of CFG was proposed by Pollard (1984), called Head Grammars (HG), which introduced some wrapping operations beyond the concatenation operation in CFG. Some formal properties of HG were investigated by Roach (1984). HGs like CFGs are string generating systems.

Joshi, Levy, and Takahashi (1975) introduced a grammatical formalism, called Tree Adjoining Grammars (TAG), a tree generating system, and investigated some of their formal properties. Joshi (1985) showed how TAGs factor recursion and the domain of dependencies in a novel way, leading to 'localization' of dependencies, their long distance behavior following from the operation of composition, called 'adjoining'. TAGs have more power than CFGs and this extra power is a

corollary of factorization of recursion and the domain of dependencies. This extra power appeared to be adequate for the various phenomena requiring formal power more than CFG. The linguistic significance of TAGs has been discussed in Joshi (1985), Kroch and Joshi (1985, 1986), Kroch (1986), and Kroch and Santorini (1986). Based on the formal properties of TAGs, (Joshi (1985)), proposed that the class of grammars that is necessary for describing natural languages might be characterized as *mildly context-sensitive grammars* (MCSG, MCSL for the corresponding languages) possessing at least the following properties: 1) context-free languages (CFL) are properly contained in MCSL; 2) languages in MCSL can be parsed in polynomial time; 3) MCSGs capture only certain kinds of dependencies, e.g., nested dependencies and certain limited kinds of crossing dependencies (e.g., in the subordinate clause constructions in Dutch or some variations of them, but perhaps not in the so-called MIX (or Bach) language, which consists of equal numbers of a's, b's, and c's in any order 4) languages in MCSL have constant growth property, i.e., if the strings of a language are arranged in increasing order of length then two consecutive lengths do not differ by arbitrarily large amounts. In fact, any given length can be described as a linear combination of a finite set of fixed lengths. This property is slightly weaker than the property of semilinearity. It is intended to be an approximate characterization of the linguistic intuition that sentences of a natural language are built from a finite set of clauses of bounded structure using certain simple linear operations. The characterization of this intuition by the constant growth property is approximate because it refers to the growth of strings and not to the growth of structures.

It should be noted that these properties do not precisely define MCSG but rather give only a rough characterization, as the properties are only necessary conditions, and further some of the properties are properties of structural descriptions rather than the languages, hence, difficult to characterize precisely. This characterization of MCSG, obviously motivated by the formal properties of TAGs, would have remained only as a remark if it were not for some subsequent developments.

In response to a talk by Geoffrey Pullum at COLING 84, Joshi pointed out that all the known formal properties of HGs appeared to be exactly the same as those of TAGs. Later, in 1986, it was shown that with a slight modification of HGs (which is necessary as the wrapping operations are undefined for null strings), HGs are equivalent to TAGs. (Vijay-Shanker, Weir, and Joshi (1986) and Weir, Vijay-Shanker and Joshi (1986)).

Since then, two other formalisms were also shown to be equivalent to TAGs. These are Linear

Indexed Grammars (LIG), (Gazdar (1985)) and Combinatorial Categorical Grammars (CCG), (as developed by Steedman in some of his recent papers (1987, 1988)). Thus four quite different formalisms have been shown to be equivalent and thus belong to MCSG. These formalisms are different from each other in the sense that the formal objects and operations they employ are quite distinct and they are motivated by attempts to capture different aspects of language structure. Each of these formalisms have a domain of locality which is larger than that specifiable in a CFG. By a domain of locality we mean the elementary structures of a formalism over which dependencies such as agreement, subcategorization, filler-gap, etc. can be specified. However, it is not the case that each one of these formalisms extends the domain of locality to the same extent. TAGs extend the domain of locality far enough such that recursion is factored away from the domain of dependencies.

When two formalisms based on apparently completely different ideas turn out to be equivalent, there is a possibility that we are getting a handle on some fundamental properties of the objects that these formalisms were designed to describe. When more than two distinct formalisms turn out to be equivalent, the possibility is even greater. In fact, a deeper understanding of the relationships between these formalisms is obtained if we look at the derivation structures (related to structural descriptions) provided by each formalism. A first attempt to capture the closeness of some of these formalisms at the level of derivation structures resulted in the Linear Context-Free Rewriting Systems (LCFR) described in Section 6.

The plan for the rest of the paper is as follows. In Section 2, we have presented an introduction to TAGs including some simple examples. We have a little bit more detailed discussion of TAGs as the theory of TAGs has played a key role in our investigations of the relationships between different grammatical formalisms. In fact, most of the equivalences described in this paper have been established via TAGs. In this section, we have also described an extension of TAGs, called Multicomponent TAGs (MCTAG), first discussed in Joshi, Levy, and Takahashi (1975) and later precisely defined in Weir (1988). MCTAGs also belong to MCSGs and are in fact equivalent to LCFRs, discussed in Section 6.

In Section 3, we have briefly described Head Grammars (HG) and shown their equivalence to TAG. In Section 4, we shown the equivalence of Linear Indexed Grammars (LIG) to TAG, and in Section 5, the equivalence of Combinatory Categorical Grammars (CCG) and LIG and thereby, to

TAG and HG. In Section 6, we have presented Linear Context-Free Rewriting Systems (LCFR) the motivation of which was discussed earlier.

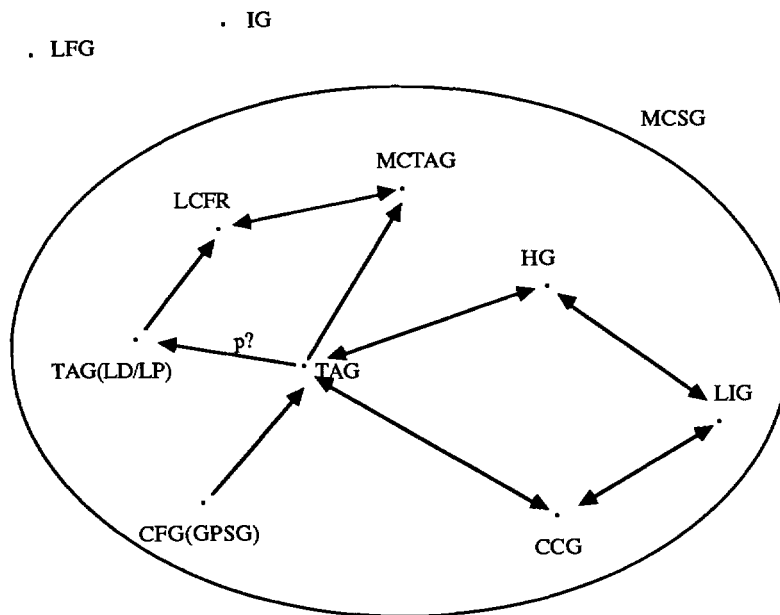
In Section 7, we have briefly presented Feature-Structure based TAGs (FTAGs), where adjunction becomes function application and unification. FTAGs, in general, are unconstrained; however, if the feature-structures associated with each node in an elementary tree are bounded, then this restricted FTAG (RFTAG) is equivalent to TAG. This restriction on feature-structures is similar to that in GPSG. However, since TAGs have an extended domain of locality, RFTAGs (equivalent to TAGs) are more powerful than GPSGs (equivalent to CFGs).

In Section 8, we consider a variant of TAGs, the lexicalized TAGs. Although adjoining can simulate substitution, by adding the operation of substitution explicitly, we obtain lexicalized TAGs. Such a lexicalization was implicit in TAGs in the sense that all the elementary trees need not be explicitly stated. However, the framework of lexicalized TAGs brings this out explicitly. Lexicalized TAGs are equivalent to TAGs. The relationship between TAGs and CCGs also becomes clear in the framework of lexicalized TAGs.

Finally, in Section 9, we consider a generalization of TAGs, called TAG (LD/LP), which decouples (local) domination for linear precedence, and allows a treatment of complex word-order patterns including those that exhibit long distance behavior similar to the filler-gap dependencies, but different in the sense that ‘movement’ is not to a grammatically defined position. The languages of TAGs are obviously contained in TAGs (LD/LP). It is not known yet whether the containment is proper.

Figure 1 summarizes all these relationships. The nodes are labelled by the grammar formalisms. The containments shown are, of course, with respect to the corresponding languages. Two formalisms outside of MCSG are also shown in the figure.

In this paper, we have presented our results informally, giving examples to illustrate the ideas in the proofs. In a subsequent paper, we will present more technical details and some detailed proofs.



- $A\bullet \longrightarrow \bullet B$ B properly contains A
- $A\bullet \xrightarrow{p?} \bullet B$ B contains A , proper containment not known
- $A\bullet \longleftrightarrow \bullet B$ $A = B$

[The containments are with respect to the languages].

CFG(GPSG): Context-free grammars (Generalized phrase structure grammars)

TAG: Tree adjoining grammars

HG: Head grammars

LIG: Linear indexed grammars

CCG: Combinatory categorial grammars

LCFR: Linear context-free rewriting systems

TAG(LD/LP): TAG (Local domination/Linear precedence)

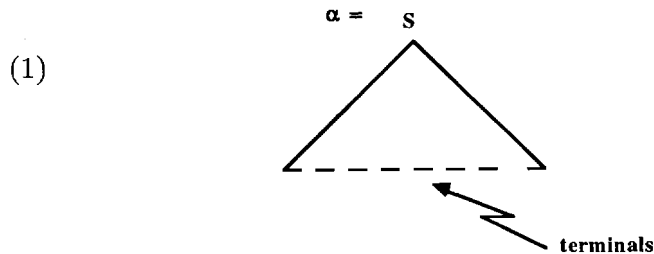
IG: Indexed grammars

LFG: Lexical functional grammars

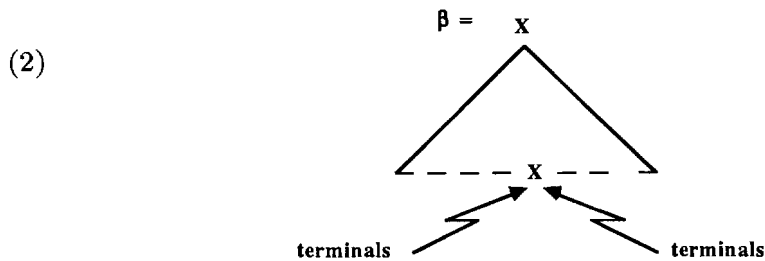
Figure 1: Mildly Context-Sensitive Grammar Formalisms (MCSG)

2 TREE ADJOINING GRAMMAR FORMALISM

A *tree adjoining grammar (TAG)* $G = (I, A)$ where I and A are finite sets of *elementary trees*. The trees in I will be called the *initial trees* and the trees in A , the *auxiliary trees*. A tree α is an initial tree if it is of the form in (1):



That is, the root node of α is labelled S and the frontier nodes are all non-terminals. A tree β is an auxiliary tree if it is of the form in (2):



That is, the root node of β is labelled X where X is a non-terminal and the frontier nodes are all terminals except one which is labelled X , the same label as that of the root. The node labelled X on the frontier will be called the *foot node* of β . The internal nodes are non-terminals. The initial and the auxiliary trees are not constrained in any manner other than as indicated above. The idea, however, is that both the initial and auxiliary trees will be *minimal* in some sense. An initial tree will correspond to a *minimal* sentential tree (i.e., without recursing on any non-terminal) and an auxiliary tree, with root and foot node labelled X , will correspond to a *minimal* recursive structure that must be brought into the derivation, if one recurses on X .

We will now define a composition operation called *adjoining* (or *adjunction*), which composes an auxiliary tree β with a tree γ . Let γ be a tree containing a node n bearing the label X and let β be an auxiliary tree whose root node is also labelled X . (Note that β must have, by definition, a node (and only one such) labelled X on the frontier.) Then the adjunction of β to γ at node n will be the tree γ' that results when the following operations are carried out: 1) The sub-tree of γ

dominated by n , call it t , is excised; 2) The auxiliary tree β is attached at n and its root node is identified with n ; 3) The sub-tree t is attached to the foot node of β and the root node n of t is identified with the foot node of β .

Figure 2 below illustrates this operation.

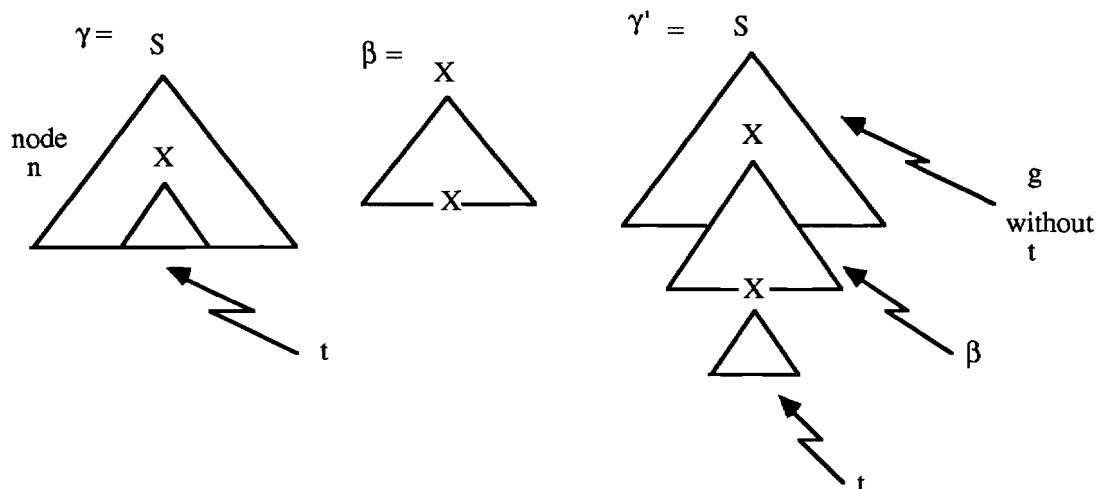


Figure 2: Adjunction

The intuition underlying the adjoining operation is a simple one but the operation is distinct from other operations on trees that have been discussed in the literature. In particular, we want to emphasize that adjoining is not a substitution operation in the usual sense.

Adjunction can, however, simulate substitution. A variant of TAG (called lexicalized TAG) which uses adjunction and also substitution explicitly is discussed in Section 8. Lexicalized TAGs are equivalent to TAGs. For the rest of the paper we will only consider adjunction as defined above.

