COMBINATORY CATEGORIAL GRAMMARS: GENERATIVE POWER AND RELATIONSHIP TO LINEAR CONTEXT-FREE REWRITING SYSTEMS*

David J. Weir Aravind K. Joshi

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Abstract

Recent results have established that there is a family of languages that is exactly the class of languages generated by three independently developed grammar formalisms: Tree Adjoining Grammars, Head Grammars, and Linear Indexed Grammars. In this paper we show that Combinatory Categorial Grammars also generates the same class of languages. We discuss the structural descriptions produced by Combinatory Categorial Grammars and compare them to those of grammar formalisms in the class of Linear Context-Free Rewriting Systems. We also discuss certain extensions of Combinatory Categorial Grammars and their effect on the weak generative capacity.

1 Introduction

There have been a number of results concerning the relationship between the weak generative capacity (family of string languages) associated with different grammar formalisms; for example, the theorem of Gaifman, et al. [3] that Classical Categorial Grammars are weakly equivalent to Context-Free Grammars (CFG's). More recently it has been found that there is a class of languages slightly larger than the class of Context-Free languages that is generated by several different formalisms. In particular, Tree Adjoining Grammars (TAG's) and Head Grammars (HG's) have been shown to be weakly equivalent [15], and these formalism are also equivalent to a restriction of Indexed Grammars considered by Gazdar [6] called Linear Indexed Grammars (LIG's) [13].

In this paper, we examine Combinatory Categorial Grammars (CCG's), an extension of Classical Categorial Grammars developed by Steedman and his collaborators [1,12,9,10,11]. The main result in this paper is that CCG's are weakly equivalent to TAG's, HG's, and LIG's. We prove this by showing in Section 3 that Combinatory Categorial Languages (CCL's) are included in Linear Indexed Languages (LIL's), and that Tree Adjoining Languages (TAL's) are included in CCL's.

After considering their weak generative capacity, we investigate the relationship between the structural descriptions produced by CCG's and those of other grammar formalisms. In [14] a number of grammar formalisms were compared and it was suggested that an important aspect of their descriptive capacity was reflected by the derivation structures that they produced. Several formalisms that had previously been described as mildly context-sensitive were found to share a number of properties. In particular, the derivations of a grammar could be represented with trees that always formed the tree set of a context-free grammar. Formalisms that share these properties were called Linear Context-Free Rewriting Systems (LCFRS's) [14].

On the basis of their weak generative capacity, it appears that CCG's should be classified as mildly context-sensitive. In Section 4 we consider whether CCG's should be included in the class of LCFRS's. The derivation tree sets traditionally associated with CCG's have Context-free path sets, and are similar to those of LIG's, and therefore differ from those of LCFRS's. This does not, however, rule out the possibility that there may be alternative ways of representing the derivation of CCG's that will allow for their classification as LCFRS's.

Extensions to CCG's have been considered that enable them to compare two unbounded structures (for example, in [12]). It has been argued that this may be needed in the analysis of certain coordination phenomena in Dutch. In Section 5 we discuss how these additional features increase the power of the formalism. In so doing, we also give an example demonstrating that the Parenthesis-free Categorial Grammar formalism [5,4] is more powerful that CCG's as defined here. Extensions to TAG's (Multicomponent TAG) have been considered for similar

*This work was partially supported by NSF grants MCS-82-19116-CER, MCS-82-07294, DCR-84-10413, ARO grant DAA29-84-9-0027, and DARPA grant N0014-85-K0018. We are very grateful to Mark Steedman, K. Vijay-Shanker and Remo Pareschi for helpful discussions.

reasons. However, in this paper, we will not investigate the relationship between the extension of CCG's and Multicomponent TAG.

2 Description of Formalisms

In this section we describe Combinatory Categorial Grammars, Tree Adjoining Grammars, and Linear Indexed Grammars.

2.1 Combinatory Categorial Grammars

Combinatory Categorial Grammar (CCG), as defined here, is the most recent version of a system that has evolved in a number of papers [1,12,9,10,11].

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 categories[1], where
$V_N \subseteq C(V_N)$ and
if $c_1, c_2 \in C(V_N)$ then
$(c_1/c_2) \in C(V_N)$ and $(c_1\backslash c_2) \in C(V_N)$.
R is a finite set of combinatory rules, described below.

