ANTHONY E. ADES AND MARK J. STEEDMAN

ON THE ORDER OF WORDS

1. Introduction

A complete theory of human grammatical performance must include components of two distinct kinds. The first is a grammar, defining a set of strings which comprises all and only the sentences of the language in question. The second component is a processor, or device which applies the rules of the grammar so as to either produce or analyse the grammatical strings. The processor can be characterised in two quite independent respects. It may be defined as a particular variety of automaton, characterised as (for example) working from left to right or from right to left through the text, as applying the rules of the grammar 'top down' or 'bottom up', or as doing semantic interpretation 'in parallel' or 'in series' with syntactic processing. But the processor can also be characterised by the particular mechanism which it uses to resolve ambiguities as to which particular rule of grammar should be applied at a particular point in analysis or generation. Mechanisms of this last kind must prevent the explosion of processing demands and the inefficiencies that will result if such ambiguities are allowed to proliferate.

What follows is a theory of competence in the accepted sense: we shall have nothing further to say about efficiency, and very little to say about possible mechanisms for the resolution of ambiguity. However, there is every reason for a theory of competence to be straightforwardly related to what is known about human linguistic performance. In particular, it is an advantage in a competence theory if it is clear how it could be turned into a processor which works 'from left to right' through the text, and appears to deal with semantics as nearly as possible in parallel with syntax, as Marslen-Wilson (1973), Tyler and Marslen-Wilson (1977), Marslen-Wilson and Tyler (1980), Crain (1980), and others have suggested that the human parser does. It is only once it is clear how a human listener could so process all and only the grammatical strings of a language that the further limitations that are imposed by the mechanisms which resolve local ambiguities (and which on occasion misresolve them) can be investigated. The goal of the present paper is therefore to outline a proposal which is simultaneously a competence theory and a theory of what the human listener would have to do to construct any

Linguistics and Philosophy 4 (1982), 517–558. 0165–0157/82/0044–0517$04.20

given parsing for all and only the grammatical strings of the language, without regard to the logically posterior question of ambiguity resolution or how that *particular* parse might be arrived at.

Because it is the process of analysis that has been most extensively studied by psychologists, the model is presented as an idealised recogniser, rather than as the idealised generator that is more common in modern linguistics. The choice of direction has of course no significance for the theory of grammar. For the purpose at hand it is no more necessary to explain how this 'recogniser' might be equipped to handle ambiguities efficiently than it is for generative grammarians to explain how their 'generators' might choose which base rules to apply.

1.1. *The linguistic problem*

What one might have expected of natural languages is that their grammars would be context free (CF), with all the attendant benefits of a straightforward relation between syntax and semantics, as well as various desirable effects upon learnability and processing demands. But of course Transformational Grammar (TG) is founded on the suggestion that natural languages can *not* be usefully described with CF grammars. Most importantly, transformationalists have pointed out that significant generalisations are liable to be missed if a grammar of English fails to describe such sets of sentences as the following, not merely as grammatical, but also as related.

(1) (a) He must love her.
(b) Must he love her?
(c) Her he must love!

Chomsky formalised the relation by deriving such sets from a single underlying form, closely related to (1a). To this underlying structure, generated by a CF base grammar, could apply a number of structure changing transformations, rearranging constituents into the interrogative and topicalised forms (1b, c) among others.

However, a principal concern of Transformational grammarians has been to specify ways in which the form of the transformations that occur in natural languages are more constrained than unrestricted structure changing operations, for it is certain that they are drawn from a very limited class indeed. Moreover, even the transformations that *are* found in a language are subject to unexplained restrictions upon their applicability. For instance, having defined movement transformations that relate the numbered noun phrases in (2a) to the canonical positions marked t_1 and t_2, they must be restricted to prevent them generating (2b),

for several linguists have noticed that in sentences where more than one constituent has been 'moved' by transformations, they can be linked to their respective extraction sites by nested, but not crossed, lines (Bach, 1977, p. 150; Fodor, 1978, pp. 51, 52).

(2) (a) (The wood)$_1$ is too rough for (these nails)$_2$ to be easy for me to hammer t_2 into t_1.
(*b*) *(The nails)$_1$ are too blunt for (this rough wood)$_2$ to be easy for me to hammer t_1 into t_2.

What is more, this 'Nested Dependency Constraint' applies with great generality over the languages of the world[1].

The many accounts delimiting the form that transformational rules may take and the conditions under which they may apply constitute and indispensable contribution to the study of language, and the generalisations that have emerged constitute in great part the data upon which the present study rests. However, even the most ingeniously restricted set of transformations and constraints upon them does not explain why it is that only transformations drawn from that set exist, constrained in those particular ways, since they appear to require for their processing an apparatus of a power that would be capable of performing much less restricted operations as well. The task of explaining the form of natural language will only be complete when it is shown that the restrictions stem from the involvement of some specific class of mechanism that is more restricted that which is implicated by transformations.