The definition of adjunction allows more complex constraints to be placed on adjoining. Associated with each node is a *selective adjoining (SA)* constraint specifying the subset of auxiliary trees which can be adjoined at this node. Trees can only be included in the SA constraint associated with a particular node if their root and foot are labeled with the same nonterminal that labels the node. A mechanism is provided for ensuring that adjunction is performed at a node. This is done by associating an *obligatory adjoining (OA)* constraint with that node.

We should note that the SA and OA constraints are more than mere notational convenience,

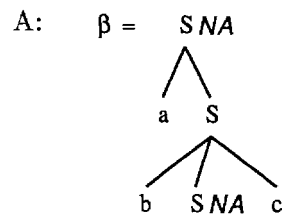
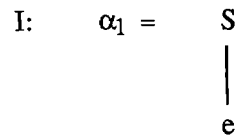
since they increase the generative power of the formalism. If the SA constraint specifies an empty subset of trees, then adjunction cannot be performed at this node—we call this constraint the *null adjoining (NA)* constraint.

(Later in Section 7, we will describe a Feature Structure Based TAG (FTAG). In this framework, adjoining becomes function application and unification, and the constraints described above are implicit in the feature-structures and the success or failure of unification during composition. A restricted version of FTAG, (RFTAG), described in Section 7, is equivalent to TAG. For the rest of the paper, we will consider the constraints as describe above).

Let us now define two auxiliary notions, the tree set of a TAG grammar and the string language of a TAG. Suppose $G=(I,A)$ is a TAG with a finite set of initial trees, a finite set of auxiliary trees, and the *adjoining* operation, as above. Then we define the *tree set* of a TAG G , $T(G)$, to be the set of all trees derived in G starting from initial trees in I . We further define the *string language* (or *language*) of G to be the set of all terminal strings of the trees in $T(G)$. The relationship between TAGs, context-free grammars, and the corresponding string languages can then be summarized in the following theorems Joshi, Levy, and Takahashi (1975), Joshi (1985): 1) For every context-free grammar, G' , there is a TAG, G , which is both weakly and strongly equivalent to G' . In other words, $L(G) = L(G')$ and $T(G) = T(G')$; 2) There exists a non-empty set of TAG grammars G_1 such that for every $G \in G_1$, $L(G)$ is context-free but there is no CFG G' such that $T(G') = T(G)$, i.e., TAGs are capable of providing structural descriptions for context-free languages that are not obtainable by a context-free grammar; 3) There exists a non-empty set of TAG grammars G_2 such that for every $G \in G_2$, $L(G)$ is strictly context sensitive; that is, there is no CFG grammar G' such that $L(G) = L(G')$, i.e., TAGs are strictly more powerful than CFGs; 4) There exist context-sensitive languages for which there are no equivalent TAGs, i.e., TAGs are properly contained in context-sensitive languages; 5) TAGs are semi-linear and hence have the constant growth property; 6) TAGs can capture only certain limited kinds of crossed dependencies. This follows from the nature of the automaton that corresponds to a TAG, called an embedded push-down automaton (EPDA), which is a generalization of the push-down automaton (PDA); 7) TAGs can be parsed in polynomial time, in fact, with a time based Kn^6 , where n is the length of the string and K is a constant depending on the grammar.

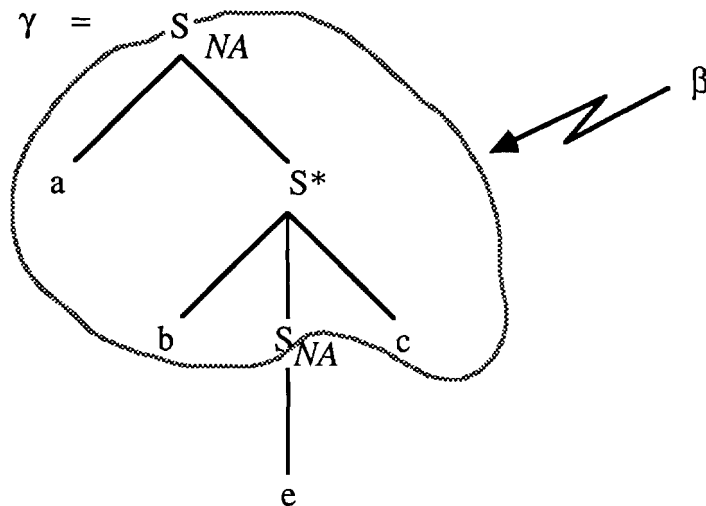
2.1 Some examples of formal languages

Example 2.1 Let $G = (I, A)$ be a TAG with local constraints where

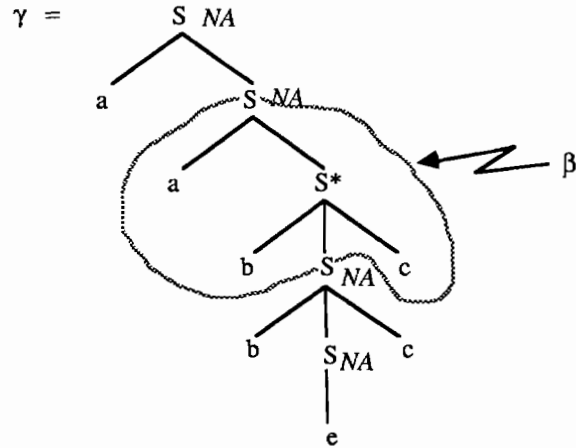


There are no constraints in α_1 . In β no auxiliary trees are adjoinable at the root node and the foot node and for the center S node there are no constraints.

Starting with α_1 and adjoining β to α_1 at the root node we obtain



Adjoining β to the center S node (the only node at which adjunction can occur) we have



It is easy to see that G generates the string language

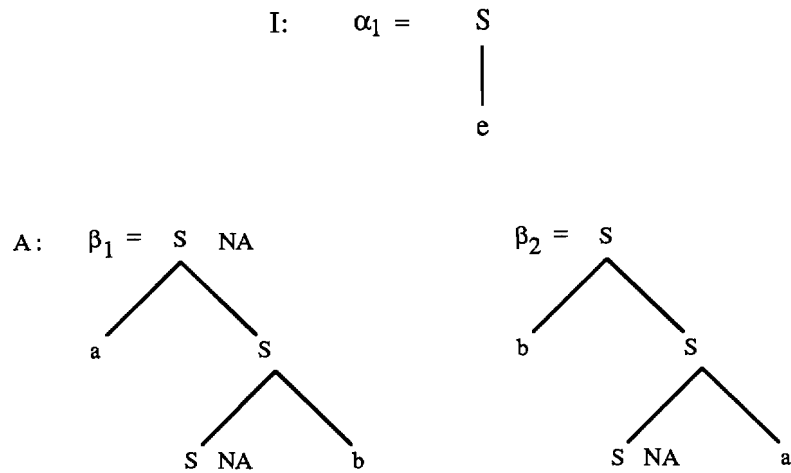
$$L = \{a^n b^n e c^n / n \geq 0\}$$

Example 2.2 Let G' be a TAG similar to G in Example 2.1, except that in G' there are no constants in β . G' generates

$$L = \{w e c^n / n \geq 0, \# a\text{'s in } w = \# b\text{'s in } w = n, \\ \text{and for any proper initial string } u \\ \text{of } w, \# a\text{'s in } u \geq \# b\text{'s in } u.\}$$

This language is closely related to the context-sensitive language discussed in Higginbotham (1984), which can also be shown to be a TAG language.

Example 2.3 Let $G = (I, A)$ be a TAG with local constraints where



G generates the language

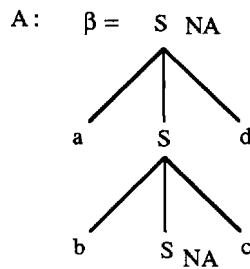
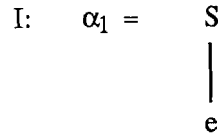
$$L = \{wew/w \in \{a, b\}^*\}$$

Example 2.4 Let G' be a TAG which is the same as G in Example 2.3 but without any local constraints. The corresponding language is

$$L = \{wew'/w, w' \in \{a, b\}^*, w = w' = 2n, \\ \# a\text{'s in } w = \# a\text{'s in } w' = \# b\text{'s} \\ \text{in } w = \# b\text{'s in } w' = n\}$$

This language is related to the Swiss-German example in Shieber (1984).

Example 2.5 Let $G = (I, A)$ be a TAG with local constraints where



G generates

$$L = \{a^n b^n e c^n d^n / n \geq 1\}$$

Note that it can be shown that languages

$$L^1 = \{a^n b^n c^n d^n e^n / n \geq 1\}$$

and

$$L^2 = \{www / w \in \{a, b\}^*\}$$

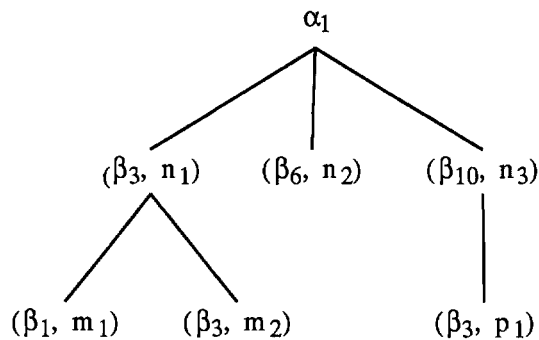
cannot be generated by TAGs either with or without local constraints (Joshi 1985). Other languages such as $L' = \{a^{n^2} / n \geq 1\}$ also cannot be generated by TAG. This is because the strings of a TAG have the constant growth property.

2.2 Derivation in a TAG

We will not describe formally the notion of derivation in a TAG, however we will give an informal discussion which will make the notion of derivation in TAG precise enough for our purpose. Adjoining is an operation defined on an elementary tree, say γ , an auxiliary tree, say β , and a node

(i.e., an address) in γ , say \mathbf{n} . Thus, every instance of adjunction is of the form “ β is adjoined to γ at \mathbf{n} ,” and this adjunction is always and only subject to the local constraints associated with \mathbf{n} . Although we very often speak of adjoining a tree to a node in a complex structure, we do so only for convenience. Strictly speaking, adjoining is always at a node in an elementary tree; and, therefore, it is more precise to talk about adjoining at an address in an elementary tree. More than one auxiliary tree can be adjoined to an elementary tree as long as each tree is adjoined at a distinct node. After these auxiliary trees are adjoined to the elementary tree, only nodes in the auxiliary trees are available for further adjunction. This precision in the definition of adjunction will be necessary when we define multicomponent adjunction in Section 2.3 below.

Now suppose that α is an initial tree and that β_1, β_2, \dots are auxiliary trees in a TAG, G . Then the derivation structure corresponding to the generation of a particular string in $L(G)$ might look as follows:



α_1 is an initial tree. β_3 , β_6 and β_{10} are adjoined at nodes n_1 , n_2 , and n_3 respectively in α_1 , where n_1 , n_2 , and n_3 are all distinct nodes. β_1 and β_3 are adjoined to β_3 at nodes m_1 and m_2 respectively. Again, m_1 and m_2 are distinct. β_6 has no further adjunctions but β_8 is adjoined to β_{10} at node p_1 . Note that the derivation structure D implicitly characterizes the 'surface' tree that is generated by it. (See Section 7 for the relationship of TAG and the unification formalism). In this way the derivation structure can be seen as the basic formal object constructed in the course of sentence. Associated with it will be two mappings, one to a surface syntactic tree and the other

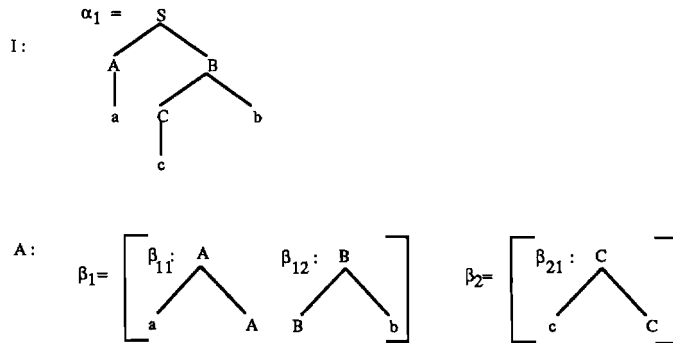
to a semantic interpretation, as below. (We are not concerned with semantic interpretation in this paper).

$$\text{surface tree} \longleftarrow \text{derivation structure} \longrightarrow \text{semantic interpretation}$$

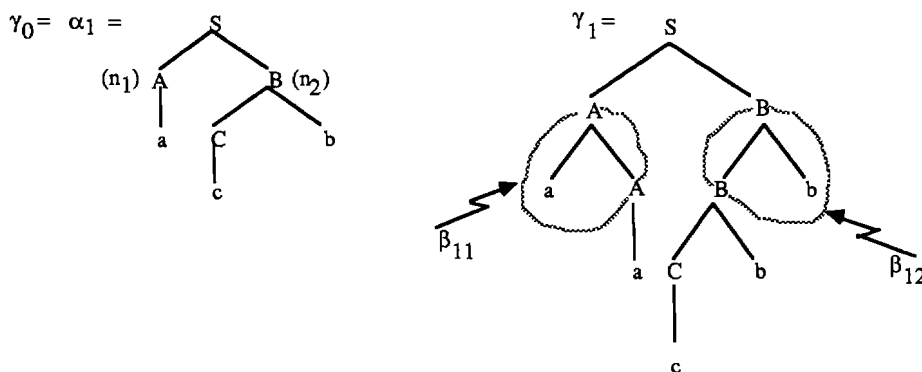
In a CFG, the derivation structure is the same as the surface structure. In a TAG this is not the case. Several of the formalisms that we have shown equivalent are comparable at the level of the derivation structures. Based on this observation in Section 6, we have discussed a framework called Linear Context-Free Rewriting System (LCFR) which captures the commonality at the level of derivation structures.