We now give the combinatory rules, where x, y, z are variables over categories, and each $|_i$ denotes either $\backslash$ or $/$.

1. forward application:

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

2. backward application:

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

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

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

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

$$(\ldots(y|_1 z_1)|_2 \ldots|_n z_n) \quad (x\backslash y) \rightarrow (\ldots(x|_1 z_1)|_2 \ldots|_n z_n)$$

[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.

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.

$$\alpha c \beta \underset{G}{\Rightarrow} \alpha c_1 c_2 \beta$$

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.

$$\{a_1 \ldots a_n \mid S \underset{G}{\overset{*}{\Rightarrow}} c_1 \ldots c_n, \; c_i \in f(a_i), a_i \in V_T \cup \{\epsilon\}, 1 \leq i \leq n\}$$

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, which is the scheme employed in Steedman's recent work.

2.2 Linear Indexed Grammars

Linear Indexed Grammars (LIG's) were introduced by Gazdar [6], and are a restriction of Indexed Grammars introduced by Aho [2]. LIG's can be seen as an extension of CFG's in which each nonterminal is associated with a stack.

An LIG, G, is denoted by $G = (V_N, V_T, V_S, S, P)$ where
V_N is a finite set of nonterminals,
V_T is a finite set of terminals,
V_S is a finite set of stack symbols,
$S \in V_N$ is the start symbol, and
P is a finite set of productions, having the form

$$A[] \rightarrow a$$

$$A[\cdot\, l] \rightarrow A_1[] \ldots A_i[\cdot] \ldots A_n[]$$

$$A[\cdot] \rightarrow A_1[] \ldots A_i[\cdot\, l] \ldots A_n[]$$

where $A_1, \ldots A_n \in V_N$, $l \in V_S$, and $a \in V_T \cup \{\epsilon\}$.

The notation for stacks uses $[\cdot\, l]$ to denote an arbitrary stack whose top symbol is l. This system is called *Linear* Indexed Grammars because it can be viewed as a

restriction of Indexed Grammars in which only one of the non-terminals on the right-hand-side of a production can inherit the stack from the left-hand-side.

The derives relation is defined as follows.

$$\alpha A[l_m \dots l_1 l]\beta \underset{G}{\Rightarrow} \alpha A_1[] \dots A_i[l_m \dots l_1] \dots A_n[]\beta$$

$$\text{if} \quad A[\cdot \cdot l] \rightarrow A_1[] \dots A_i[\cdot\cdot] \dots A_n[] \in P$$

$$\alpha A[l_m \dots l_1]\beta \underset{G}{\Rightarrow} \alpha A_1[] \dots A_i[l_m \dots l_1 l] \dots A_n[]\beta$$

$$\text{if} \quad A[\cdot\cdot] \rightarrow A_1[] \dots A_i[\cdot \cdot l] \dots A_n[] \in P$$

$$\alpha A[]\beta \underset{G}{\Rightarrow} \alpha a \beta$$

$$\text{if} \quad A[] \rightarrow a \in P$$

The language, $L(G)$, generated by G is

$$\{ w \mid w \in V_T^*, S[] \overset{*}{\underset{G}{\Rightarrow}} w \}$$

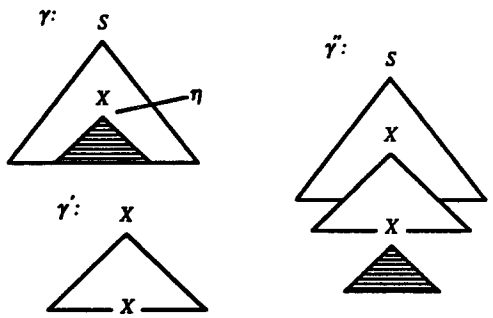
2.3 Tree Adjoining Grammars

A TAG [8,7] is denoted $G = (V_N, V_T, S, I, A)$ where
V_N is a finite set of nonterminals,
V_T is a finite set of terminals,
S is a distinguished nonterminal,
I is a finite set of initial trees and
A is a finite set of auxiliary trees.

Initial trees are rooted in S with $w \in V_T^*$ on their frontier. Each internal node is labeled by a member of V_N.