A particularly important class of constraints are those upon 'unbounded movements' such as the constraint exemplified above, because the negative constraints that prohibit the transformational apparatus from performing an otherwise perfectly computable operation are particularly problematic if the competence grammar is to be directly realisable as a processor, as we have suggested it should be. The only straightforward interpretation of a standard theory grammar as a processor – as opposed to the many less absurd but equally less specified interpretations – is as one which first builds a surface structure, then proceeds via the transformations to a deep structure and an interpretation. But the constraints of Ross (1967) would imply that such a processor first builds a presumably perfectly interpretable structure, via the transformational rules, but then has to reject it after consulting a list of forbidden movements or structures. The filters of Chomsky and Lasnik (1977) have similarly unpalatable implications for such a processor.

An important step towards an explanation has been provided by the

Base Generation Hypothesis, which is broadly represented in the work of Bresnan (1978), Brame (1976, 1978), Gazdar (1981a, 1981b) and Peters (1980), among others. The hypothesis eschews transformations, and generates all constituent orders directly in some form of base grammar. Gazdar (1981b), in particular, has argued persuasively that there is no good reason to regard English or any other language as being more complex than a CF language, though not that it is necessarily best described using a CF grammar alone. Nevertheless, to the extent that, in the interests of capturing generalisations, these systems include rules of grammar which (whilst they themselves only generate a CF language) are nevertheless drawn from classes of rules which potentially have greater than CF power, the question of why natural languages should include rules from such powerful classes remains unanswered. For if the mechanism can compute such rules there is no particular reason for it to confine itself to just the ones which happen to generate CF languages. And if the rules are *not* computed by a processor, then the generalisations remain unexplained[2]. Typical of such special rules are the metarules of Gazdar, and the potentially arbitrary relationships between surface syntax and deep structure or functional/semantic representation of Brame (1978), Bresnan (1978), and the related computational models of Woods (1968) and Marcus (1977), among others. Among the complications attendant upon such rules are increased difficulty of learning, non-isomorphism between syntactic and semantic rules and non-correspondence between rules of grammar and steps in processing.

1.2. *Outline*

The rest of the paper will take the following form. In the next section, we interpret certain constraints on 'movement' as suggesting that a CF grammar or some natural generalisation thereof can be devised for both 'base' and 'movement' phenomena, and point out that such grammars are directly compatible with a psychologically plausible mechanism which proceeds from left to right, parsing and building a semantic interpretation in one pass. We then present a categorial lexicon and four rule schemata whose function is to determine word and constituent order. A related processor which applies the rules is described, without of course any discussion of the further problem of ambiguity resolution.

In Section 3, the model is applied to the basic forms of English main clauses. We confine ourselves to phenomena of 'unbounded movement' and the class of Root Transformations (Emonds, 1975), as it is these that have proved most resistent to reformulation in nontransformational terms.[3]

Section 4 shows that the phenomena associated with the Left Branch Condition of Ross (1967) can be naturally expressed *within* the grammar, rather than by ad hoc negative constraints exterior to the grammar, as can phenomena which have motivated major constraints upon Root Transformations.

In Section 5, certain extraction constraints are shown to follow from the way in which each of the rule schemata expresses generalisations about the types of semantic-syntactic entities that they combine, and from other general features of the model.

In the final section, the present proposals are briefly compared to some alternatives from both linguistic and psychological points of view.

2. The Model

2.1. *Stacking properties and Context Free-ness*

Both Bach (1977) and Fodor (1978) pointed out that the unacceptability of crossed dependencies illustrated in (2) would be predicted by a left-to-right processor that placed preposed constituents into a Push Down Store (PDS) or stack, and restored them later to their underlying positions. A mechanism of this type had already been introduced in the machine parsing literature. Woods (1973, 119), following a suggestion made by Thorne Bratley and Dewar (1968), showed that preposed items such as relative pronouns could be restored to their canonical position by placing them in a store called HOLD, from which they could later be retrieved when the site of their extraction was encountered. He speculated that the same mechanism could be used for other varieties of movements, and further conjectured that the HOLD store was a PDS. (See also Kaplan, 1973).

Something like the Nested Dependency Constraint seems also to apply to all those constructions which Emonds (1976) classified as arising from Root Transformations[4]:

(3) (a) Whom_1 did_2 you 0_2 see 0_1.
(b) [Far more serious]$_1$ was_2 the leak in the roof 0_2 0_1.
(c) [Leaning against the bedpost]$_1$ was_2 a policeman 0_2 0_1.
(d) [Up the street]$_1$ walked_2 Bill 0_2 0_1.
(e) [Seldom]$_1$ had_2 he 0_2 0_1 heard a more ridiculous proposal.

If, as these data strongly suggest[5], *all* movement is constrained by the nesting property, it becomes tempting to make a significant simplification in the theory. It is suspicious that a stack is needed to constrain movement from canonical base structure positions, for a stack will also

be necessary to process the surface strings themselves, which are describable as context free. Our proposal is simply that it is the *same* stack that is responsible both for 'movements' and for processing surface CF grammar. The effect of this proposal is to remove the Nested Dependency Constraint and the stack from the orbit of performance constraints on movement and to make them a principle of grammar. The implication is that both movement and base phenomena are the manifestation of one context free or near-context free system.