2.3 Multicomponent Tree Adjoining Grammars (MCTAG)

In Joshi, Levy, Takahashi (1975) a version of the adjoining operation is introduced under which, instead of a single auxiliary tree, a set of such trees is adjoined to a given elementary tree. We define the adjunction of such a set as the simultaneous adjunction of each of its component trees to a distinct node (address) in an elementary tree. This adjunction can, of course, take place only if the local constraints associated with each affected node of the elementary tree are satisfied. Consider, for example, the following grammar $G = (I, A)$:



β_1 is an auxiliary set consisting of the two trees β_{11} and β_{12} . Here is a sample derivation in G :

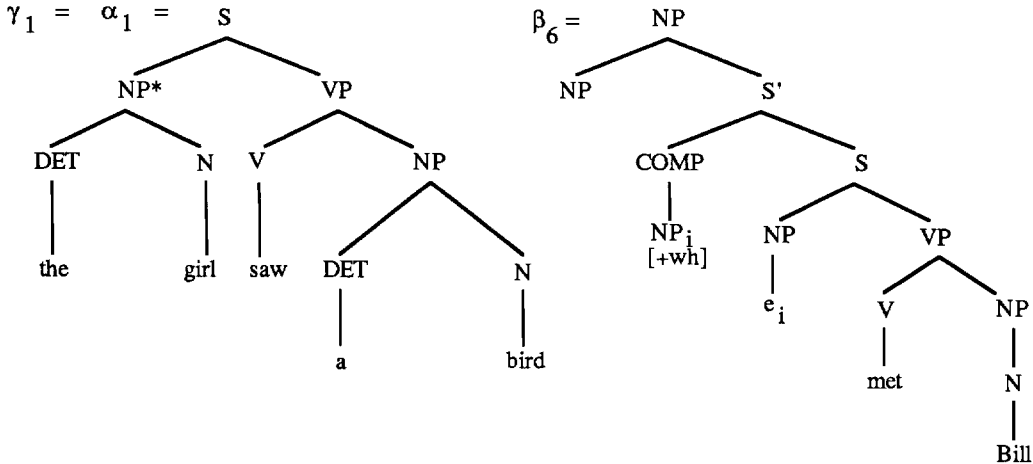


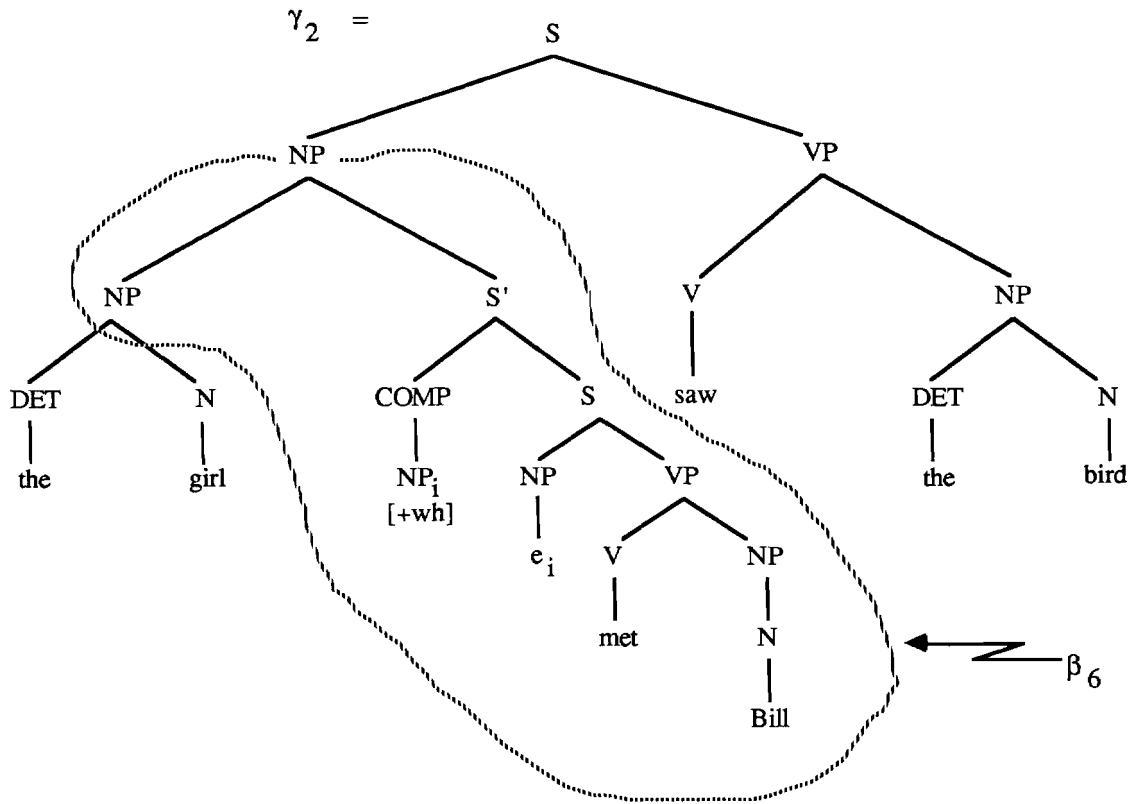
γ_1 above, it should be clear, is obtained by the adjunction of the components β_{11} and β_{12} of the auxiliary set β_1 to γ_0 at the nodes n_1 and n_2 respectively. In the current example, the set β_1 has two component trees and β_2 has only one component. If every auxiliary tree set of a TAG has only one component, we have a TAG as defined earlier. It can be shown that the number of components in the auxiliary sets does not make any difference to the generative capacity i.e., both the weak and strong (with respect to tree sets generated and not, of course, with respect to the derivation structures) generative capacities of multicomponent TAG are the same as that for TAG where each auxiliary set has exactly one component. On the other hand, derived auxiliary sets can be defined by adjoining an auxiliary set, say β_1 , to another auxiliary set, say β_2 , as follows. Each component of β_1 is adjoined to one (and exactly one) component of β_2 and all adjunctions are at distinct nodes. Note that since it is not required that each component of β_1 adjoins to the same component of β_2 , one component may adjoin to one component and another component to a different component of β_2 , i.e., adjunctions of components are not to the same component (elementary tree) of β_1 , but they are all adjunctions to the *same auxiliary set*. Thus, *locality* of adjoining can be defined in two ways: (1) by requiring that all components of an auxiliary set adjoin to the *same elementary tree*, (2) by requiring that all components of an auxiliary set adjoin to the *same auxiliary set*, not necessarily to the same elementary tree. The first type of locality does not add to the generative capacity of the MCTAG. The second type of locality does add to the weak generative capacity of the MCTAG; however, the resulting class of languages still falls within the class of “mildly context

sensitive” languages. With the second type of locality an MCTAG can be defined for the language $L' = \{a^n b^n | n \geq 1\}$ such that the a 's all hang from one path from the root node S and the b 's all hang from another path from the root node. Such a structural description cannot be provided by TAG where each auxiliary set has exactly one component (see also Joshi (1985)). For further details of MCTAGs, see Weir (1988). Weir (1988) has also shown that MCTAGs (with the second definition of locality) are equivalent to LCFRs (see Section 6).

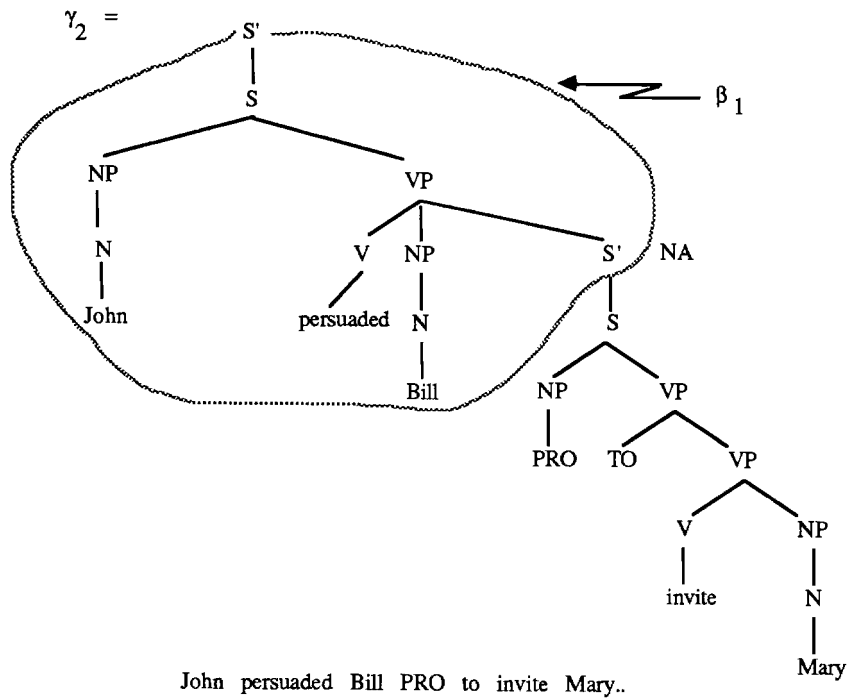
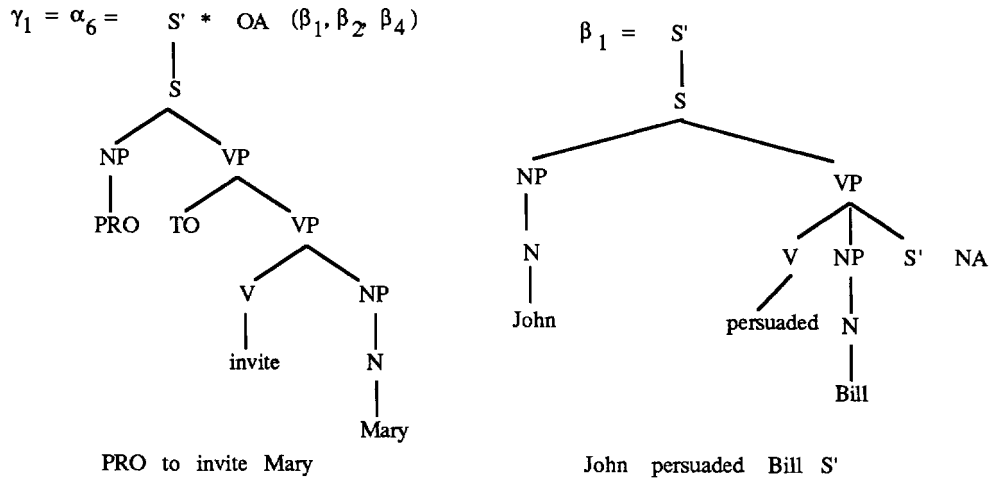
2.4 Some Linguistic Examples:

Example 2.6 Starting with the initial tree $\gamma_1 = \alpha_1$ and then adjoining β_6 at the indicated node (marked by *) in α_1 , we obtain γ_2 .



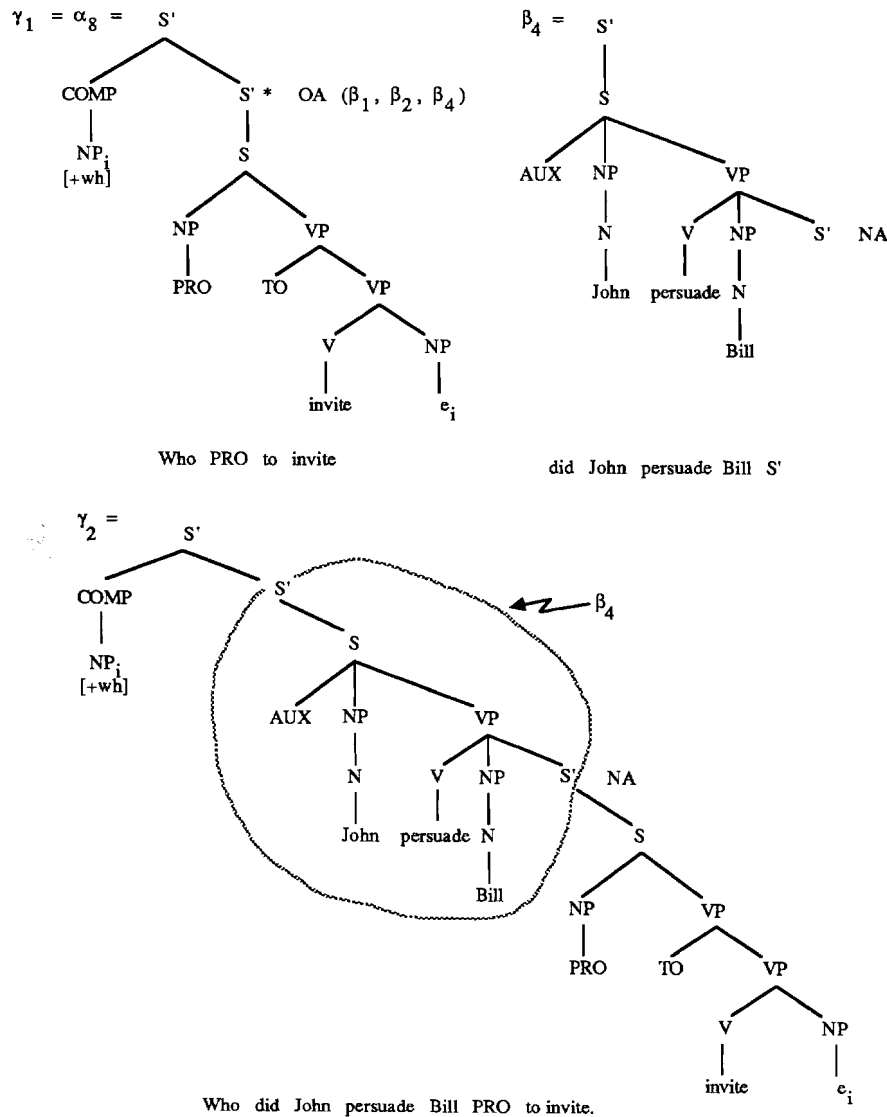


Example 2.7 Starting with the initial tree $\gamma_1 = \alpha_6$ and adjoining β_1 at the indicated node in α_6 we obtain γ_2 .



Since the initial tree α_6 is not a root sentence, it must undergo an adjunction at its root node, for example, by the auxiliary tree β_1 as shown above. Thus, for α_6 we have specified a local constraint $OA(\beta_1, \beta_2, \beta_4)$ for the root node, indicating that α_6 must undergo an adjunction at the root node by an auxiliary tree β_1 .

Example 2.8 Starting with the initial tree $\gamma_1 = \alpha_8$ and adjoining β_4 to α_8 at the indicated node in α_8 , we obtain γ_2 .