Auxiliary trees have $w_1 A w_2 \in V_T^* V_N V_T^*$ on their frontier. The node on the frontier labeled A is called the *foot* node, and the root is also labeled A. Each internal node is labeled by a member of V_N.

Trees are composed by **tree adjunction.** When a tree γ' is adjoined at a node η in a tree γ the tree that results, γ'', is obtained by excising the subtree under η from γ and inserting γ' in its place. The excised subtree is then substituted for the foot node of γ'. This operation is illustrated in the following figure.



Each node in an auxiliary tree labeled by a nonterminal is associated with adjoining constraints. These constraints specify a set of auxiliary trees that can be adjoined at that node, and may specify that the node has obligatory adjunction (OA). When no tree can be adjoined at a node that node has a null adjoining (NA) constraint.

The string language $L(G)$ generated by a TAG, G, is the set of all strings lying on the frontier of some tree that can be derived from an initial trees with a finite number of adjunctions, where that tree has no OA constraints.

3 Weak Generative Capacity

In this section we show that CCG's are weakly equivalent to TAG's, HG's, and LIG's. We do this by showing the inclusion of CCL's in LIL's, and the inclusion of TAL's in CCL's. It is know that TAG and LIG are equivalent [13], and that TAG and HG are equivalent [15]. Thus, the two inclusions shown here imply the weak equivalence of all four systems. We have not included complete details of the proofs which can be found in [16].

3.1 CCL's $\subseteq$ LIL's

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.
>
> $c = A$ for $A \in V_N$
> $c = (c_0|_1 c_1|_2 \dots |_n c_n)$, for $n \geq 1$, where $c_0 \in V_N$ and each c_i 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 $c = (c_0|_1 c_1|_2 \ldots |_n c_n)$ the immediate components are $c_0, c_1, \ldots, c_n$.

The components of a category c are its immediate components, as well as the components of its immediate components.

Although in CCG's 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.

$c \in D_C(G)$ if c is a component of c' where $c' \in f(a)$ 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 because 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.

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_i in the combinatory rules in Section 2.1. 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 can be increased in such a way that only x is needed. Notice that x will appears 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 LIG, G'. 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 $(A|_1 c_1|_2 \ldots |_n c_n)$, for some $n \geq 1$. These can be represented in a LIG as $A[]$ or $A[]_1 c_1|_2 \ldots |_n c_n]$, respectively. The primary category in a combinatory rule will be unspecified except for the identity of its left and rightmost immediate components. 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 $A[\cdot \cdot c]$.

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

If $c \in f(a)$ where $a \in V_T \cup \{\epsilon\}$ then

$$\text{if } \quad c = A \quad \text{then} \quad A[] \rightarrow a \in P$$

$$\text{if } \quad c = (A|_1 c_1|_2 \ldots |_n c_n) \quad \text{then} \quad A[]_1 c_1|_2 \ldots |_n c_n] \rightarrow a \in P$$

We are assuming an extension of the notation for productions that is given in Section 2.2. Rather than adding or removing a single symbol from the stack, a fixed number of symbols can be removed and added in one production. Furthermore, any of the nonterminals on the right of productions can be given stacks of some fixed size.

3.2 TAL's $\subseteq$ CCL's

We briefly describe the construction of a CCG, G' from a TAG, G, such that G and G' are equivalent.

For each nonterminal, A of G there will be two nonterminals A^a and A^c in G'. The nonterminal of G' will also include a nonterminal A_i for each terminal a_i of the TAG. The terminal alphabets will be the same. The combinatory rules of G' are as follows.

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 $((X^c|_1 c_1)|_2 c_2)$, 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.

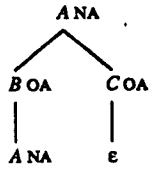
f is defined on the basis of the auxiliary trees in G. Without loss of generality we assume that the TAG, G, has trees of the following form.

I contains one initial tree:

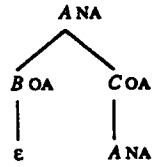
Thus, in considering the language derived by G, we need only be concerned with trees derived from auxiliary trees whose root and foot are labeled by S.

There are 5 kinds of auxiliary trees in A.