This interpretation of the nesting property, like other base-generative approaches, offers the attractive possibility of a syntax and a syntactic processor that are non-autonomous in the strongest sense of the word. Since there is no need for a separate pass through the string to restore displaced constituents to canonical position before semantic interpretation can begin, it follows that semantic interpretations can be assembled in parallel with syntactic processing. Indeed, there is no need for any autonomous syntactic structure to be built at all: the semantic representation can be built directly. Moreover, once an analyser can be made to build semantic interpretations immediately, it can also be made to evaluate subexpressions while the analysis is still in progress. An example of how syntactic processing, semantic interpretation and evaluation can be combined into a single left to right pass is to be found in Davies and Isard (1972). The technique, which is well known to compiler writers (cf. Bolliet, 1968), has been exploited in computer models of language interpretation by, for example, Winograd (1972), Johnson-Laird (1977) and Steedman and Johnson-Laird (1978). A grammar of the base-generative variety therefore offers the attraction of a linguistic theory which is straightforwardly relatable to a considerable amount of evidence suggesting that human listeners not only assemble semantic interpretations as nearly as possible in parallel with syntactic processing, but also use semantics and even reference interactively to guide syntactic and lexical decisions (Marslen-Wilson, 1973; Tyler and Marslen-Wilson, 1977; Marslen-Wilson and Tyler, 1980; Crain 1980)[6].

2.2 *The Categorial Notation*

The rules that follow will be presented using the notation of Categorial Grammar (CG) (Ajdukiewicz, 1935; Bar-Hillel, 1960; Lyons, 1968; Lewis, 1970). A Categorical Grammar consists of a series of lexical entries specifying the syntactic role of each morpheme in the language. Such entries take the following form:

(4) Morpheme: X/Y

For example, determiners bear the category NP/N, identifying them as entities which combine with nouns to yield noun-phrases. For primitive categories Y is null: the category of a noun is simply N.

An item with a category of the form X/Y is to be thought of as a function. For example the category VP/NP of transitive verbs identifies them as functions from NPs onto VPs. In the first place, such functions can be thought of as mapping between syntactic entities. However, the categories can also be thought of as a shorthand for the semantics of the entities in question, and the functions can then be thought of as mapping between semantic entities, such as interpretations, intensions, or whatever. (For example, the category VP/NP of a transitive verb can be thought of as a shorthand for a function from NP intensions onto VP interpretations.) We have relegated to an Appendix our few further remarks concerning the nature of such semantics, and will continue for the most part to use the syntactic shorthand, since for present purposes it makes no difference whether semantic entities are thought of as expressions in Lambda calculus, procedures, or anything else. The important assumption is parallel to the basic assumption of Montague grammar: there is a one-to-one correspondence between syntactic categories and rules of semantic interpretation.

As befits their basically semantic nature, the functions described above do not themselves determine whether their arguments are to appear to their right or to their left. The responsibility for ordering lies with a number of 'combination rules', so called because of their direct relation to operations of a processor. These rules of the grammar are described next.

2.3. *Forward Combination*

The first of the combination rules combines a function with an argument to its right. It can provisionally be written as the following rule schema:

(5) Forward Combination
$X/Y \; Y \Rightarrow X$

X and Y are variables and are allowed to match any category.

Consider the following fragment of lexicon:

(6) the: NP/N
frog: N
find: VP/NP

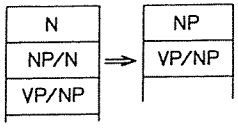
An example of the operation of the Forward Combination rule is the following, where the variables X and Y match the atomic categories NP

and N, respectively:

(7) $[\text{the}]_{NP/N}\ [\text{frog}]_{N} \Rightarrow [\text{the frog}]_{NP}$

The Forward Combination rule (5) can be thought of as a rewrite rule schema which happens to have been written down backwards. But it may be easier to imagine a process that from time to time puts the successive words of the string onto a stack, and that this and other combination rules can apply to the top two items on the stack, replacing them with the result. In the case of Forward Combination an X/Y and a Y on top of it will be replaced by the single result X. (Such a 'processor' is related to a class of Non-Deterministic Push Down Automata known as 'Shift and Reduce' parsers). For example, in parsing the phrase *find the frog*, the words *find*, *the*, and *frog*, are placed onto the stack by the automaton. Then the Forward Combination rule applies to reduce $[\text{the}]_{NP/N}$ and $[\text{frog}]_{N}$ to $[\text{the frog}]_{NP}$, as in the following diagram:

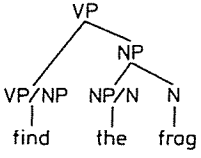
(8)



At this point the conditions for Forward Combination are met again. The VP is constructed from $[\text{find}]_{VP/NP}$ and $[\text{the frog}]_{NP}$.

The above process is isomorphic to the familiar kind of phrase structure tree:

(9)



However, it should be remembered that this structure really only describes the successive stages of building a directly evaluable semantic interpretation.