Note that the link in γ_1 is *preserved* in γ_2 ; it is *stretched*, resulting in a so-called unbounded dependency. Also note that, as in previous examples, α_8 is an initial tree that cannot serve as a root

sentence and the obligatory adjunction possibilities are as indicated. Again the local constraint (ϕ) at the foot node of β_4 prevents further adjoining at this node in γ_2 .

3 Head Grammars – Head wrapping and tree adjoining

In this section, we will briefly describe head grammars (HG) (Pollard 1984), and relate them to TAGs. For further details, see Weir, Vijay-Shanker, and Joshi, (1986); Vijay-Shanker, Weir, and Joshi, (1986); Joshi, Vijay-Shanker, and Weir, (1986). HGs are not to be confused with Head Driven Phrase Structure Grammars (HPSG) (Pollard 1985).

3.1 Head Grammars

Head Grammars are string rewriting systems like CFG's, but differ in that each string has a distinguished symbol corresponding to the head of the string. These are therefore called **headed strings**. The formalism allows not only concatenation of headed strings but also so-called **head wrapping** operations which split a string on one side of the head and place another string between the two sub-strings. When we wish to explicitly mention the head we use the notation $w_1\bar{a}w_2$; alternatively, we simply denote a headed string by \bar{w} . Productions in a HG are of the form $A \rightarrow f(\alpha_1, \dots, \alpha_n)$ or $A \rightarrow \alpha_1$ where: A is a nonterminal; α_i is either a nonterminal or a headed string; and f is either a concatenation or a head wrapping operation. Roach (1985) has shown that there is a normal form for Head Grammars which uses only the following operations.

$$LC1(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1\bar{a}_1u_2v_1a_2v_2$$

$$LC2(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1a_1u_2v_1\bar{a}_2v_2$$

$$LL1(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1\bar{a}_1v_1a_2v_2u_2$$

$$LL2(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1a_1v_1\bar{a}_2v_2u_2$$

$$LR1(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1v_1a_2v_2\bar{a}_1u_2$$

$$LR2(u_1\bar{a}_1u_2, v_1\bar{a}_2v_2) = u_1v_1\bar{a}_2v_2a_1u_2$$

$LC1$ concatenates the two strings, the head of the resulting string comes from the first string. Similarly $LC2$. $LL1$ inserts the second string into the first string to the right of the head of the

first string, i.e., the head of the first string is to the left. The head of the resultant string is the head of the first string. Similarly, for $LL2$, $LR1$, and $LR2$.

Pollard's definition of headed strings includes the headed empty string $\bar{\lambda}$. However the term $fi(\bar{w}_1, \dots, \bar{w}_i, \dots, \bar{w}_n)$ is undefined when $\bar{w}_i = \bar{\lambda}$. This nonuniformity has led to difficulties in proving certain formal properties of HGs (Roach (1985)). This difficulty can be removed by formulating HGs as follows.

Instead of headed strings, we will use so-called *split strings*. Unlike a headed string which has a distinguished symbol, a split string has a distinguished *position* about which it may be split. There are 3 operations on split strings: W , $C1$, and $C2$. The operations $C1$ and $C2$ correspond to the operations $LC1$ and $LC2$ in HGs. They are defined as follows:

$$C1(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 \uparrow w_2 u_1 u_2$$

$$C2(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 w_2 u_1 \uparrow u_2$$

Since the split point is not a symbol (which can be split either to its left or right) but a position between strings, separate left and right wrapping operations are not needed. The wrapping operation, W , is defined as follows:

$$W(w_1 \uparrow w_2, u_1 \uparrow u_2) = w_1 u_1 \uparrow u_2 w_2$$

It can be shown that this reformulation is equivalent to HG. We will use this reformulation in our further discussion.

3.2 Wrapping and Adjoining

The weak equivalence of HGs and TAGs is a consequence of the similarities between the operations of wrapping and adjoining. It is the roles played by the split point and the foot node that underlies this relationship. When a tree is used for adjunction, its foot node determines where the excised subtree is reinserted. The strings in the frontier to the left and right of the foot node appear on the left and right of the frontier of the excised subtree. As shown in the Figure 3 below, the foot node can be thought of as a position in the frontier of a tree, determining how the string in the frontier is split.

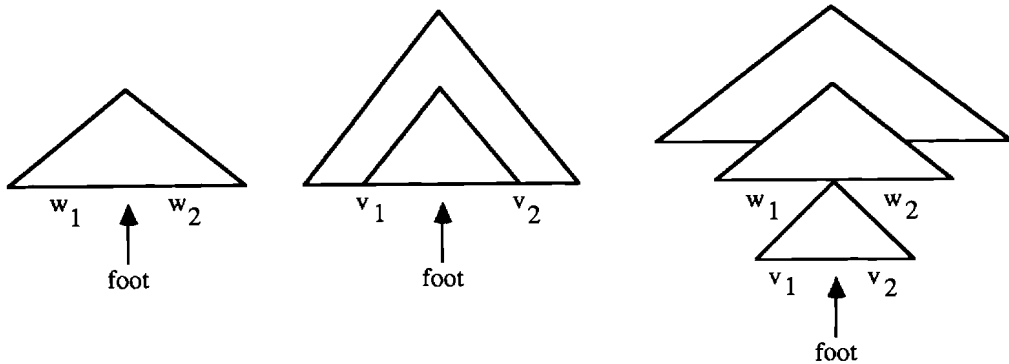


Figure 3: Wrapping and Adjoining

Adjoining in this case, corresponds to wrapping $w_1 \uparrow w$ around the split string $v_1 \uparrow v_2$. Thus, the split point and the foot node perform the same role. The proofs showing the equivalence of TAGs and HGs is based on this correspondence.

3.3 Inclusion of TAL in HL

We will briefly present a scheme for transforming a given TAG to an equivalent HG. We associate with each auxiliary tree a set of productions such that each tree generated from this elementary tree with frontier $w_1 X w_2$ has an associated derivation in the HG, using these productions, of the split string $w_1 \uparrow w_2$. The use of this tree for adjunction at some node labelled X can be mimicked with a single additional production which uses the wrapping operation.

For each elementary tree we return a sequence of productions capturing the structure of the tree in the following way. We use nonterminals that are named by the nodes of elementary trees rather than the labels of the nodes. For each node η in an elementary tree, we have two nonterminal X_η and Y_η allowing for the possibility that an adjunction occurs at η ; X_η derives the strings appearing on the frontier of trees derived from the subtree rooted at η ; Y_η derives the concatenation of the strings derived under each daughter of η . If η has daughters η_1, \dots, η_k then we have the production:

$$Y_\eta \rightarrow Ci(X_{\eta_1}, \dots, X_{\eta_k})$$

where the node η_i dominates the foot node (by convention, we let $i = 1$ if η does not dominate the

foot node). Adjunction at η , is simulated by use of the following production:

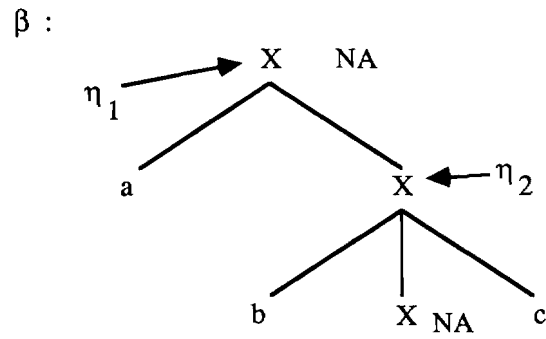
$$X_\eta \rightarrow W(X_\mu, Y_\eta)$$

where μ is the root of some auxiliary tree which can be adjoined at η . If adjunction is optional at η then we include the production:

$$X_\eta \rightarrow Y_\eta.$$

Notice that when η has an NA or OA constraint we omit the second or third of the above productions, respectively.

We illustrate the construction with an example showing a single auxiliary tree and the corresponding HG productions. In this example, $\mu_1, \mu_2, \dots, \mu_n$ are the root nodes of the trees that can be adjoined at η_2 in β .



HG productions corresponding to β :

$$X_{\eta_1} \rightarrow Y_{\eta_1},$$

$$Y_{\eta_1} \rightarrow C^2(a, X_{\eta_2}),$$

$$X_{\eta_2} \rightarrow W(X_{\mu_1}, Y_{\eta_2}),$$

$$X_{\eta_2} \rightarrow W(X_{\mu_n}, Y_{\eta_2}),$$

$$X_{\eta_2} \rightarrow Y_{\eta_2},$$

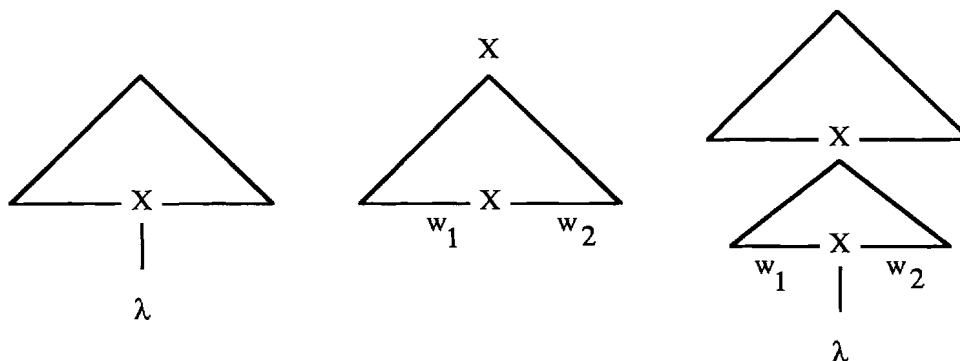
$$Y_{\eta_1} \rightarrow C2(b, X_{\eta_3}c),$$

$$X_{\eta_3} \rightarrow Y_{\eta_3},$$

$$Y_{\eta_3} \rightarrow \lambda$$

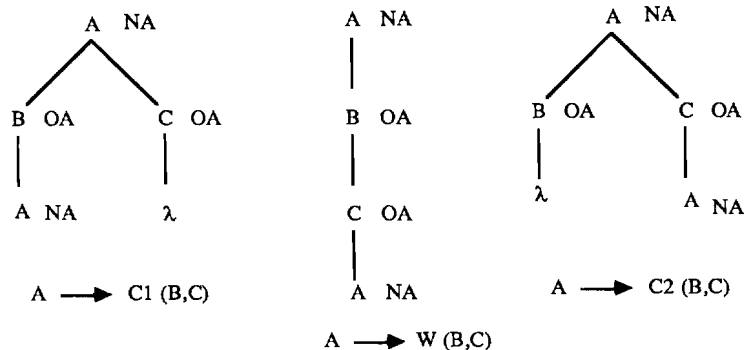
3.4 Inclusion of HL in TAL

In this construction, we use elementary trees to simulate directly the use of productions in HG to rewrite nonterminals. Generation of a derivation tree in string-rewriting systems involves the substitution of nonterminal nodes, appearing in the frontier of the unfinished derivation tree, by trees corresponding to productions for that nonterminal. From the point of view of the string languages obtained, tree adjunction can be used to simulate substitution, as illustrated in the following example.



Notice that although the node where adjoining occurs does not appear in the frontier of the tree, the presence of the node labelled by the empty string does not effect the string language.

For each production in the HG we have an auxiliary tree. A production in an HG can use one of the three operations: $C1$, $C2$, and W . Correspondingly we have three types of trees, shown below.



Drawing the analogy with string-rewriting systems: NA (null adjoining) constraints at each root have the effect of ensuring that a nonterminal is rewritten only once; NA constraints at the foot node ensures that, like the nodes labelled by λ , they do not contribute to the strings derived; OA (obligatory adjoining) constraints are used to ensure that every nonterminal introduced is rewritten at least once.

The two trees mimicking the concatenation operations differ only in the position of their foot node. This node is positioned in order to satisfy the following requirement: for every derivation in the HG there must be a derived tree in the TAG for the same string, in which the foot is positioned at the split point.

The tree associated with the wrapping operation is quite different. The foot node appears below the two nodes to be expanded because the wrapping operation of HGs corresponds to the $LL1$ operation of HGs in which the head (split point) of the second argument becomes the new head (split point). Placement of the nonterminal, which is to be wrapped, above the other nonterminal achieves the desired effect as described earlier.

While straightforward, this construction does not capture the linguistic motivation underlying TAGs. The auxiliary trees directly reflect the use of the concatenation and the wrapping operations. Elementary tree for natural languages are constrained to capture meaningful linguistic structures. In the TAGs generated in the above construction, the elementary trees are incomplete in this

respect: as reflected by the extensive use of the OA constraints. Since HGs do not explicitly give minimal linguistic structures in the sense of TAG, it is not surprising that such a direct mapping from HGs to TAGs does not recover this information.

3.5 Notational Differences between TAGs and HGs

TAGs and HGs are notationally very different, and this has a number of consequences that influence the way in which the formalisms can be used to express various aspects of language structure. The principal differences derive from the fact that TAGs are a tree-rewriting system unlike HGs which manipulate strings or pairs of strings.

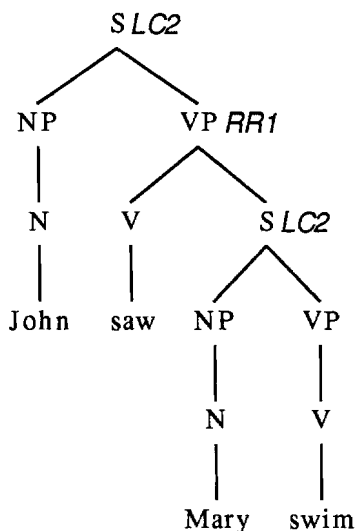
The elementary trees in a TAG, in order to be linguistically meaningful, must conform to certain constraints that are not explicitly specified in the definition of the formalism. In particular, each elementary tree must constitute a *minimal linguistic structure* elaborated up to preterminal (terminal) symbols and containing a head and all its complements or a modifier. Initial trees have essentially the structure of simple sentences; auxiliary trees correspond to minimal recursive constructions and generally constitute structures that act as modifiers of the category appearing at their root and foot nodes.