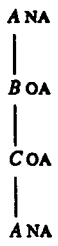
1. For each tree of the following form include $A^a/C^a/B^c \in f(\epsilon)$ and $A^c/C^a/B^c \in f(\epsilon)$



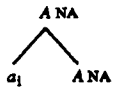
2. For each tree of the following form include $A^a\backslash B^a/C^c \in f(\epsilon)$ and $A^c\backslash B^a/C^c \in f(\epsilon)$



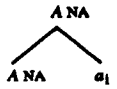
3. For each tree of the following form include $A^a/B^c/C^c \in f(\epsilon)$ and $A^c/B^c/C^c \in f(\epsilon)$



4. For each tree of the following form include $A^a\backslash A_i \in f(\epsilon)$, $A^c\backslash A_i \in f(\epsilon)$ and $A_i \in f(a_i)$



5. For each tree of the following form include $A^a/A_i \in f(\epsilon)$, $A^c/A_i \in f(\epsilon)$ and $A_i \in f(a_i)$



The CCG, G', in deriving a string, can be understood as mimicking a derivation in G of that string in which trees are adjoined in a particular order, that we now describe. We define this order by describing the set, $T_i(G)$, of all trees produced in i or fewer steps, for $i \geq 0$.

$T_0(G)$ is the set of auxiliary trees of G.

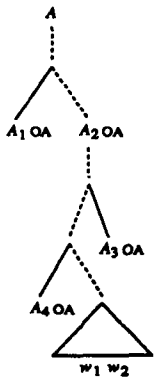
$T_i(G)$ is the union of $T_{i-1}(G)$ with the set of all trees γ produced in one of the following two ways.

1. Let γ' and γ'' be trees in $T_{i-1}(G)$ such that there is a unique lowest OA node, η, in γ' that does not dominate the foot node, and γ'' has no OA nodes. γ is produced by adjoining γ'' at η in γ'.

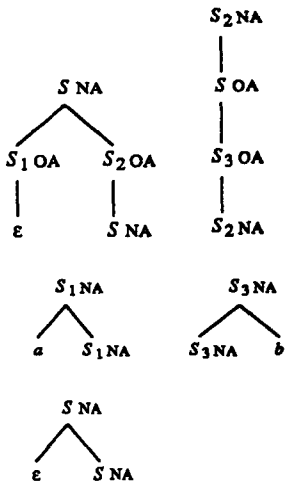
2. Let γ' be trees in $T_{i-1}(G)$ such that there is OA node, η, in γ' that dominates the foot node and has no lower OA nodes. γ is produced by adjoining an auxiliary tree β at η in γ'.

Each tree $\gamma \in T_i(G)$ with frontier $w_1 A w_2$ has the property that it has a single spine from the root to a node that dominates the entire string $w1Aw_2$. All of the OA nodes remaining in the tree fall on this spine, or hang immediately to its right or left. For each such tree γ there will be a derivation tree in G', whose root is labeled by a category c and with frontier $w_1 w_2$, where c *encodes* the remaining obligatory adjunctions on this spine in γ.

Each OA nodes on the spine is encoded in c by a slash and nonterminal symbol in the appropriate position. Suppose the OA node is labeled by some A. When the OA node falls on the spine c will contain $/A^c$ (in this case the direction of the slash was arbitrarily chosen to be forward). When the OA node falls to the left of the spine c will contain $\backslash A^a$, and when the OA node falls to the right of the spine c will contain $/A^a$. For example, the following tree is encoded by the category $A\backslash A_1^a/A_2^c/A_3^a\backslash A_4^a$



We now give an example of a TAG for the language $\{a^n b^n \mid n \geq 0\}$ with crossing dependencies. We then give the CCG that would be produced according to this construction.



$S^a \backslash S^a_1 / S^c_2 \in f(\epsilon)$
$S^a_2 / S^c / S^c_3 \in f(\epsilon)$
$S^a_1 \backslash A \in f(\epsilon)$
$S^a_3 / B \in f(\epsilon)$
$A \in f(a)$
$S^a \backslash S_\epsilon \in f(\epsilon)$
$S_\epsilon \in f(\epsilon)$

$S^c \backslash S^a_1 / S^c_2 \in f(\epsilon)$
$S^c_2 / S^c / S^c_3 \in f(\epsilon)$
$S^c_1 \backslash A \in f(\epsilon)$
$S^c_3 / B \in f(\epsilon)$
$B \in f(b)$
$S^c \backslash S_\epsilon \in f(\epsilon)$