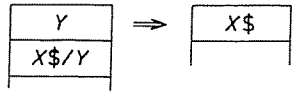
At this point, the basic notation must be elaborated a little. Some verbs, of course, can take more than one argument. For example, *put* is a function from NPs onto a function from PPs onto VPs, implying a category (VP/PP)/NP. It will be notationally convenient for the presentation of the later rules to omit the brackets under a convention of 'left associativity', and to write the category of *put* as VP/PP/NP, where it is understood that the rightmost slash is the 'major' or highest level slash. Then the Forward Combination rule is generalised as follows. We define

a string symbol, \$, to be either null or a string of alternating atomic category symbols and slashes beginning with a slash and ending with a symbol, (such as /NP, /PP/NP, /NP/NP/NP, and so on). The form of the Forward Combination rule is now:

(10) Forward Combination
$X\$/Y\ Y \Rightarrow X\$$

In its incarnation as a rule in an acceptor, it can be represented pictorially as follows:

(11)



It is a theoretical advantage of the categorial notation (as for other lexically-based grammars) that it is unnecessary to specify the various expansions of, for example, VP, in the grammar *and* redundantly in the lexicon. All the work is done by the categorization of, for instance, *eat* as VP/NP, *walk*, as VP/PP and *put* as VP/PP/NP. (Of course, any morpheme may have more than one categorization. The selection of the category which is appropriate to a given parsing is a consideration neither of the grammar, nor of the structure of the corresponding automaton per se, but of the ambiguity resolving mechanism).

2.4 *Backwards Combination*

The second combination rule is similar to Forward Combination, except that it combines a function with an argument to its left, or in terms of the processor, one which is beneath it on the stack.

Consider such strings as:

(12) (a) Her he must love.
(b) Who are you looking at?

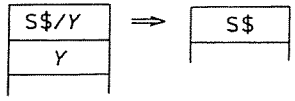
Each begins with a preposed constituent that has been 'moved' from the sentence that follows. Because the model we are proposing has only one stack with which to combine items, and because our rules are confined to ones that operate only on adjacent items, there is no alternative to regarding the strings *He must love* and *are you looking at* as being functions of type S/NP, that map (object) NP's onto sentences. Thus, both sentences (12) are instances of NP followed by S/NP. Because in English no other category (at least among the ones we are considering here) combines with an argument to its left, we stipulate that Backwards Combination is

restricted to S-nodes, and write it thus:

(13) Backwards Combination
$Y\ S\$/Y \Rightarrow S\$$

where $ is a string of the kind defined earlier. The rule can be represented pictorially as:

(14)



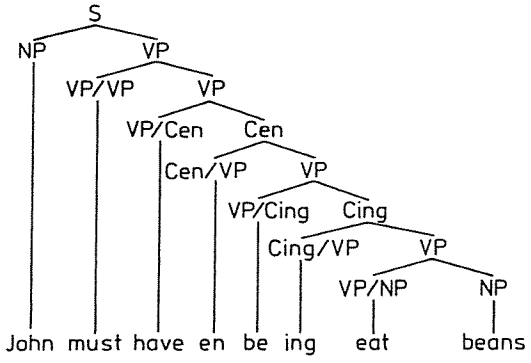
Exactly what kind of entity corresponds to incomplete sentences like [He must love]$_{S/NP}$, and how such entities are built, is the subject of the next section.

2.5. *Forward Partial Combination*

For a right-branching structure, the Forward Combination Rule alone would allow no combination to take place until the rightmost lowest word had been put onto the top of the stack. Consider, for example, the sentence (15) and the structure implied by Jackendoff (1977), among others:

(15) (a) John must have been eating beans

(b)



If there is to be any possibility of constructing a semantic interpretation before *beans* is encountered, then something other than the basic Forward rule must be capable of combining one node with another before the later one is complete. For example, if the categories of *must* and *have* are:

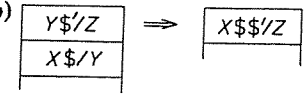
(16) must: VP/VP
have: VP/Cen

(where Cen is the category of the past participle, as in 15b), it would be natural to wish to combine them into a category VP/Cen corresponding to the semantic interpretation of the compound *must have*. In any case, whether or not this 'partial combination' is desirable a priori, it is a forced move if the processor is to build entities corresponding to incomplete clauses, of the form $S\$/X$, mentioned in the account of topicalization and the Backwards Combination rule. Within the present framework, the job can be done with a single further rule schema.

The rule is:

(17) (a) Forward Partial Combination
$X\$/Y\ Y\$'/Z \Rightarrow X\$\$'/Z$

(b)



(where $ and $' are two (possibly different, possibly null) strings of the kind defined earlier in Section 2.3 above)[7]. Like the earlier rules, the above is to be thought of as combining the semantic interpretations associated with the two items to which it applies into a single semantic interpretation. Just as the operations of Forward and Backward Combination correspond to the basic semantic operation of *application* of a function to its argument, so Partial Combination corresponds to the equally basic and well-defined semantic operation of *composing* two functions of category $X\$/Y$ and $Y\$'/Z$ into a single function of category $X\$\$'/Z$. Intuitively it is obvious that if you have a function of type VP/VP for *must* and one of type VP/Cen for *have*, then you know everything necessary to build their composition – a function of type VP/Cen for *must have*. The semantics that we have in mind is discussed briefly in the Appendix. There is considerable precedent for rules of this type, both within Categorial Grammar (Ajdukiewicz, 1935; Bar-Hillel, 1960; Geach, 1972), and from one other linguist concerned with processing and direct assembly of semantic representations (Kimball, 1975)[8].