A hypothesis that underlies the linguistic intuitions of TAGs is that all dependencies are captured within elementary trees. This is based on the assumption that elementary trees are the appropriate domain upon which to define dependencies, rather than, for example, productions in a CFG. Since in string-rewriting systems, dependent lexical items can not always appear in the same production, the formalism does not prevent the possibility that it may be necessary to perform an unbounded amount of computation in order to check that two dependent lexical items agree in certain features. However, since in TAGs dependencies are captured by bounded structures, we expect that the complexity of this computation does not depend on the derivation. Features such as agreement may be checked within the elementary trees (instantiated up to lexical items) without need to percolate information up the derivation tree in an unbounded way. Some checking is necessary between an elementary tree and an auxiliary tree adjoined to it at some node, but this checking is still local and bounded. Similarly, elementary trees, being minimal linguistic structures, capture all of the sub-categorization information.

TAGs have only one operation of composition, namely, adjoining. HGs have concatenation

and wrapping a variety of ways. Further, TAGs are differentiated from HGs due to the fact that TAGs generate phrase-structure trees. As a result, the elementary trees must conform to certain constraints such as left-to-right ordering and dominance relations. Unlike other string-rewriting systems that use only the operation of concatenation, HGs do not associate a phrase-structure tree with a derivation: wrapping, unlike concatenation, does not preserve the word order of its arguments.

It is still possible to associate a phrase-structure with a derivation in HGs that indicates the constituents and we use this structure when comparing the analyses made by the two systems. These trees are not really phrase-structure trees but rather trees with annotations which indicate how the constituents will be wrapped (or concatenated). It is thus a derivation structure, recording the history of the derivation. With an example we now illustrate how a constituent analysis is produced by a derivation in a HG, corresponding to *John Mary saw swim* as required in a Dutch subordinating clause.



Although a TAG generates trees (phrase structure trees), these trees are not the derivation trees. For a tree γ generated in a TAG, G , there is a derivation structure associated with γ . Because TAG and HG are different systems, it is not possible to directly compare them with respect to their “strong” generative capacities. (In Weir, Vijay-Shanker and Joshi, (1986), a few linguistic examples have been discussed comparing the structural descriptions provided by HG and TAG). HG

and TAG are comparable at the level of the derivation structures they produce. This aspect will be discussed in Section 6.

4 Linear Indexed Grammars (LIG)

Indexed grammars (IG) were introduced by Aho (1968) as a generalization of CFG, and their mathematical properties have been investigated extensively. The class of ILs (indexed languages) is properly contained in the class of context-sensitive languages (CSL) and properly contains CFLs. IGs were not introduced as grammatical formalisms for natural language; however, since IGs are more powerful than CFGs and, as some recent investigations have shown, that some additional power beyond that of CFGs is required, IGs have received some attention from linguists. Gazdar (1985) has presented a discussion of the relevance of IGs to natural language. The class of ILs as a whole is clearly larger than the class of so-called mildly context-sensitive languages (MCSL), simply because ILs, in general, do not have the constant growth property, because not all ILs are semi-linear. For example, $L = \{a^{n^2} | n > 1\}$, $L = \{a^{2^n} | n \geq 1\}$ are ILs, but not semi-linear.

IGs are defined as follows [We will adopt the notation used in Gazdar (1985) which is essentially the same as in Hopcroft and Ullman (1979)]. Let A, B, C, \dots be the nonterminals; a, b, c, \dots the terminals; W, W_1, W_2, \dots strings of terminals and nonterminals; indices i, j, k, \dots ; stacks of indices $[]$, $[..]$, $[i..]$ where $[]$ denotes an empty sack, $[..]$ a possibly empty stack, and $[i..]$ a stack whose topmost index is i . The productions are as follows:

1. $A[..] \rightarrow W[..]$
2. $A[..] \rightarrow B[i..]$
3. $A[i..] \rightarrow W[..]$

$W[..]$ is a short hand for, for example, $A_1[...]\dots A_n[...]$ i.e., $W[..]$ stands for a righthand side in which each *nonterminal* in W is $[...]$ associated with it.

Rule 1 copies the stack on A to all the nonterminal daughters. It is assumed by convention that no stacks are associated with the terminals. Rule 2 pushes an index i on the stack passed from A to a unique nonterminal daughter. Rule 3 pops an index i and then copies stack on A to all the nonterminal daughters. Gazdar (1985) adds some additional rules, albeit redundantly. These are

4. $A[..] \rightarrow W_1[]B[.]W_2[]$
5. $A[..] \rightarrow W_1[]B[i.]W_2[]$

$$6. A[i..] \rightarrow W_1[]B[.]W_2[]$$

Rule 4 copies the stack on A to exactly one nonterminal daughter. Rule 5 and 6 push and pop an index on the stack of a designated daughter.

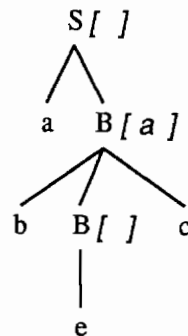
An IG which has only rules of the form 4, 5, and 6, will be called LIG, a linear indexed grammar. It can be shown that LIGs and TAGs, and therefore HGs, are weakly equivalent. (Vijay-Shanker 1987). We will illustrate briefly the relationship between LIG and TAG by means of an example.

Let G be a LIG as follows

1. $S[.] \rightarrow aA[a..]$
2. $S[.] \rightarrow aB[a..]$
3. $A[.] \rightarrow aA[a..]$
4. $A[.] \rightarrow aB[a..]$
5. $B[a..] \rightarrow bB[.]c$
6. $B[] \rightarrow e$

Let $w_1 = abec$. The derivation of w_1 is

(7)

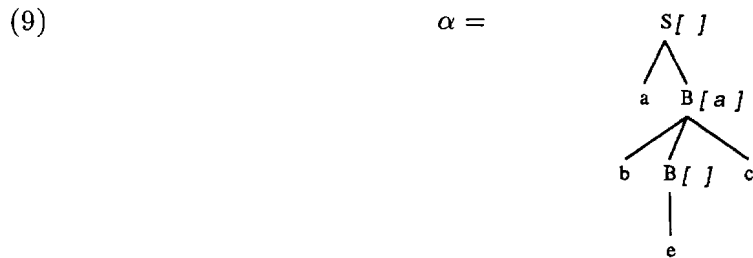


Let $w_2 = aabbcc$. The derivation of w_2 is

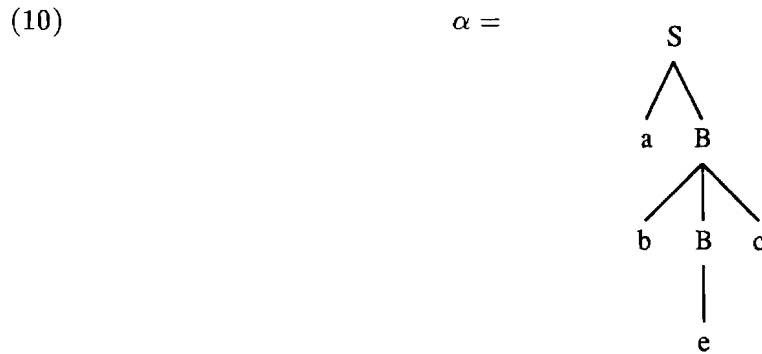


Comparing (7) and (8), one can see that $(n - 1)$ applications of (3), one applications of (4), then n applications (5) allows us to add n a 's, n b 's and n c 's in the right order.

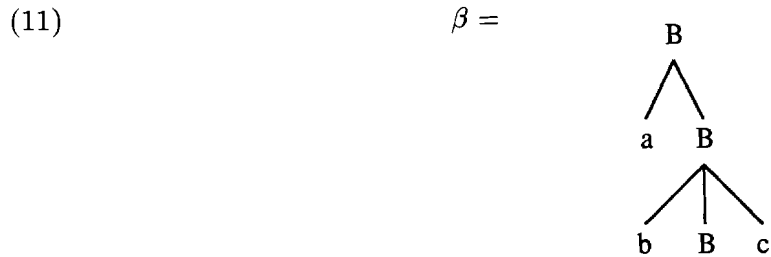
Note that (7) is a minimal derivation in G . We now take (7) to be an elementary tree (initial tree) say, α of a TAG G'



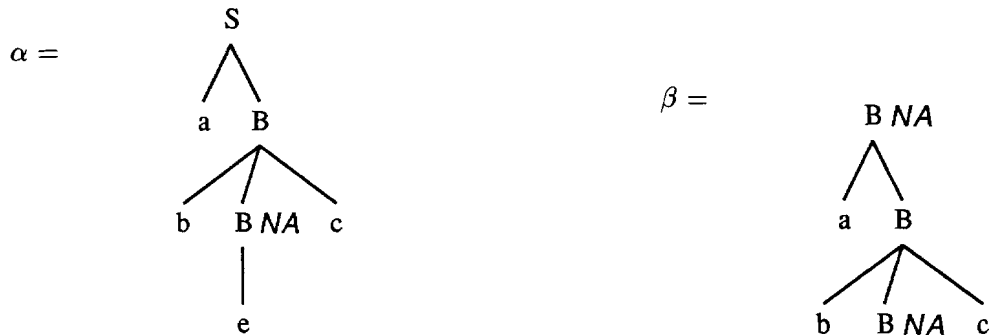
There is no need to keep the stacks around each one of the nonterminals appearing in specific addresses in α . Thus we can have



If we now introduce an auxiliary tree, say β



which introduces one a , one b , and one c , in the right order, we can simulate the effect of applying (3), (4), and (5) in LIG, assuming β is adjoined only to the node B which is the sibling of a . β should not be adjoined to the node B which dominates e in α . Similarly in β , since β can be adjoined to β itself, we want this adjoining to take place only for the interior B node β , and not to root and foot nodes of β . We can achieve this by placing the null adjoining constraint at the appropriate nodes. Hence, the equivalent TAG, G' is

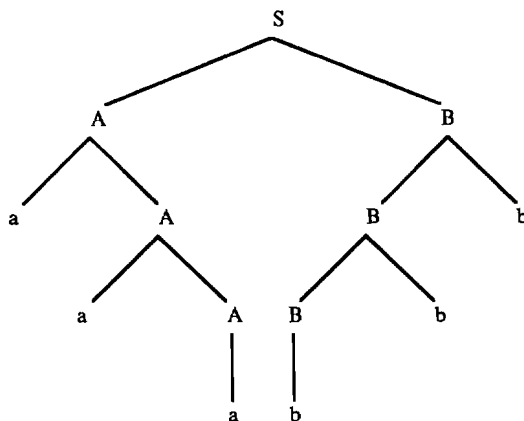


The operation of adjoining indirectly pairs the pushes and pops of indices in just the right way. Also, this information is implicitly communicated from the root of an auxiliary tree to the foot of the auxiliary along the spine of the auxiliary tree (i.e., the path from the root to the foot). This corresponds roughly to the LIG constraint that the stack passes on from the left hand side of a rule to only a designated daughter.

The equivalence of LIG and TAG thus explains a conjecture of Bill Marsh (referred to by Gazdar (1985)) that the languages $L_1 = \{a^n b^n c^n d^n e^n | n \geq 1\}$ and $L_2 = \{www | w \in \{0, 1\}^*\}$ cannot be generated by LIG.

MCTAG, multicomponent TAGs, as described in Section 2 are more powerful than TAGs both weakly and strongly. For example, MCTAGs can generate, for each $k \geq 1$, $L_3 = \{a_1^n, a_2^n, \dots, a_k^n | n \geq$

1} and can also generate the double spined structural descriptions for $L_A = \{a^n b^n | n \geq 1\}$ i.e., structural descriptions of the form



LIGs are inadequate for this purpose, because they do not permit branching stacks. MCTAGs are capable of achieving some of the effects permitted by branching stacks, however, still maintaining the constant growth property, thus they are more constrained. Whether there is any appropriate extension of LIG that permits branching stacks without leading to full power of IG is not known at present. Finding a subset of IG which is exactly or nearly equivalent to MCTAG is a challenging open problem.

Note that although the equivalence of LIG and TAG has been established, there are some key differences between them. If we consider the example described above, it is immediately clear that the ‘dependent’ a , b , and c are always in the same elementary tree. This is only implicit in LIG. The dependency can only be inferred by examining the state of the stacks at the various stages of the derivation. This is due to the fact that TAGs factor recursion and dependency, while a string rewriting system, of which LIG is an example, does not do so.

5 Combinatory Categorical Grammars

In this section, we examine Combinatory Categorical Grammars (CCGs), an extension of Classical Categorical Grammars (Ajdukiewicz (1935)) developed by Steedman and his collaborators (Ades and Steedman (1982), Steedman (1985, 87, 88)). Classical Categorical Grammars are known to be weakly equivalent to CFGs (Bar-Hillel, Gaifman, and Shamir (1964)), and the main result in this section is that under a certain definition, (which corresponds to Steedman’s recent work) CCGs are weakly equivalent to TAGs, HGs, and LIGs. We show this by showing that Combinatory Categorical Languages (CCLs) are included in Linear Indexed Languages (LILs), and that Tree Adjoining Languages (TALs) are included in CCLs (Weir and Joshi (1988)).

On the basis of their weak equivalence with TAGs, and HGs, it appears that CCGs should be classified as a mildly context-sensitive grammar formalism. The derivation tree sets traditionally associated with CCGs have context-free path sets, and are similar to those of LIGs, and therefore differ from those of LCFRSs. This does not, however, rule out the possibility that there may be alternative ways of representing the derivation of CCGs that will allow for their classification as LCFRSs.

The complexity of TAL recognition is $O(n^6)$. Thus, a corollary of our result is that this is also a property of CCLs. Although there has been previous work (Pareschi and Steedman (1987); Wittenburg (1986)) on the parsing of CCGs, they have not suggested a specific upper bound on recognition.

5.1 Definition of CCGs

Combinatory Categorical Grammar (CCG), as defined here, is the most recent version of a system that has evolved in a number of papers by Steedman (1985, 87, 88). In this section we first define CCGs, and then show that the class of string languages generated by CCGs is equal to the languages generated by TAGs (and LIGs).

Definition 5.1 A CCG, G , is denoted by (V_T, V_N, S, f, R) where

V_T is a finite set of terminals (lexical items),

V_N is a finite set of nonterminals (atomic categories),

S is a distinguished member of V_N ,

f is a function that maps elements of $V_T \cup \{\epsilon\}$ to finite subsets of $C(V_N)$, the set of categories¹, where

$$V_N \subseteq C(V_N) \text{ and if } c_1, c_2 \in C(V_N) \text{ then } (c_1/c_2) \in C(V_N) \text{ and } (c_1 \setminus c_2) \in C(V_N).$$

R is a finite set of combinatory rules.