4 Derivations Trees

Vijay-Shanker, Weir and Joshi [14] described several properties that were common to various constrained grammatical systems, and defined a class of such systems called Linear Context-Free Rewriting Systems (LCFRS's). LCFRS's are constrained to have linear non-erasing composition operations and derivation trees that are structurally identical to those of context-free grammars. The intuition behind the latter restriction is that the rewriting (whether it be of strings, trees or graphs) be performed in a context-free way; i.e., choices about how to rewrite a structure should not be dependent on an unbounded amount of the previous or future context of the derivation. Several well-known formalisms fall into this class including Context-Free Grammars, Generalized Phrase Structure Grammars (GPSG), Head Grammars, Tree Adjoining Grammars, and Multicomponent Tree Adjoining Grammars. In [14] it is shown that each formalism in the class generates semilinear languages that can be recognized in polynomial time.

In this section, we examine derivation trees of CCG's and compare them with respect to those of formalisms that are known to be LCFRS's. In order to compare CCG's with other systems we must choose a suitable method for the representation of derivations in a CCG. In the case of CFG, TAG, HG, for example, it is fairly clear what the elementary structures and composition operations should be, and as a result, in the case of these formalisms, it is apparent how to represent derivations.

The traditional way in which derivations of a CCG have been represented has involved a binary tree whose nodes are labeled by categories with annotations indicating which combinatory rule was used at each stage. These derivation trees are different from those systems in the class of LCFRS's in two ways. They have context-free path sets, and the set of categories labeling nodes may be infinite. A property that they share with LCFRS's is that there is no dependence between unbounded paths. In fact, the derivation trees sets produced by CCG's have the same properties as those produced by LIG's (this is apparent from the construction in Section 3.1).

Although the derivation trees that are traditionally associated with CCG's differ from those of LCFRS's, this does not preclude the possibility that there may be an alternative way of representing derivations. What appears to be needed is some characterization of CCG's that identifies a finite set of elementary structures and a finite set of composition operations.

The equivalence of TAG's and CCG's suggests one way of doing this. The construction that we gave from TAG's to CCG's produced CCG's having a specific form which can be thought of as a normal form for CCG's. We can represent the derivations of grammars in this form with the same tree sets as the derivation tree sets of the TAG from which they were constructed. Hence CCG's in this normal form can be classified as LCFRS's.

TAG derivation trees encode the adjunction of specified elementary trees at specified nodes of other elementary trees. Thus, the nodes of the derivation trees are labeled by the names of elementary trees and tree addresses. In the construction used in Section 3.2, each auxiliary tree produces assignments of *elementary* categories to lexical items. CCG derivations can be represented with trees whose nodes identify elementary categories and specify which combinatory rule was used to combine it.

For grammars in this normal form, a unique derivation can be recovered from these trees, but this is not true of arbitrary CCG's where different orders of combination of the elementary categories can result in derivations that must be distinguished. In this normal form, the combinatory rules are so restrictive that there is only one order in which elementary categories can be combined. Without such restrictions, this style of derivation tree must encode the order of derivation.

5 Additions to CCG's

CCG's have not always been defined in the same way. Although TAG's, HG's, and CCG's, can produce the crossing dependencies appearing in Dutch, two additions to CCG's have been considered by Steedman in [12] to describe certain coordination phenomena occurring in Dutch. For each addition, we discuss its effect on the power of the system.

5.1 Unbounded Dependent Structures

A characteristic feature of LCFRS's is that they are unable to produce two structures exhibiting an unbounded dependence. It has been suggested that this capability may be needed in the analysis of coordination in Dutch, and an extension of CCG's has been proposed by Steedman [12] in which this is possible. The following schema is included.

$$x^* \text{ conj } x \rightarrow x$$

where, in the analysis given of Dutch, x is allowed to match categories of arbitrary size. Two arbitrarily large structures can be encoded with two arbitrarily large categories. This schema has the effect of checking that the encodings are identical. The addition of rules such as this increases the generative power of CCG's, e.g., the following language can be generated.

$$\{(wc)^n \mid w \in \{a,b\}^*\}$$

In giving analysis of coordination in languages other than Dutch, only a finite number of instances of this schema are required since only bounded categories are involved. This form of coordination does not cause problems for LCFRS's.

5.2 Generalized Composition

Steedman [12] considers a CCG in which there are an *infinite* number of composition rules for each $n \geq 1$ of the form

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

$$(\ldots(y|_1 z_1)|_2 \ldots|_n z_n) \quad (x\backslash y) \rightarrow (\ldots(x|_1 z_1)|_2 \ldots|_n z_n)$$

This form of composition is permitted in Parenthesis-free Categorial Grammars which have been studied in [5,4], and the results of this section also apply to this system.

With this addition, the generative power of CCG's increases. We show this by giving a grammar for a language that is known not to be a Tree Adjoining language. Consider the following CCG. We allow unrestricted use of *arbitrarily* many combinatory rules for forward or backwards generalized composition and application.

$$\begin{array}{ll} f(\epsilon) = \{S\} & \\ f(a_1) = \{A_1\} & f(b_1) = \{B_1\} \\ f(a_2) = \{A_2\} & f(b_2) = \{B_2\} \\ f(c_1) = \{S\backslash A_1/D_1/S\backslash B_1\} & f(d_1) = \{D_1\} \\ f(c_2) = \{S\backslash A_2/D_2/S\backslash B_2\} & f(d_2) = \{D_2\} \end{array}$$

When the language, L, generated by this grammar is intersected with the regular language

$$\{\, a_1^* a_2^* b_1^* c_1^* b_2^* c_2^* d_2^* d_1^* \,\}$$

we get the following language.

$$\{\, a_1^{n_1} a_2^{n_2} b_1^{n_1} c_1^{n_1} b_2^{n_2} c_2^{n_2} d_2^{n_2} d_1^{n_1} \mid n_1, n_2 \geq 0 \,\}$$

The pumping lemma for Tree Adjoining Grammars [13] can be used to show that this is not a Tree Adjoining Language. Since Tree Adjoining Languages are closed under intersection with Regular Languages, L can not be a Tree Adjoining Language either.

6 Conclusions

In this paper we have considered the string languages and derivation trees produced by CCL's. We have shown that CCG's generate the same class of string languages

as TAG's, HG's, and LIG's. The derivation tree sets normally associated with CCG's are found to be the same as those of LIG's. They have context-free path sets, and nodes labeled by an unbounded alphabet. A consequence of the proof of equivalence with TAG is the existence of a normal form for CCG's having the property that derivation trees can be given for grammars in this normal form that are structurally the same as the derivation trees of CFG's. The question of whether there is a method of representing the derivations of arbitrary CCG's with tree sets similar to those of CFG's remains open. Thus, it is unclear, whether, despite their restricted weak generative power, CCG's can be classified as LCFRS's.

References

[1] A. E. Ades and M. J. Steedman. On the order of words. *Ling. and Philosophy*, 3:517–558, 1982.

[2] A. V. Aho. Indexed grammars — An extension to context free grammars. *J. ACM*, 15:647–671, 1968.

[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] J. Friedman, D. Dai, and W. Wang. The weak generative capacity of parenthesis-free categorial grammars. In 11^{th} *Intern. Conf. on Comput. Ling.*, 1986.

[5] J. Friedman and R. Venkatesan. Categorial and Non-Categorial languages. In 24^{th} *meeting Assoc. Comput. Ling.*, 1986.

[6] G. Gazdar. *Applicability of Indexed Grammars to Natural Languages.* Technical Report CSLI-85-34, Center for Study of Language and Information, 1985.

[7] 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 1983.

[8] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1), 1975.

[9] M. Steedman. Combinators and grammars. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, Foris, Dordrecht, 1986.

[10] M. Steedman. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 1987.

[11] M. Steedman. Gapping as constituent coordination. 1987. m.s. University of Edinburgh.

[12] M. J. Steedman. Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–568, 1985.

[13] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars.* PhD thesis, University of Pennsylvania, Philadelphia, Pa, 1987.

[14] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In 25^{th} *meeting Assoc. Comput. Ling.*, 1987.

[15] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Tree adjoining and head wrapping. In 11^{th} *International Conference on Comput. Ling.*, 1986.

[16] D. J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis, University of Pennsylvania, Philadelphia, Pa, in prep.