The rule of Partial combination implies that items like [John will have]$_{S/Cen}$ may on occasion be in some sense constituents, corresponding to fully interpreted semantic representations. However, the theory is more orthodox than it might appear in this respect: the classical constituency relations and the associated semantic interpretations are defined in the lexicon. The claim is simply that there are stages in processing where representations of what are in classical terms incomplete constituents are assembled. As stated above the claim appears to follow independently from both the facts of preposing (12) and from

the facts about psychological processing and its interaction with semantics. (English does not appear to include a rule of 'Backward Partial Combination', although the treatment of affixes below bears some resemblance to such an operation. However, since the same arguments from semantic coherence and psychological plausibility would apply, we anticipate its occurrence in other languages). It will be noted that the assembly of such 'partial' representations can accomplish much more than the construction of autonomous, uninterpreted syntactic structures. The entities produced by partial combination correspond to fully interpreted functions that may be applied to arguments of the appropriate category, without being modified by a syntactic process such as movement of a constituent into the structure.

2.6. *Affix-verb Compounds and the Introduction of the S-node*

The lexical categories that have been introduced so far have been uncontroversial, and are typically found in other categorial grammars. Our treatment of the affixes, and particularly of Tense, is, however, slightly different from those others.

Verb-affix compounds like *having* and *been* are composed of two parts each of which plays an important role in the combination of adjacent elements. We shall assume that it is words, not morphemes, that are entered to the top of the stack, but that the category of compound words can be determined from the categories of their constituent morphemes. As suggested in (15), following Jackendoff (1977), we take *en* and *ing* as complementizers over VP, i.e. functions from VPs onto complements categorized as Cen and Cing respectively. Since every verb has a category of the form VP$, and such affixes are functions over VPs, the Affix Cancellation rule can initially be stated as:

(18) Affix Cancellation
$$VP\$.X/VP \Rightarrow X\$$$

(The dot indicates that the morphemes are parts of a single word.) Thus, *eating* becomes categorized as Cing/NP, and *eaten* as Cen/NP:

(19) (a) $\underset{VP/NP\,Cing/VP}{eat.ing} \Rightarrow \underset{Cing/NP}{eating}$

(b) $\underset{VP/NP\,Cen/VP}{eat.en} \Rightarrow \underset{Cen/NP}{eaten}$

The Tense affix combines with verb stems in the same way. However, its category is somewhat more complex. We assume Tense to be a function from NP (subjects) onto functions from VPs onto Sentences:

(20) Tense: S/VP/NP

The above category makes Tense the 'glue' that attaches a subject and a predicate. However, since Tense always comes attached to the verb as an affix, it acts by modifying the verb under the Affix Cancellation rule, which must be elaborated so that it will apply to the more complex category of Tense. It is written:

(21) Affix Cancellation
$$VP\$.X/VP\$' \Rightarrow X\$\$'$$

($ and $' are two (possibly null, possibly different) strings of the sort that sort discussed previously). The rule and the categorization for Tense have the effect of making Tense map verb stems (which are all functions onto VP having the form VP$) onto functions onto S of the form S$/NP, which first consume a subject NP, then consume the argument(s) originally required by the verb itself. The following compounds receive the following values:

(22) (a) $\underset{VP/NP_o}{eat} . \underset{S/VP/NP_s}{ed} \Rightarrow \underset{S/NP_o/NP_s}{ate}$

(b) $\underset{VP/VP}{will} . \underset{S/VP/NP_s}{s} \Rightarrow \underset{S/VP/NP_s}{will}$

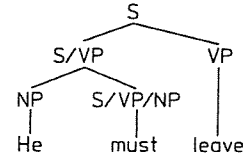
(c) $\underset{VP/Cen}{have} . \underset{S/VP/NP_s}{ed} \Rightarrow \underset{S/Cen/NP_s}{had}$

(d) $\underset{VP/PP/NP_o}{put} . \underset{S/VP/NP_s}{ed} \Rightarrow \underset{S/PP/NP_o/NP_s}{put}$

(The subcripts identifying NP's as either object or subject are included only for the reader's convenience: the semantic representation of the affix and verb stem, together with the Affix Cancellation rule will automatically associate NP arguments with the appropriate (surface[9]) function).

The account of tense offered here induces such highly unorthodox analyses as the following:

(23)



A related analysis has recently been proposed by Schmerling (1980) on independent syntactic grounds. However, we offer no further justification for this startling analysis at this point, except that it is the one that works.

The four rules are presented together in their final form in the

Appendix, together with some further remarks on the semantics of Forward Partial Combination.