There are four types of combinatory rules, which involve variables x, y, z over $C(V_N)$, and each $i \in \{\setminus, /$.

1. forward application:

$$(x/y) \quad y \rightarrow x$$

2. backward application:

$$y \quad (x \setminus y) \rightarrow x$$

3. generalized forward composition for some $n \geq 1$:

$$(x/y) \quad (\dots (y|_1 z_1)|_2 \dots |_n z_n) \rightarrow (\dots (x|_1 z_1)|_2 \dots |_n z_n)$$

4. generalized backward composition for some $n \geq 1$:

$$(\dots (y|_1 z_1)|_2 \dots |_n z_n) \quad (x \setminus y) \rightarrow (\dots (x|_1 z_1)|_2 \dots |_n z_n)$$

Restrictions can be associated with the use of the combinatory rule in R . These restrictions take the form of constraints on the instantiations of variables in the rules. These can be constrained in two ways.

1. The initial nonterminal of the category to which x is instantiated can be restricted.
2. The entire category to which y is instantiated can be restricted.

Derivations in a CCG involve the use of the combinatory rules in R . Let the *derives* relation be defined as follows.

$$\alpha c \beta \xrightarrow[G]{} \alpha c_1 c_2 \beta$$

if R contains a combinatory rule that has $c_1 c_2 \rightarrow c$ as an instance, and α and β are (possibly empty) strings of categories. The string languages, $L(G)$, generated by a CCG, G , is defined as follows.

$$\{a_1 \dots a_n \mid S \xrightarrow[G]{*} c_1 \dots c_n, c_i \in f(a_i), a_i \in V_T \cup \{\epsilon\}, 1 \leq i \leq n\}$$

Although there is no type-raising rule, its effect can be achieved to a limited extent since f can assign type-raised categories to lexical items. This is the scheme employed in Steedman's recent work.

5.2 Weak Generative Capacity

In this section we show that CCGs are weakly equivalent to TAGs, HGs, and LIGs. We do this by showing the inclusion of CCLs in LILs, and the inclusion of TALs in CCLs. We have already seen that TAG and LIG are equivalent (and TAG and HG are equivalent. (Weir, Vijay-Shanker and Joshi (1986). Thus, the two inclusions shown below imply the weak equivalence of all four systems (TAG, HG, LIG, and CCG).

5.2.1 CCLs \subseteq LILs

We describe how to construct a LIG, G' , from an arbitrary CCG, G such that G and G' are equivalent. Let us assume that categories are written without parentheses, unless they are needed to override the left associativity of the slashes.

A category c is **minimally parenthesized** if and only if one of the following holds.

$$c = A \text{ for } A \in V_N$$

$$c = (A|_1 c_1|_2 \dots |_n c_n), \text{ for } n \geq 1, \text{ where } A \in V_N \text{ and each } c_i \text{ is minimally parenthesized.}$$

It will be useful to be able to refer to the **components** of a category, c . We first define the immediate components of c .

when $c = A$ the immediate component is A ,

when $c = (A|_1 c_1|_2 \dots |_n c_n)$ the immediate components are A, c_1, \dots, c_n .

The components of a category c are its immediate components, as well as the components of its immediate components. The immediate components are the categories arguments. Thus, $c = (A|_1c_1|_2\dots|_nc_n)$ is a category that takes has n arguments of category c_1, \dots, c_n to give the **target** category A .

Although in CCGs there is no bound on the number of categories that are derivable during a derivation (categories resulting from the use of a combinatory rule), there is a bound on the number of *components* that derivable categories may have. This would no longer hold if unrestricted type-raising were allowed during a derivation.

Let the set $D_C(G)$ be defined as follows.

$$c \in D_C(G) \text{ if } c \text{ is a component of } c' \text{ where } c' \in f(a) \text{ for some } a \in V_T \cup \{\epsilon\}.$$

Clearly for any CCG, G , $D_C(G)$ is a finite set. $D_C(G)$ contains the set of all **derivable** components, i.e., for every category c that can appear in a sentential form of a derivation in some CCG, G , each component of c is in $D_C(G)$. This can be shown, since, for each combinatory rule, if it holds of the categories on the left of the rule then it will hold of the category on the right. The number of derivable *categories* is unbounded because they can have an unbounded number of immediate components.

Each of the combinatory rules in a CCG can be viewed as a statement about how a pair of categories can be combined. For the sake of this discussion, let us name the members of the pair according to their role in the rule.

The first of the pair in forward rules and the second of the pair in backward rules will be named the **primary** category. The second of the pair in forward rules and the first of the pair in backward rules will be named the **secondary** category.

As a result of the form that combinatory rules can take in a CCG, they have the following property. When a combinatory rule is used, there is a bound on the number of immediate components that the secondary categories of that rule may have. Thus, because immediate constituents must belong to $D_C(G)$ (a finite set), there is a bound on the number of categories that can fill the role of secondary categories in the use of a combinatory rule. Thus, there is a bound on the number of instantiations of the variables y and z ; in the combinatory rules in Section 5.1. The only variable that can be instantiated to an unbounded number of categories is x . Thus, by enumerating each

of the finite number of variable bindings for y and each z_i , the number of combinatory rules in R (while remaining finite) can be increased in such a way that only x is needed. Notice that x will appear only once on each side of the rules (i.e., they are linear).

We are now in a position to describe how to represent each of the combinatory rules by a production in the linear indexed grammars LIG, G' (see Section 4). In the combinatory rules, categories can be viewed as stacks since symbols need only be added and removed from the right. The secondary category of each rule will be a ground category: either A , or $(A|_1c_1|_2\dots|_nc_n)$, for some $n \geq 1$. These can be represented in a LIG as $A[]$ or $A[|_1c_1|_2\dots|_nc_n]$, respectively. The primary category in a combinatory rule will be unspecified except for the identity of its left and rightmost immediate components. If its leftmost component is a nonterminal, A , and its rightmost component is a member of $D_C(G)$, c , this can be represented in a LIG by $A[\cdot c]$.

In addition to mapping combinatory rules onto productions we must include productions in G' for the mappings from lexical items.

If $A \in f(a)$ where $a \in V_T \cup \{\epsilon\}$ then $A[] \rightarrow a \in P$

If $(A|_1c_1|_2\dots|_nc_n) \in f(a)$ where $a \in V_T \cup \{\epsilon\}$ then $A[|_1c_1|_2\dots|_nc_n] \rightarrow a \in P$

We now illustrate this construction by giving a LIG from a CCG that generates the language

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

Example 5.1 Let a CCG be defined as follows, where we have omitted unnecessary parenthesis.

$$\begin{aligned} A \in f(a) \quad B \in f(b) \quad C \in f(c) \quad D \in f(d) \\ S/S_1 \in f(\epsilon) \quad S_1 \in f(\epsilon) \quad S_1 \backslash A / D / S_1 \backslash B / C \in f(\epsilon) \end{aligned}$$

The following combinatory rules are permitted.

- Forward application involving x/y and y either when x begins with S_1 and y is C , or when x begins with S and y is S_1 or D .
- Backward application involving y and $x \backslash y$ when x begins with S and y is A or B .
- Forward composition involving x/y and $y \backslash z_1 / z_2 / z_3 \backslash z_4$ when x begins with S and y is S_1 .

The productions of the LIG that would be constructed from this CCG are as follows. The first 7 rules result from the definition of f . [In the notation for stacks below, the top of the stack appears to the right. In Section 4, in the notation for stacks, the top of the stack appears to the left. We have changed the notation from Section 4 for convenience because the “top” of a category in CCGs is on the right also.]

$$\begin{aligned}
A[] &\rightarrow a & B[] &\rightarrow b & C[] &\rightarrow c & D[] &\rightarrow d \\
S_1[] &\rightarrow \epsilon & S_1[\backslash A/D/S_1\backslash B/C] &\rightarrow \epsilon & S_1[\cdot] &\rightarrow S_1[\cdot / C] C[] \\
S[\cdot / S_1] &\rightarrow \epsilon & S[\cdot] &\rightarrow S[\cdot / S_1] S_1[] & S[\cdot] &\rightarrow S[\cdot / D] D[] \\
S[\cdot] &\rightarrow A[] S[\cdot \backslash A] & S[\cdot] &\rightarrow B[] S[\cdot \backslash B] \\
S[\cdot \backslash X_1/X_2/X_3\backslash X_4] &\rightarrow S[\cdot / S_1] S_1[\backslash X_1/X_2/X_3\backslash X_4]
\end{aligned}$$

for all $X_1, \dots, X_4 \in V_N$.

5.2.2 TALs \subseteq CCLs

We will just give the main idea of the construction of a CCG, G' from a TAG, G , such that G and G' are equivalent. It is important to appreciate that the order in which categories are combined is crucial in a CCG derivation, since the same categories combined in different orders give different strings.

Each of the auxiliary trees will result in certain assignments of categories by f to a terminal or the empty string. Each occurrence of adjunction will be mimicked by the use of a combinatory rule.

Adjunction into nodes to the right (left) of the foot node (which corresponds to concatenation) will be simulated by backward (forward) application. Adjunction into nodes dominating the foot node of a tree (which corresponds to wrapping) will be simulated in the CCG by composition. It is necessary that we ensure that the subsidiary category in every occurrence of composition has just been introduced into the derivation by an assignment of f (see Figure 4). This will correspond to the adjunction of an auxiliary tree that has not had any trees adjoined into it. It can be shown that composition is guaranteed only in this context.

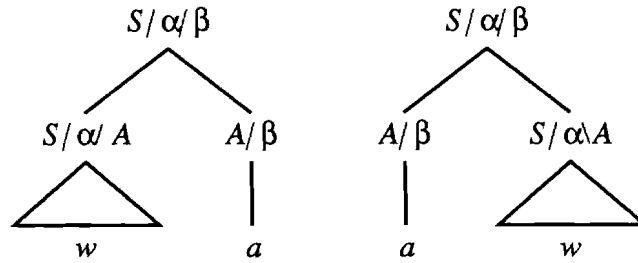


Figure 4: Context of composition

Forward and backward application are restricted to cases where the secondary category is some X^a , and the left immediate component of the primary category is some Y^a .

Forward and backward composition are restricted to cases where the secondary category has the form $((X^c|_1c_1)|_2c_2)$, or $(X^c|_1c_1)$, and the left immediate component of the primary category is some Y^a .

An effect of the restrictions on the use of combinatory rules is that only categories that can fill the secondary role during composition are categories assigned to terminals by f . Notice that the combinatory rules of G' depend only on the terminal and nonterminal alphabet of the TAG, and are independent of the elementary trees.

The construction depends on a particular normal form for TAGs. We will omit all the details here. The tree in Figure 5 is encoded by the category

$$A \setminus A_1^a / A_2^c / A_3^a \setminus A_4^a$$

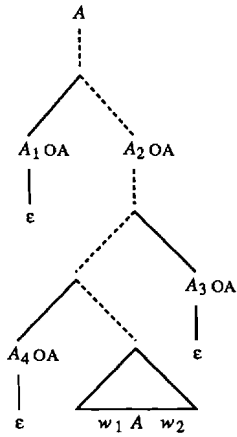


Figure 5: Tree encoding $A \setminus A_1^a / A_2^c / A_3^a \setminus A_4^a$

Example 5.2 Figure 6 shows an example of a TAG for the language $L_2 = \{a^n b^n \mid n \geq 0\}$ with crossing dependencies.

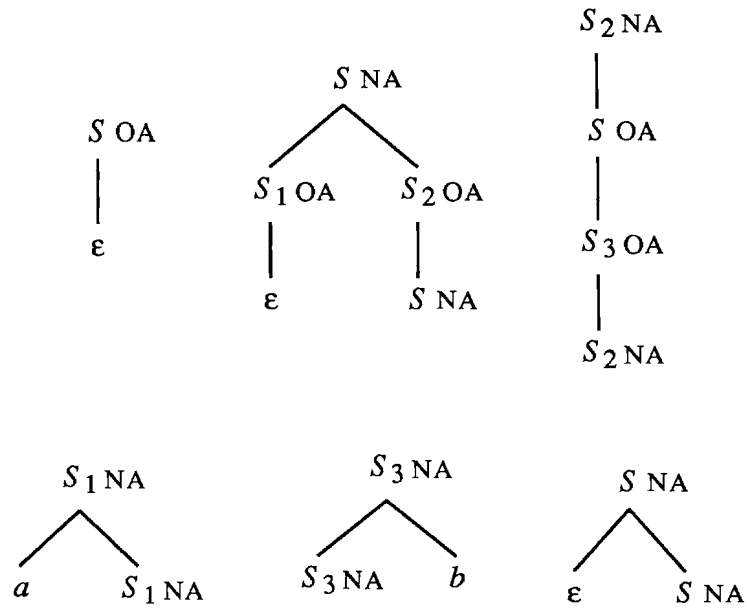


Figure 6: TAG for L_2

We give the CCG that would be produced according to this construction.

$$\begin{array}{ll}
S^a \setminus S_1^a / S_2^c \in f(\epsilon) & S^c \setminus S_1^a / S_2^c \in f(\epsilon) \\
S_2^a / S^c / S_3^c \in f(\epsilon) & S_2^c / S^c / S_3^c \in f(\epsilon) \\
S_1^a \setminus A \in f(\epsilon) & S_1^c \setminus A \in f(\epsilon) \\
S_3^a / B \in f(\epsilon) & S_3^c / B \in f(\epsilon) \\
A \in f(a) & B \in f(b) \\
S^a \setminus S_\epsilon \in f(\epsilon) & S^c \setminus S_\epsilon \in f(\epsilon) \\
S_\epsilon \in f(\epsilon) &
\end{array}$$

The CCGs produced according the construction given here have the property that parenthesis are redundant. It can be shown in general that the use of parenthesis in CCGs does not increase the generative power of the formalism. Parenthesis-free CGs differ from CCGs (Friedman, Dai, and Wang (1986), Friedman and Venkatesan (1986)). It can be shown that parenthesis-free CGs generate languages not generable by CCGs (Weir (1988)).

For further details of this construction and a discussion of derivation trees, see (Weir (1988)).

6 Linear Context-Free Rewriting Systems (LCFR)

So far we have discussed a number of formalisms and shown equivalences among them. All these systems share certain properties which make them members of the so-called “mildly context-sensitive” grammar formalisms. All these systems involve some type of context-free rewriting. Vijay-Shanker, Weir and Joshi (1987) and Weir (1988) have described a system, called linear context-free rewriting system (LCFR) which attempts to capture the common properties shared by these formalisms.

6.1 Generalized Context-Free Grammars

We define Generalized Context-Free Grammars (GCFGs), first discussed, though with a somewhat different notation, by Pollard (1984).

Definition 6.1 A GCFG G is written as $G = (V, S, F, P)$, where

V is a finite set of variables

S is a distinguished member of V

F is a finite set of function symbols

P is a finite set of productions of the form

$$A \rightarrow f(A_1, \dots, A_n)$$

where $n \geq 0$, $f \in F$, and $A, A_1, \dots, A_n \in V$.

The set of terms (trees), $T(G)$ derived from a GCFG, G is the set of all t such that $S \xrightarrow[G]{*} t$ where the *derives* relation is defined as follows.

- $A \xrightarrow[G]{*} f()$ if $A \rightarrow f()$ is a production.
- $A \xrightarrow[G]{*} f(t_1, \dots, t_n)$ if $A \rightarrow f(A_1, \dots, A_n)$ is a production, and $A_i \xrightarrow[G]{*} t_i$ for $1 \leq i \leq n$.

Notice that in GCFGs, rewriting choices during the derivation are independent of context. GCFG will generate a set of trees that can be interpreted as derivation tree in various grammar formalisms. Based on GCFG, we can now define a formalism called **Linear Context-Free Rewriting Systems**, (LCFR), which captures the common properties shared by the formalisms discussed earlier. By giving an interpretation for each of the functions in F , each term (tree) in $T(G)$ can be seen as encoding the derivation of some derived structure. It can be shown that context-free

grammars (CFG), head grammars (HG), tree adjoining grammars (TAG), and multicomponent tree adjoining grammars (MCTAG) are examples of LCFRs.²

In each of the LCFRSs that we have given, the functions (combining strings, trees, or sequences of strings and trees) share certain constrained properties. It is difficult to completely characterize the entire class of such functions that will be so constrained because we are considering formalisms with arbitrary structures. Instead, we will give two restrictions on the functions. We would like these restrictions to ensure that the functions do not “copy”, “erase”, or “restructure” unbounded components of their arguments. The result of composing any two structures should be a structure whose “size” is the sum of its constituents plus some constant. Every intermediate structure that a grammar derives contributes some terminals to the string that is yielded by the structure that is finally derived. However, the symbols in the yield of an intermediate structure do not necessarily form a continuous substring of the final string. In general, though, we can write the yield of an intermediate structure as a finite sequence of substrings of the final string. The composition operations are “size” preserving. Thus, with respect to the yield of the structures being manipulated, the composition operations do no more than reorder their arguments and insert a bounded number of additional terminals. It can be shown that LCFRs are semi-linear (and hence obey the constant growth property) and are parsable in polynomial time. For further details, see (Vijay-Shanker, Weir, and Joshi (1987) and Weir (1988)). Weir (1988) has also shown that languages generated by MCTAGs are equal to the languages generated by LCFRs.

Recently Kasami, Seki, and Fuji (1988) have studied a system called multiple context-free grammars, which is the same as LCFRs. They have obtained some additional properties of the classes of languages generated by their system, in particular, they have shown that the “non-erasing” property does not change the power.

7 Feature Structure Based TAG (FTAG) and Restricted FTAG (RFTAG)

7.1 Feature Structure Based Tree Adjoining Grammars (FTAG)

The linguistic theory underlying TAGs is centered around the factorization of recursion and localization of dependencies into the elementary trees. The “dependent” items usually belong to the same elementary tree³. 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 TAGs 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 generated⁴). 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.

In unification grammars, a feature structure is associated with a node in a derivation tree in order to describe that node and its relation 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 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 structures are associated with 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

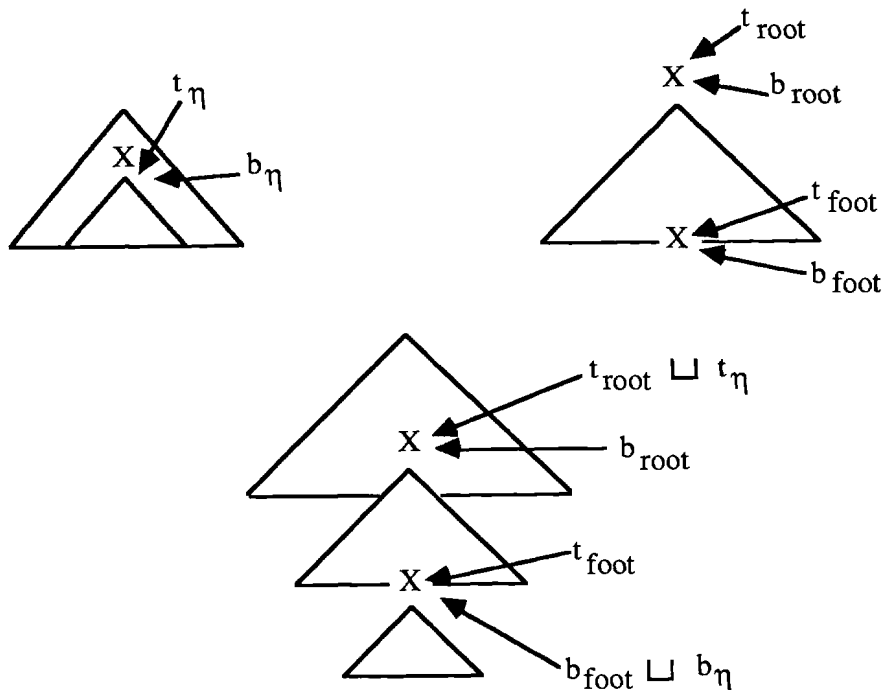


Figure 7: Feature structures and adjunction

auxiliary tree is consistent with the spirit of TAGs. 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_η and b_η place restrictions on the auxiliary tree that must be adjoined η . Note that if the node does not have OA constraint then we should expect t_η and b_η to be compatible (i.e., unifiable). For example, in the final sentential tree, this node will be viewed as a single entity.

Thus, in general, with every internal node, η , at which adjunction can take place we associate two structures, t_η and b_η . With each terminal node, we would associate only one structure⁵.

Let us now consider the case when adjoining takes place as shown in Figure 7. 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 . 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 (Vijay-Shanker (1987) and Joshi and Vijay-Shanker (1988)).

7.2 Restricted FTAG

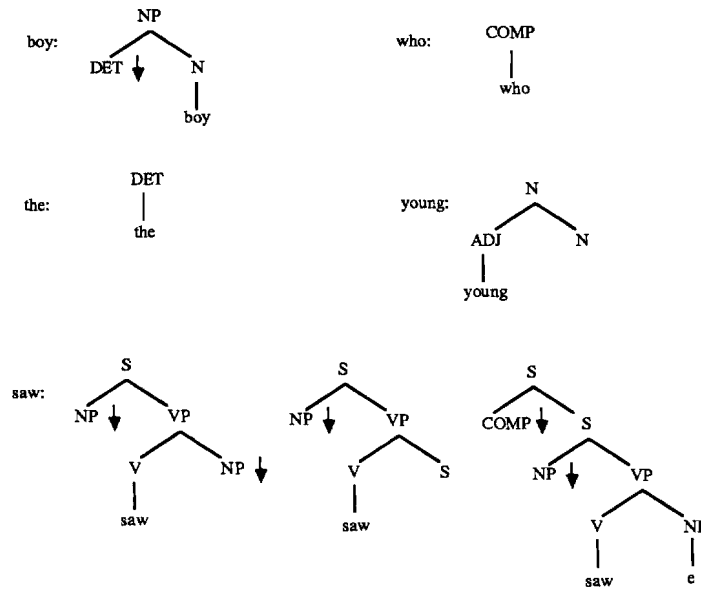
FTAGs as defined above are not constrained just as CFG-based unification grammars are not constrained. However, if we restrict the feature structures associated with each node of an elementary tree to be bounded then RFTAGs can be shown to be equivalent to TAGs (Vijay-Shanker (1987)). This restriction is the same as in GPSG; however, because of the larger domain of locality of TAGs and the operation of adjoining, RFTAGs are more powerful than GPSGs.

8 Lexicalized TAG

We call a grammar ‘lexicalized’ if it consists of (a) a finite set of structures associated with each lexical item, which is intended to be the ‘head’ of these structures, and (b) an operation or operations for composing these structures. The finite set of structures define the domain of locality over which constraints are specified, and these are local with respect to their ‘heads.’ It can be shown that, in general, a context-free grammar (CFG) cannot be lexicalized. Even if a CFG can be lexicalized it is not always the case that we can guarantee that the lexical item associated with a structure is the linguistically appropriate ‘head.’ Both these results hold even if the domain of locality is extended to trees. This is so because CFG has substitution as the only operation. If, however, we add adjoining as an operation, along with substitution, then we can appropriately lexicalize a CFG (see Abeille, Schabes, and Joshi (1988) for further details about lexicalized grammars, in particular, lexicalized TAGs).

TAGs are ‘naturally’ lexicalized because of their extended domain of locality and the operation of adjoining. In a lexicalized TAG we allow substitution in addition to adjunction. Adjoining can simulate substitution, however in a lexicalized TAG, we allow substitutions explicitly. The definitions of elementary trees are as before except that at the frontiers we can have nodes which are substitution nodes, in the sense that we have to substitute elementary or derived structures at the substitution nodes. Adjoining is defined as before. In the example below, the substitution nodes are marked by ↓.

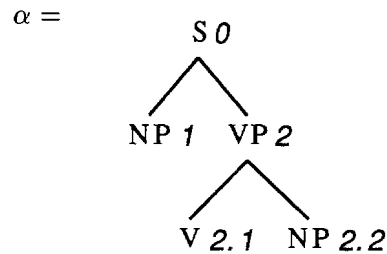
Example:



It can be shown that lexicalized TAGs are equivalent to TAGs. An Earley-type parser has been described in (Schabes and Joshi (1988)). For further details about lexicalized TAGs, see (Abeille, Schabes, and Joshi (1988)).

9 TAGs—Local Dominance and Linear Precedence: TAG(LD/LP)

The extended domain of locality of TAGs has implications for how domination and precedence can be factored. We will now take the elementary trees of TAG as *elementary structures* (initial and auxiliary) specifying only domination structures over which linear precedences can be defined. In fact, from now on we will define an *elementary structure* (ES) as consisting of the *domination structure plus a set of linear precedences*. Thus, α below is the domination structure of an ES.



The addresses for nodes serve to identify the nodes. They are not to be taken as defining the tree ordering. They are just labels for the nodes.

Let LP_1^α be a set of linear precedence statements associated with α

$$LP_1^\alpha = \left[\begin{array}{l} 1 < 2 \\ 2.1 < 2.2 \end{array} \right]$$

where $x < y$ (x precedes y) if x and y are nondominating nodes (i.e., x does not dominate y and y does not dominate x) and if x dominates z_1 and y dominates z_2 , then $z_1 < z_2$.

Note that LP_1^α corresponds exactly to the standard tree ordering. Given LP_1^α , (1) is the only terminal string that is possible with the ES (α, LP_1^α) , where α is the domination structure and LP_1^α is the linear precedence statement.

$$NP_1 V NP_2 \tag{1}$$

Suppose that instead of LP_1^α , we have

$$LP_2^\alpha = \left[\begin{array}{l} 1 < 2.1 \\ 2.1 < 2.2 \end{array} \right]$$

First note that in $1 < 2.1$, 2.1 is not a sister of 1. We can define precedences between nonsisters because the precedences are defined over α , the domain of locality.

Once again, the only terminal string that is possible with the ES (α, LP_2^α) is

$$NP_1 V NP_2 \tag{2}$$

but there is an important difference between (α, LP_1^α) and (α, LP_2^α) which will become clear when we examine what happens when an auxiliary tree is adjoined to α . Before we discuss this point, let us consider

$$LP_3^\alpha = \emptyset$$

i.e., there are no precedence constraints. In this case, we will get all six possible orderings

$$NP_1 V NP_2, NP_1 NP_2 V, V NP_2 NP_1, NP_2 V NP_1, V NP_1 NP_2, NP_2 NP_1 V$$

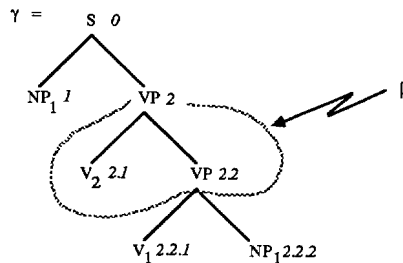
Let us return to (α, LP_1^α) and (α, LP_2^α) . As we have seen before, both ES give the same terminal string. Now let us consider an ES which is an auxiliary structure β (analogous to an auxiliary tree) with an associated LP, LP^β .

$$LP^\beta = [1 < 2]$$

When β is adjoined to α at the VP node in α . We have



When β is adjoined to α at the VP node in α , we have



We have put indices on NP and V for easy identification. NP_1, V_1, NP_2 belongs to α and V_2 belong to β . If we have LP_1^α associated with α and LP^β with β , after adjoining the LPs are updated in the obvious manner.

$$LP_1^\alpha = \left[\begin{array}{c} 1 < 2 \\ 2.2.1 < 2.2.2 \end{array} \right] \quad LP^\beta = [2.1 < 2.2]$$

The resulting LP for γ is

$$\begin{aligned} LP^\gamma &= LP_1^\alpha \cup LP^\beta \\ &= \left[\begin{array}{c} 1 < 2 \\ 2.1 < 2.2 \\ 2.2.1 < 2.2.2 \end{array} \right] \end{aligned}$$

Thus γ with LP^γ gives the terminal string

$$NP_1V_2V_1NP_2 \tag{3}$$

Instead of LP_1^α , if we associate LP_2^α with α then after adjoining β to α as before, the updated LPs are

$$LP_2^\alpha = \left[\begin{array}{c} 1 < 2.1 \\ 2.2.1 < 2.2.2 \end{array} \right] \quad LP^\beta = [2.1 < 2.2]$$

The resulting LP for γ is

$$\begin{aligned} LP^\gamma &= LP_2^\alpha \cup LP^\beta \\ &= \left[\begin{array}{c} 1 < 2.2.1 \\ 2.1 < 2.2 \\ 2.2.1 < 2.2.2 \end{array} \right] \end{aligned}$$

Thus (γ, LP^γ) gives the terminal strings

$$NP_1V_2V_1NP_2 \tag{4}$$

$$V_2NP_1V_1NP_2 \quad (5)$$

(4) is the same as (3), but in (5) NP_1 has ‘moved’ past V_2 . If we adjoin β once more to γ at the node VP at 2, then with LP_1^α associated with α , we will get

$$NP_1V_3V_2V_1NP_2 \quad (6)$$

and with LP_2^α associated with α , we will get

$$NP_1V_3V_2V_1NP_2 \quad (7)$$

$$V_2NP_1V_3V_1NP_2 \quad (8)$$

$$V_3V_2NP_1V_1NP_2 \quad (9)$$

Let us consider another LP for α , say LP_3^α

$$LP_3^\alpha = [1 < 2.1]$$

Then we have the following terminal strings for α (among others)

$$NP_1V_1NP_2 \quad (10)$$

$$NP_1NP_2V \quad (11)$$

It can be easily seen that given LP_3^α associated with α and LP^β associated β as before, after two adjoining with β_1 , we will get

$$NP_1V_3V_2V_1NP_2 \quad (12)$$

$$NP_1V_3V_2NP_2V_1 \quad (13)$$

$$NP_1V_3NP_2V_2V_1 \quad (14)$$

$$NP_1NP_2V_3V_2V_1 \quad (15)$$

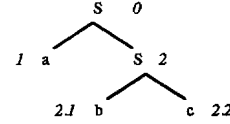
and, of course, several others. In (13), (14), and (15), NP_2 , the complement of V_1 in α has ‘moved’ past V_1 , V_2 , and V_3 respectively.

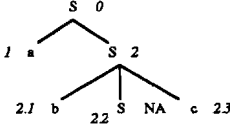
The elementary structures (ES) with their domination structure and the LP statements factor the constituency (domination) relationships from the linear order. The complex patterns arise due to the nature of the LP and the operation of adjoining. The main point here is that both the constituency relationships (including the filler-gap relationship) and the linear precedence relationship are defined on the elementary structures. *Adjoining preserves these relationships*. We have already seen in Section 2 how the constituency relationships are preserved by adjoining. Now we have seen how the linear precedence relationships are preserved by adjoining. Thus we have a uniform treatment of these two kinds of dependencies; however, the crucial difference between these two kinds of dependencies clearly shows up in our framework. In Joshi (1987), we have shown how TAG (LD/LP) can be used to several word order variations in Finnish, first described by Karttunen (1986).

The idea of factoring constituency (domination) relationships and linear order is basically similar to the ID/LP format of GPSG. However, there are important differences. First the domain of locality is the elementary structure (and not the rewrite rules or local trees), secondly we have defined the LP for each elementary structure. Of course, a compact description of LP over a set of elementary structures can be easily defined, but when it is compiled out it will be in the form we have stated it here. We will call a TAG in which the (local) domination relationships and linear precedence relationships are factored out, a TAG in the LD/LP representation, or a TAG(LD/LP).

In order to give further insight into the degree of word order variation possible in TAG(LD/LP), we will give an example.

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

I: $\alpha =$  $LP^\alpha = \begin{bmatrix} 1 < 2.1 \\ 2.1 < 2.2 \end{bmatrix}$

A: $\beta =$  $LP^\beta = \begin{bmatrix} 1 < 2.1 \\ 2.1 < 2.3 \end{bmatrix}$

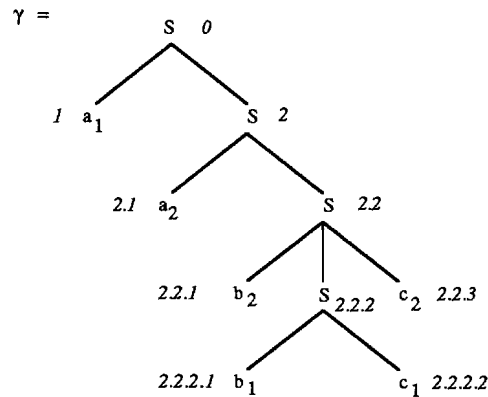
If G were a regular TAG, i.e., α and β were trees, then clearly the language generated by G , $L(G)$ is (as we have seen in Section 2)

$$L(G) = \{a^n b^n c^n | n \geq 1\}$$

However, G is a TAG(LD/LP). It is clear that the only string that corresponds to α is

abc

If we adjoin β to α at the node 2 in α , we get



[We have indexed a 's, b 's, and c 's for convenience.]

The updated LP^α and LP^β are

$$LP^\alpha = \left[\begin{array}{c} 1 < 2.2.2.1 \\ 2.2.2.1 < 2.2.2.2 \end{array} \right]$$

$$LP^\beta = \left[\begin{array}{c} 2.1 < 2.2.1 \\ 2.2.1 < 2.2.3 \end{array} \right]$$

Hence,

$$LP^\gamma = LP^\alpha \cup LP^\beta$$

$$= \left[\begin{array}{c} 1 < 2.2.2.1 \\ 2.2.2.1 < 2.2.2.2 \\ 2.1 < 2.2.1 \\ 2.2.1 < 2.2.3 \end{array} \right]$$

Thus some of the possible strings corresponding to γ are:

$$a_1 a_2 b_2 b_1 c_1 c_2 \tag{16}$$

$$a_1 a_2 b_2 b_1 c_2 c_1 \tag{17}$$

$$a_1 a_2 b_1 b_2 c_1 c_2 \tag{18}$$

$$a_1 a_2 b_2 b_1 c_2 c_1 \tag{19}$$

$$a_1 b_1 c_1 a_1 b_1 c_1 \tag{20}$$

$$a_2 b_2 c_2 a_1 b_1 c_1 \tag{21}$$

$$a_1 a_2 b_1 c_1 b_1 c_2 \tag{22}$$

There are 20 strings in all. Each string contains 2 a 's, 2 b 's, and 2 c 's. The corresponding a 's, b 's, and c 's appear in the order a, b, c . This example shows that the elements of an auxiliary structure can 'scatter' in a fairly complex manner over the elements of the elementary structure to which it is adjoined, the complexity arising out of the specifications for LP and the adjoining operation itself.

If in the TAG(LD/LP) above both LP^α and LP^β are empty i.e., there are no LP statements, then the language generated by G is the so-called MIX language (or Bach language), which consists of strings of equal number of a 's, b 's, and c 's in any order. MIX can be regarded as the extreme case of free word order. It is not known yet whether TAG, HG, CCG and LIG can generate MIX. This has turned out to be a very difficult problem. In fact, it is not even known whether an IG can generate MIX.

9.1 Language of TAG (LD/LP):

It is clear the languages of standard TAG (i.e., when the elementary structures are trees with the standard tree ordering) are contained in the class of languages of TAG(LD/LP). Whether the containment is proper or not is not known at present. Languages of TAG(LD/LP) continue to satisfy the constant growth property, hence cannot generate languages of the form $L = \{a^{n^2} | n \geq 1\}$, $L = \{a^{2^n} | n \geq 1\}$, etc.

Chapter Notes

1. Note that f can assign categories to the empty string, ϵ , though, to our knowledge, this feature has not been employed in the linguistic applications of CCG.
2. Weir (1988) has examined derivation trees associated with CCGs. The derivation trees traditionally associated with CCGs differ from those of LCFRs, this does not preclude the possibility that there may be an alternative way of representing derivations. Weir (1988) gives a normal form for CCGs which allows their being classified as LCFRs.
3. It is sometimes possible for “dependent” items to belong to an elementary tree and the immediate auxiliary tree that is adjoined in it.
4. Such a scheme would be an alternate way of embedding TAGs in an unificational framework. However, it does not capture the linguistic intuitions underlying TAGs, and loses the attractive feature of localizing dependencies.
5. 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.

References

- [1] A.E. Ades and M.J. Steedman. On the order of words. *Linguistics and Philosophy*, 3:517–558, 1982.
- [2] K. Ajdukiewicz. Die syntaktische konnexitat. *Studia Philosophica*, 1:1–27, 1935. English translation in: Polish logic 1920-1939, ed. by Storrs McCall, 207-231. Oxford University Press.
- [3] Y. Bar-Hillel, C. Gaifman, and E. Shamir. On categorial and phrase structure grammars. In *Language and Information*, Addison-Wesley, Reading, MA, 1964.
- [4] R. Berwick and A. Weinberg. *The Grammatical Basis of Linguistic Performance*. MIT Press, Cambridge, MA, 1984.
- [5] J.W. Bresnan, R.M. Kaplan, P.S. Peters, and A. Zaenen. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13:613–635, 1982.
- [6] C. Culy. The complexity of the vocabulary of bambara. *Linguistics and Philosophy*, 8:345–351, 1985.
- [7] J. Friedman, D. Dai, and W. Wang. The weak generative capacity of parenthesis-free categorial grammars. In 11th Intern. Conf. on Comput. Ling., 1986.
- [8] J. Friedman and R. Venkatesan. Categorial and Non-Categorial languages. In 24th meeting Assoc. Comput. Ling., 1986.
- [9] G. Gazdar. *Applicability of Indexed Grammars to Natural Languages*. Technical Report CSLI-85-34, Center for Study of Language and Information, 1985.
- [10] G. Gazdar. Phrase structure grammars. In P. Jacobson and G. Pullum, editors, *The Nature of Syntactic Recognition*, D. Reidel, Dordrecht, Holland, 1982.
- [11] G. Gazdar, E. Klein, G.K. Pullum, and I.A. Sag. *Generalized Phrase Structure Grammars*. Blackwell Publishing, Oxford, 1985. Also published by Harvard University Press, Cambridge, MA.
- [12] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

- [13] A.K. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions — Tree Adjoining Grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing — Theoretical, Computational and Psychological Perspective*, Cambridge University Press, New York, NY, 1985. Originally presented in May 1983 at the Workshop on Natural Language Parsing at the Ohio State University.
- [14] A.K. Joshi. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*, John Benjamins, Amsterdam, 1987.
- [15] A.K. Joshi. Word-order variation in natural language generation. In *Proceedings of the Annual Conference of the American Association for Artificial Intelligence (AAAI-87)*, Seattle, July 1987. (To appear also in *AI Journal*, 1988).
- [16] A.K. Joshi, L.S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal Computer Systems Science*, 10(1), 1975.
- [17] T. Kasami, H. Seki, and M. Fujii. *Generalized context-free grammars, multiple context-free grammars and head grammars*. Technical Report, Dept. Information and Computer Science, Osaka University, Osaka, Japan, 1988.
- [18] A.S. Kroch. Asymmetries in long distance extraction in a tag grammar. In M. Baltin and A. S. Kroch, editors, *New Conceptions of Phrase Structure*, University of Chicago, Press, Chicago, IL, 1986.
- [19] A.S. Kroch. Subjacency in a tree adjoining grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*, John Benjamins, Amsterdam, 1987.
- [20] A.S. Kroch and A.K. Joshi. Analyzing extraposition in a tree adjoining grammar. In G. Huck and A. Ojeda, editors, *Syntax and Semantics: Discontinuous Constituents*, Academic Press, New York, NY, 1986.
- [21] A.S. Kroch and A.K. Joshi. *Linguistic Relevance of Tree Adjoining Grammars*. Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1985. (To appear also in *Linguistics and Philosophy*).

- [22] A.S. Kroch and B. Santorini. The derived constituent structure of West Germanic verb-raising construction. In R. Freiden, editor, *Proceedings of the Princeton Workshop on Grammar*, MIT Press, 1988. To appear.
- [23] R. Pareschi and M.J. Steedman. A lazy way to chart-parse with categorial grammars. In 25th meeting Assoc. Comput. Ling., 1987.
- [24] C. Pollard. *Generalized Phrase Structure Grammars, Head Grammars and Natural Language*. PhD thesis, Stanford University, 1984.
- [25] C. Pollard. Lecture notes on head-driven phrase-structure grammar. 1985. Center for the Study of Language and Information, Stanford University, Stanford, CA.
- [26] G.K. Pullum. Free word order and phrase structure rules. In J. Pustejovsky and eds. P. Sells, editors, *Proceedings of NELS 12*, Amherst, MA, 1982.
- [27] K. Roach. Formal properties of head grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*, John Benjamins, Amsterdam, 1987.
- [28] W.C. Rounds. LFP: A logic for linguistic descriptions and an analysis of its complexity. To appear in *Comput. Ling.*
- [29] Y. Schabes, A. Abeille, and A.K. Joshi. Parsing strategies with ‘lexicalized’ grammars: applications to tree adjoining grammars. In 12th International Conference on Comput. Ling., 1988.
- [30] Y. Schabes and A.K. Joshi. An Earley-type parsing algorithm for tree adjoining grammars. In 26th meeting Assoc. Comput. Ling., 1988.
- [31] S.M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.
- [32] M. Steedman. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 1987.
- [33] M. J. Steedman. Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–568, 1985.

- [34] S. Steedman. Combinators and grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, Foris, Dordrecht, 1986.
- [35] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1987.
- [36] K. Vijay-Shanker and A.K. Joshi. Some computational properties of tree adjoining grammars. In *23rd meeting Assoc. Comput. Ling.*, pages 82–93, 1985.
- [37] K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *25th meeting Assoc. Comput. Ling.*, 1987.
- [38] K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. Tree adjoining and head wrapping. In *11th International Conference on Comput. Ling.*, 1986.
- [39] D.J. Weir and A.K. Joshi. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In *26th meeting Assoc. Comput. Ling.*, 1988.
- [40] D.J. Weir, K. Vijay-Shanker, and A.K. Joshi. The relationship between tree adjoining grammars and head grammars. In *24th meeting Assoc. Comput. Ling.*, 1986.
- [41] K.B. Wittenburg. Natural language parsing with combinatory categorial grammar in a graph-unification based formalism. 1986. D.Phil. Dissertation, University of Texas at Austin.