2.7. *Summary of the Model*

The possibility of any two items combining (in whatever order), and the result of their combination, is determined solely by their categories, which in turn simply reflect their semantics. The categorial lexicon can thus be thought of as defining a grammar or class of grammars which is 'free' with respect to the linear order of functions and arguments. To that extent, it resembles an order-free base grammar. It is the four combination rules, which are equivalent to production rule schemata, that have the role of determining what linear order(s) two combinable items may occur in. Other languages may have (somewhat) different lexicons and/or different restrictions on rule schemata drawn from the same restricted class. (For example, (24) suggests that some more general form of backward combination occurs in German VP.)

(24) ... dass er Eier gegessen haben muss
... that he eggs eaten have must
'... that he must have eaten eggs'

The rules by themselves are rules of grammar in the usual sense of the term. But they can, with the addition of a single Push Down Stack, be thought of as specifying a Non-Deterministic Push Down Automaton (NDPDA), when taken in conjunction with a trivial control mechanism which can be informally stated as follows:

(25) Until the string is empty and no rule matches the topmost items on the stack, either (a) apply a rule to the topmost items and replace them with the result, or (b) put the next word on top of the stack.

The above is all that is needed to convert the 'competence' grammar into a model of the steps a human listener performs while undertaking a given parse. (It ignores, as ever, the separate question of how a listener might choose *which* parse(s) to pursue). Any base generative grammar can be embodied in such an automaton. What is interesting about this one is that, because of the Partial Combination rule, it is immediately compatible with the direct word-by-word assembly of a semantic interpretation in a single left-to-right parse, even for incomplete right-branching constituents, without the mediation of an autonomous syntactic representation.

3. Basic Sentence Constructions

The simple model outlined in the preceding section, together with lexical specifications of the kind introduced there, will accept a wide variety of English main clause constructions and will refuse to accept virtually all the ungrammatical ones. In the following sections, the possible permutations of a simple clause containing a subject an auxilliary and a transitive VP are first examined exhaustively, and then further main clause constructions are considered.

3.1. '*He must love her*'

There are twenty-four permutations of the words *He*, *must*, *love*, and *her*, only a few of which are grammatical. In order to demonstrate the model, it will be useful to consider all twenty-four. The categories of *he* and *her* are assumed to be NP, *love* is VP/NP, and *must* is VP/VP.S/VP/NP (a function from VPs onto VPs, combined with Tense), which reduces by Affix Cancellation to S/VP/NP. The examples are presented, for clarity, in terms of the automaton rather than the grammar rules alone. The working of the automaton will be illustrated using the convention that two items which combine will be underlined with the result written underneath the line. The line will be indexed with the initials of the rule that applied. Its result can of course take part in a further combination: thus the sequence of underlinings down the page represents the successive states of the automaton. (Affix Cancellation will not be shown). Consider, first, the 'canonical' form of the sentence:

(26) (a) He must love her

$$\frac{\overline{NP}\ \ \overline{S/VP/NP}}{S/VP}\,B$$

Once *He* and *must* have been put on the stack, Backwards Combination (13) applies and leaves $[\text{He must}]_{S/VP}$ on top of the stack. When the main verb *love* is entered, the conditions are met for Forward Partial Combination:

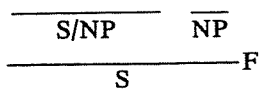
(26) (a) He must love her

$$\frac{\overline{S/VP}\ \ \overline{VP/NP}}{S/NP}\,FP$$

which leaves $[\text{He must love}]_{S/NP}$ on top of the stack. Finally, the object

her is entered and Forward Combination applies:

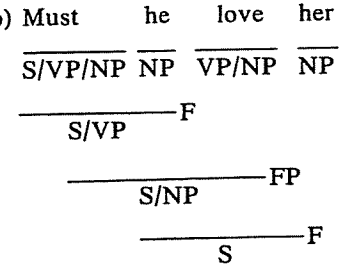
(26) (a) He must love her



The pass completes with the symbol S on top of the stack, which is the necessary and sufficient condition for a sentence to be grammatical according to the model.[10]

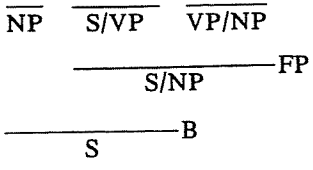
The analysis of the Question form is very similar

(26) (b) Must he love her

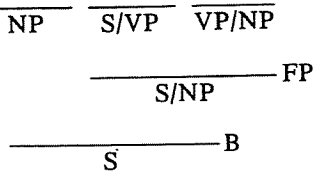


The process is the same as for the 'canonical' (26a), except that *must* and *he* are combined by the Forward rather than the Backward rule. Indeed, it is clear that Subject and tensed verb can occur in either order, so long as they are adjacent.

(26) (c) Her he must love ('Topicalisation')



(d) Who(m) must he love? ('Wh-question')

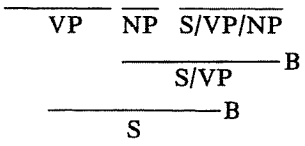


(As it stands, of course, the model does not explain why (26d) is possible only with a Wh-element, as opposed to **Her must he love*, which is just comprehensible but archaic. Neither does it explain why the subjects of

(26c, e) must be 'given' anaphors such as pronouns, rather than 'new' NPs. We return to these questions in Section 4.1.)

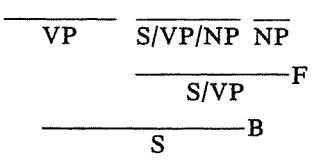
The model handles the one remaining grammatical permutation as follows:

(26) (e) Love her he must ('VP Preposing')



It allows just one further construction, and no more: the following is not grammatical, but is accepted:

(26) (f) *Love her must he



It should be obvious that no model of this kind which will accept (26d, e) – that is which will allow VP Preposing and the Object Wh-Question, – will rule out (26f). One might argue that the unacceptibility of the above is parallel to the unacceptibility of **Her must he love.* That is, preposing with inversion is only grammatical in English when the preposed entity is a Wh-item. The unacceptability of (26f) follows from the fact that there is no Wh-form for VPs in English[11].

Not one of the remaining eighteen permutations of the four words is accepted by the rules. The reason is that the form of the rules places very powerful constraints on possible rearrangements, which appear to correspond directly to the grammatical possibilities. The most important constraint is the following direct consequence of the fact that the combination rules can operate only on the top two items of the stack, and that the same stack is used both for 'movement' and structure building:

(27) The Adjacency Corollary.

> The rules are unable to combine two items that are separated on the stack by a third, unless the intervening item can first be combined with one or the other of them.

(This is a corollary of the model, not an additional assumption).

Because Tense has the category S/VP/NP, it follows that after it has been combined with its verb stem by Affix Cancellation, the first item to combine with the S-node must be the Subject. Because of the corollary (27) it follows that neither the verb phrase nor any component of it can intervene between Subject and tensed verb in an English clause. Among the remaining eighteen permutations, the following twelve are ruled out for this reason. (In some cases there are other reasons as well.)

(26) (g) *He *love her* must
(h) *He *love* must her
(i) *He *her* must love
(j) *He *her love* must
(k) *Must *her love* he
(l) *Must *love her* he
(m) *Must *her* he love
(n) *Must *love* he her
(o) *Her he *love* must
(p) *Her must *love* he
(q) *Love must *her* he
(r) *Love he *her* must

In fact virtually the only way that any material can ever intervene between the Subject and the tensed verb of an English clause under the adjacency corollary is when it constitutes a function of a category that allows it to combine with the tensed verb first, by Forward Partial Combination. We assume adverbials like *obviously* and *frequently* to bear such a category in sentences like *She frequently visits her mother.* The intervening material in Subject extractions like *Who [do you think] took the money*? is shown in Section 5.4 below to constitute a function over tensed S[12].

Another consequence of the rules and the form of the categories is that verbal complements must always follow the verb in question, unless the verb has already been absorbed into the S-node by partial combination, and the complement is topicalised (26c, d, e). The reason is that the Backwards rule applies only to functions of the form S$ which yield an S. Furthermore, the Backwards Combination rule permits only *complete* constituents to be preposed and there is no 'Backwards Partial Combination' rule in English.

(26) (s) *Her love he must

NP VP/NP S/VP B

*B

*

(t) *Her love must he
(u) *He must her love
(v) *Must he her love

For the same reason, the main verb cannot be preposed:

(26) (w) *Love he must her

VP/NP S/VP B NP

*

(x) *Love must he her

3.2. *Generality of Restrictions on English main clauses*
We shall not examine all English main clauses in such exhaustive detail. However, they are similarly constrained by the restrictions inherent in the model, as follows.

3.2.1. *Subject and tensed verb must be contiguous.* As already noted, given the rules and the categorization of Tense as S/VP/NP, tensed verb and Subject may occur in either order but cannot be separated by material from the VP, by the Adjacency Corollary (27). Consider, for example, the position of directional adverbs:

(28) (a) Jones came along ('Canonical Form')

NP S/PP/NP PP

S/PP B

S F

(b) Along came Jones ('Directional Adv. Preposing')

PP S/PP/NP NP

S/PP F

S B

(c) Along Jones came ('Adverb Topicalisation')

PP NP S/PP/NP

S/PP (B)

S (B)

(d) *Jones along came

NP PP S/PP/NP

(e) *Came along Jones

S/PP/NP PP NP

(f) *Came Jones along

S/PP/NP NP PP

S/PP (F)

S (F)

The ungrammatical strings (d) and (e) are excluded by the category of Tense and the Adjacency Corollary.

The last permutation, **Came Jones along*, is wrongly accepted by the model, because the Adjacency corollary is obeyed. This overgeneralisation has nothing to do with the position of the PP. It is rather due to a more general problem concerning the interaction of Tense and the arguments of verbs. The model also predicts that the following non-grammatical results of Subject-Tense inversion should be acceptable:

(29) (a) *Slept John?
(b) *Walked he up the street?
(c) *What ate he?
(c) *Apples ate he
(e) *Stands Scotland where it did?

Such strings have provided the motivation for a Do-Support/Deletion transformation[13]. We return to the question of how to exclude this and the earlier overgeneralisations in a later section, but note that it is at least encouraging that the overgeneralities are all constructions which are found in such nearly-related languages as Dutch and German, as well as in antique forms of English, and might therefore be expected to be governed by minor rules.

A striking example of the requirement that Subject and tensed verb should not be separated by any part of the VP is afforded by the unacceptability of sentences where a verb, itself in 'canonical position' with respect to the tensed verb, is interposed between the two:

(30) (a) Into the garden Maud was walking

PP NP S/Cing/NP Cing/NP

S/Cing (B)

S/PP (FP)

S (B)

(b) *Into the garden was walking Maud

PP S/Cing/NP Cing/PP NP

In stating the transformation Directional Adverb Preposing, Emonds (1976, 38) had to make special reference to the restriction, exemplified above, that it can only apply where there is no auxilliary verb. But this is not an isolated constraint. No such material can ever intervene between Subject and tensed verb, as can be seen from the following:

(31) (a) She might have kissed him
(b) Might she have kissed him
(c) Who might she have kissed
(d) Him she might have kissed
(e) *Might have she kissed him
(f) *Might him she have kissed
(g) *Who she have might kissed
(h) *Who might have she kissed, etc.

3.2.2. *Restrictions on the position of verbs* The other restriction that was pointed out in Section 3.1 followed from the restricted nature of backward combination in English. Unless the verb's complement has been preposed to sentence initial position, it must immediately follow the verb. This observation generalizes to other constructions.

(32) (a) *She must eaten apples have

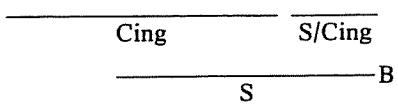
S/VP Cen VP/Cen

(b) *She has standing on the corner been

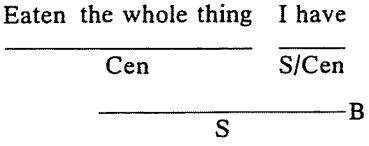
S/Cen Cing Cen/Cing

In neither case can the derivation proceed any further, as the last two items (*have* and *been*) cannot combine with their complements, nor can the complements combine with the S-node. The model permits the fronting of participial phrases as instances of topicalization, analogous to the fronting of Object NP (26c) or an Adverb (28c)[14]:

(33) (a) (She said she would be standing at the bus-stop, and)

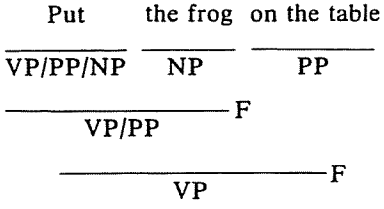


(b) (I cant believe I've eaten the whole thing, but)



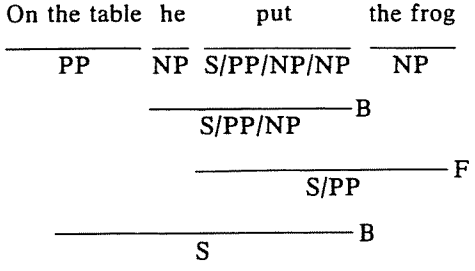
3.2.3. *Preposing from VP.* In Section 2.3 the category of the verb *put* was assumed to be VP/PP/NP, as in:

(34)

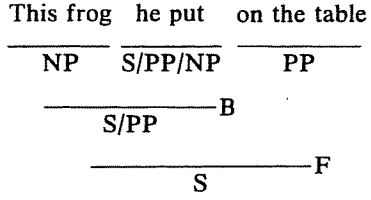


The combination of *put* with Tense by Affix Cancellation yields S/PP/NP/NP (22d). The model permits either the NP or the PP to be preposed:

(35) (a)

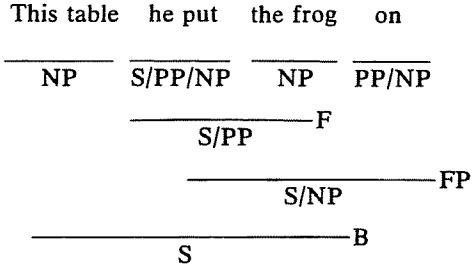


(b)



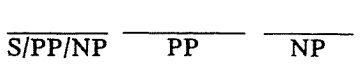
The model also allows fronting from within a PP ('Preposition Stranding', a topic which will be further discussed below):

(36)



Finally, the model rules out any sentence where the PP occurs before the NP Object (except when the PP is topicalized to sentence-initial position)[15]:

(37)



4. The Left Branch Condition and the Root Transformation Hypothesis

4.1. *Removing Overgeneralizations and Distinguishing Clause Functions* The account presented above is intended to suggest that the basic facts of English main clause constructions can be made to follow from just four very restricted rule schemata, together with an order-free categorial lexicon. However, there are a number of ways in which the model as it stands overgeneralises, and which might seem to cast doubt upon the claim. Its acceptance of **her must he love*, **love her must he*, and of sentences violating DO-support have already been mentioned (cf. Sections 3.1 and 3.2.1). A further class of overgeneralisations concerns sentences involving 'double topicalisations', such as the following, all of which are accepted by the model as it stands: