*Language and Cognitive Processes*, Vol. 1, No. 3, pp. 197–225 (1986)
© 1986 VNU Science Press.

# Grammars, parsers, and memory limitations

S. G. PULMAN
*University of Cambridge Computer Laboratory, Corn Exchange Street, Cambridge, UK*

Received June 1985; accepted in revised form January 1986

**Abstract**—Linguistic competence cannot be adequately characterized by grammatical devices of finite state power. Nevertheless, there are reasons to suspect that the human parsing device cannot adequately deal with languages that fall outside this class. This paper discusses these issues, arguing that restrictions on available parsing memory, and on our ability to operate properly when parsing recursive constructions, mean that there is an interesting sense in which human parsing resources must be characterized as finite state. This means that certain constructions regarded as grammatical according to a (richer than finite state) competence grammar, are not parsable, or are parsed in a way which is not in exact correspondence to their description by this grammar. This raises the further question of how these constructions can nevertheless be understood appropriately, given the assumption that semantic interpretation relies on syntactic structure. The paper goes on to describe an implemented computer program which embodies the claims made here about the nature of human parsing. It produces semantic interpretations (logical forms) incrementally on a left to right pass through a sentence. It uses only finite state resources while parsing a context-free (or richer) type of grammar. Although using the information provided by a competence grammar it does not build any explicit syntactic representations. Its behaviour when parsing recursive constructions is claimed to closely approximate our own.

## 1. BACKGROUND

In an early statement of the requirements of an adequate linguistic theory, Chomsky (1964, p. 120, see also 1959, p. 126) argued that such a theory should make available for a language $L(i)$ with grammar $G(i)$:

> . . . a function $g$ such that $g(i,n)$ is the description of a finite automation that takes . . . sentences . . . (of $L(i)$) . . . as input and gives structural descriptions assigned to those sentences by $G(i)$ . . . where $n$ is a parameter determining the capacity of the automaton.

This is a requirement that linguistic theory should provide some abstract account of how a grammar might be put to use in the production or comprehension of sentences, a demand which, if met, would take us 'one step closer to the actual use of language' (1961, p. 121)—a step towards a model of performance.

The requirement that the device specified by $g(i, n)$ be a finite automaton (finite state machine) of some sort arises from the simple observation that human memory and computational resources, while prodigous, must be finite. It is this fact that implies that, at some level, speaker–hearers can be modelled by a finite automaton, even though they may have internalized a (finitely stated) device capable of generating a language beyond the recognitional capacities of any finite automaton, and even though there may be nothing in their cognitive makeup with the literal internal architecture of a finite automaton:

> We have observed that such elementary formal properties of natural language as recursive nesting of dependencies made it impossible for them to be generated by finite automata, even though these properties do not exclude them from the class of context-free . . . languages. From these observations we must conclude that the *competence* of the

native speaker cannot be characterized by a finite automaton . . . Nevertheless, the *performance* of the speaker or hearer must be representable by a finite automaton of some sort. The speaker–hearer has only a finite memory, a part of which he uses to store the rules of his grammar (a set of rules for a device with unbounded memory) and a part of which he uses for computation in actually producing a sentence or 'perceiving' its structure and understanding it. (Chomsky, 1963, p. 390.)

Expressed in this way, the claim that the parsing abilities of speaker–hearers can be modelled in finite-state terms is fairly trivial: the performance of any cognitive ability by any mortal organism with finite powers is likewise guaranteed to be finite state in character. For example, suppose it turns out that human linguistic competence is characterised by some device of Turing machine power. A finite automaton augmented with two push-down stores is capable of parsing all the sentences generated by such a device (if they are capable of being parsed at all). The overall finite state character of such a combination is guaranteed if the depth of the stacks is restricted to some finite limit, say 100 million entries. Clearly, for all practical purposes, the claim that such a device is finite state is, while true, of no interest. Call this the 'trivial' version of the finite state claim.

A more interesting version of the finite state claim runs as follows. A *strict* finite state device for parsing or recognition has no memory available to it for storing the results of intermediate computations. Any device which does have such a memory, but where the size of the memory is fixed, is equivalent to some strict finite state device as far as the language generated is concerned. Unless otherwise qualified, when we refer to a 'finite state device' in what follows, it will be to a device of the latter type (and one, moreover, with a fairly small amount of fixed memory) rather than to the strict version.

The relevant property of finite state devices in this sense for our concerns is that the load on memory for such a device need not vary according to the length of the sentence being parsed. This is in contrast to richer devices, for example, an automaton enriched with an unlimited stack and thus capable of parsing a context-free language: if we regard the maximum depth of the stack as a measure of memory load, then characteristically for such devices this depth will vary according to the length of the sentence, longer sentences requiring, in general, a bigger stack to be parsed successfully. The interesting version of the finite state hypothesis maintains that this pattern of memory allocation is not an accurate characterization of the human parsing mechanism, and that the finite state description of a small and relatively fixed amount of memory independent of the length of the sentence more closely approximates what actually happens.

The 'interesting' finite state hypothesis is not a novel one. It seems clear that Chomsky had this in mind in framing the requirements we began with, rather than the trivial version (see Miller and Chomsky, 1963). More recently, Langendoen (1975), and Langendoen and Langsam (1984) have advanced the 'Efficient Finite State Parsing Hypothesis' that:

. . . natural languages are designed so that representations of the structural descriptions of acceptable expressions of the language can be assigned to them by finite state parsers. (Langendoen and Langsam, 1984.)

(which seems, incidentally, an odd way round to describe things). Other models have attempted to explain apparent properties of the human parsing system via memory limitations of one sort or another: Fodor *et al.*, (1974), Kimball (1973), Frazier and Fodor (1978), and Church (1980), among others. In some cases the constraints or strategies proposed have the effect of ensuring the finite-state-ness (in the interesting sense) of the overall system (Church is particularly explicit about this).

However, none of these authors have attempted to make such claims compatible with other requirements on a theory of parsing. At least two requirements are particularly relevant: (i) a theory of parsing should show how the process of semantic interpretation interacts with syntactic analysis; and (ii) a theory of parsing should show how the kind of online contextual resolution of linguistic ambiguity documented by, for example, Marslen-Wilson (1975), Tyler and Marslen-Wilson (1977) and Marslen-Wilson and Tyler (1980), is at least possible within the syntactic framework proposed.

In what follows, we will explore the various issues raised by the finite state parsing hypothesis and its interactions with these other two requirements, and in later sections present a description of an implemented parser-interpreter which resolves many of these questions and which seems to be a good initial candidate for the finite automaton described by Chomsky's function $g(i, n)$ in the quotation which began this section. Section 2 discusses the formal background concerning finite state and context free lànguages. Section 3 hypothesises that we are unable to parse any recursive constructions at all, entailing that parsing is finite state. Section 4 argues against some alternative approaches. Section 5 describes and illustrates a computer program which implements the analysis proposed here. Section 6 shows how the parser behaves appropriately when faced with left, right, and centre embeddings, and Section 7 summarizes and concludes.

## 2. CENTRE EMBEDDING, FINITE-STATE- AND CONTEXT-FREE-NESS

One of the earliest and most enduring of the observations in the generative grammar literature concerns the difficulty of centre embedding.[1] Under normal circumstances centre embedded structures of a degree more than two are extremely difficult to process, and even those of degree two are noticeably awkward.

(1a) The woman the man loved died
(b*) The woman the man the girl met loved died
(c**) The woman the man the girl the dog bit met loved died

Clearly other factors are also involved: Schlesinger (1968, cited in Fodor *et al.*, 1974) showed that where pragmatic information is available to help match up subjects to their appropriate verb, processing is easier. Nevertheless, multiple centre embedding is never as easy as comparable degrees of right and left embedding, and these intuitive judgements of acceptability are supported by the experimental findings of Miller and Isard (1964) which showed that difficulty of recall increases with depth of centre embedding, not with depth of embedding *per se*. Furthermore, non-recursive nesting, while it can get more difficult than left or right embedding, is still easier than recursive nesting:

(2) [If [John sent [the book that you borrowed] back to the library], then . . .]

Given that what distinguishes finite-state languages from context-free and richer languages is precisely that the former are unable to allow for unlimited centre embedding, whereas the latter are, the unacceptability of multiple centre embeddings in natural languages seems to fit nicely with the finite state parsing hypothesis. Furthermore, in (1959) Chomsky proved a theorem which showed that for a context-free grammar which only allows a small (finite) amount of centre embedding, it is possible to construct a finite-state transducer mapping from terminal strings of the language generated by the grammar to structural descriptions of the sentence. (A transducer is an abstract device mapping one language into another: here the first language is the language of the grammar, and the second a language in which its structural descriptions are couched.) The limit applies to centre embeddings only, and any amount of right or left embedding is allowed. Given the observation about the difficulty of centre embedding beyond a depth of three or four (say ten to be on the safe side), it seems to follow straightforwardly that with a theory of competence of context-free (or not much greater) power we are guaranteed to be able to construct a plausible parsing algorithm which is finite state in character.

Unfortunately, things turn out to be not quite so simple. Langendoen (1975) showed that the transducers in question do not yield quite the right amount of information to qualify them as full parsers for the context-free languages in question, even where centre embeddings are not concerned. In particular, in order to handle unlimited right and left embeddings correctly, the transducer itself needs further augmentation of a type which again enriches it to greater than finite state power.

Intuitively, this is easy to demonstrate: the formal characteristic of centre embedding which makes it non-finite-state is that it produces a sequence of constituents 'aaa . . .' which must be matched up one to one by some other sequence of constituents '. . . bbb'. The simplest example of such a language is $a^n b^n$ $(n \geqslant 1)$: some number of a's followed by the same number of b's. Right and left embeddings do not have this property, as far as the strings of terminal items generated is concerned, and so can be recognized by a strict finite state device.

However, when we consider not just the recognition of left and right embeddings, but transductions into their structural descriptions, this picture changes. At the very least, presumably, a structural description must be equivalent to a tree, or labelled bracketing indicating the constituent structure of the string of words that is input to the transducer. We can characterize this transduction abstractly, then, as a mapping from strings of words to strings of labelled left and right brackets. However, in this (putatively finite state) bracket language left and right labelled brackets are distinct symbols: '[*a*' and '*a*]' have to be explicitly paired up if these symbols are to have the intuitive interpretation we want them to have. Any form of recursion, then—left, right, or centre embedding—will produce in this bracket language 'sentences' of the form '. . . [*a*. .[*a*. .[*a*. .*a*]. .*a*]. .*a*]. .' where this sentence is well formed only if the left brackets pair up with the right brackets correctly. But this shows that the bracket language has in fact the non-finite-state character of the $a^n$, $b^n$ language considered earlier, and as a matter of definition we cannot have a finite state transduction into or out of a non-finite-state language. Thus even though a language considered as a set of strings might be finite state, an adequate characterisation of the constituent structures of those strings may well require non-finite-state resources.

Clearly this conclusion presents us with something of a problem. We are assuming that human parsing resources are finite state, but that competence is of context-free or richer power. These two facts comport well with the difficulty of centre embeddings, but are in apparent conflict with the fact that left and right embeddings of arbitrary depth present no problems in comprehension. Multiple possessive determiners:

(3) [[[[John's] friend's] mother's] dog]. .

right embedded relative clauses:

(4) [This is [the hand that signed [the treaty that stopped [the war that killed so many young men]]]]

multiple layers of sentential complements:

(5) John said [that he thought [that Bill would have believed [that Mary was guilty]]]

PP modifications of NPs:

(6) . . . a room [with [[a view [from the window]] [over [the river [near the meadows]]]]]

and such like constructions display no apparent difficulty in processing, although they all involve recursion.

## 3. NO RECURSION AT ALL

In their original discussion of centre embedding, Miller and Isard suggested that the unacceptability of multiple centre embeddings can be explained on the hypothesis that the language processing mechanism is organised in such a way that it contains subroutines corresponding to particular constructions or types of construction. These subroutines are called in the course of parsing a sentence whenever an instance of that construction is encountered. Suppose that such a subroutine has been called. Furthermore

. . . suppose that, while the subroutine is being executed, a second such construction is encountered, so that the subroutine is required to call itself. If this recursive feature were not available, confusion would result; the temporary memory for the point of re-entry into the main routine might be erased, for example, so that when it resumed, the main routine would have to treat subsequent words as if they began a new constituent of the sentence. (Miller and Isard, 1964, p. 300.)

This proposal is repeated and endorsed in Chomsky (1965, p. 14): I shall refer to it as the 'no recursion' hypothesis, on the understanding that no recursion whatsoever is the 'ideal' case and that in practice recursion to a depth of two or several more might be possible with appropriate aids or contextual help.

If the no recursion hypothesis can be sustained it provides us with one way of resolving our paradox. We simply claim that although there may be good linguistic reasons to analyse certain constructions as recursive, in practice, they are either unparseable (centre embeddings) or parsed in some way which avoids the difficulty imposed by recursion. Thus the claim that the human parsing device is finite state can be preserved simultaneously with the apparently unreconcilable claim that a correct description of competence involves grammatical devices of at least context-free power.

There are several further difficulties to be overcome if this account is to be made wholly plausible.[2] To begin with, if left and right embeddings cannot be (wholly) parsed in the normal way, we have no explanation of how it is possible for them to be understood correctly. In common with virtually all current work in linguistic theory, we assume that there is an intimate connection between syntactic structure and semantic interpretation, in particular that the process of arriving at an interpretation for a sentence on the basis of the interpretations of its parts is one that is guided by the syntactic structure of the sentence: that semantics is—to borrow a term from the theory of compiler design—'syntax-driven'. Thus even if we can argue successfully that multiple left and right recursions are only partially parsed in some way that would allow them to escape the difficulty of recursion, we would then have to provide some other account of how it was that, despite only getting a partial syntactic structure, they nevertheless seem to be capable of receiving a complete semantic interpretation. More will be said about this later.

Furthermore, there is something of a traditional lore in recent syntactic and parsing theory to the effect that human memory, or at least that part of it which is used in online syntactic processing, is organized in the form of a last-in, first-out push-down store device, or stack. This is suggested as a possibility by, among others, Woods (1973), Bach (1977), Fodor (1978), and is explicitly advocated by Ritchie (1980), and Maling and Zaenen (1982). The theories of sentence comprehension put forward by Ades and Steedman (1982), Steedman (1983) and Briscoe (1984) within the framework of extended categorial grammar likewise involve a stack as a crucial element of the parsing mechanism, albeit not in quite the same way as those envisaged by the other authors.

Clearly if these theories are well motivated it will become difficult to advance the no recursion hypothesis in any other than an *ad hoc* way, for a push-down store is the natural way to implement a device which is capable of recursive calls. Notice, incidentally, that although you cannot have recursion without something equivalent to a push-down store, the reverse is not true. It might be the case that no recursive applications of rules were ever permitted but that multiple nesting of different constructions was. In this case, it might be possible to argue for a push-down store organisation of memory independently of facts about recursion.

The major evidence for the push down store claim has been the set of phenomena associated with what has become known as the 'nested dependency constraint' (Fodor, 1978; Steedman, 1983). In transformational accounts, many constructions are described in terms of a movement rule displacing various constituents. The moved constituent is known as a 'filler', the position it moved from as a 'gap'. The nested dependency constraint prohibits associations of fillers with gaps (whether this association is regarded as syntactic or semantic) of the form:

(7) F1 F2 G1 G2

if it is possible instead to associate them on the pattern:

(8) F1 F2 G2 G1

thus the interpretation of

(9) Which boxes are these packets easy to put __ in __

follows the pattern of (8) rather than (7). Dependencies must nest rather than intersect or cross.

While these observations are suggestive, they are not compelling. Firstly, as Fodor (1978, p. 446) points out, we really need evidence from other areas of processing, for otherwise the hypothesis merely restates the data. There appears to be no such evidence from experimental studies (cf. Johnson–Laird, 1983, p. 332, who reaches similar conclusions). Secondly, there is the fact that the nested dependency constraint is not an absolute prohibition: many languages allow intersecting dependencies of various types: Norwegian and Danish (Maling and Zaenen, 1982); Turkish and Japanese (Kuno *et al.*, 1980, cited in Maling and Zaenen); Dutch (Bresnan *et al.*, 1982, Steedman 1983); and French (Steedman 1983). Even in English there are some examples apparently allowing intersecting syntactic dependencies which present no obvious difficulty:

10 [Which customer(1)] did you send the pictures _2 to _1 [that you developed yesterday(2)]

Of course, the fact (if it is a fact) that, syntactically at least, nested dependencies are the rule rather than the exception requires an explanation, but the fact that the exceptions seem to present no difficulty whatsoever means that this explanation cannot be in terms of a push-down store, and thus that this phenomenon provides no evidence in support of the claim that such a mode or organisation is an intrinsic or necessary part of the human parsing mechanism.

The no recursion hypothesis, when understood as above, applies indifferently to multiple dependencies whether they are nested or intersecting. The restriction is, recall, not on recursive calls of a routine, simply that if a routine is called more than '*n*' times then all but the information associated with the most recent '*n*' calls is lost. This will cover two kinds of case at least: (i) a routine is called(1) by some other routine, calls(2) itself and call(2) is intended to be completed before call(1); and (ii) a routine is called(1) by some other routine, calls(2) itself and has the facility to complete call(1) while in the course of completing call(2).

Both are recursive calls; only the first corresponds to the usual understanding of recursion, in fact, but our restriction will apply to both, and would strictly predict that on both types of pattern, the results of the first invocation will be inaccessible to the second (where $n = 0$).

In this context it is significant to note that not only, as predicted, are centre embeddings of a degree more than two unacceptable, but that the same is true of crossing or intersecting dependencies in those languages that permit them. For example, in Icelandic and Norwegian even three intersecting dependencies of the same type are not permitted; at most two filler-gap dependencies are allowed, and if a third crossing dependency is present it must be distinguished: one of the gaps must then be filled by a resumptive pronoun (Maling and Zaenen 1982, p. 237):

(11) F1 F2 F3 PRO1 G2 G3

(12*) F1 F2 F3 G1 G2 G3

In Dutch, it is possible to find sequences of the form

(13) . . . NP1 NP2 NP3. . . V1 V2 V3. . .

with dependencies as indicated. However, when more than two or three are involved, native speakers find the same difficulty as English speakers do with centre embeddings. Thus the recursion limitation applies equivalently to both types of construction.

The fact that the intersecting dependencies are possible at all demonstrates that the human parsing mechanism is not limited to a push-down stack type of organization, and the fact that the recursion limit applies in this case too shows that it cannot be regarded as a limit on stack depth. If the nested dependency constraint is real, then it seems most likely to reflect a feature of competence rather than of the parsing mechanism: the preference for nested rather than non-nested dependencies, for example, may well reflect the fact that the former are characteristic of context-free devices, whereas the correct treatment of the latter requires a richer type of grammatical mechanism. If the human language acquisition device rates these richer mechanisms as less natural, more difficult to acquire, then the observed preferences would receive some kind of explanation as a reflection of preferred types of grammar rather than of the mode of operation of the parser.

## 4. CENTRE EMBEDDINGS *ARE* GRAMMATICAL

Before turning to a description of the parser itself, let us try to dispose of the 'brute force' option. This is the position that since in practice, no more than, say, ten centre embeddings, and perhaps fifty right or left embeddings are ever likely to be encountered in a single sentence, we could construct a finite state parser adequate for all the examples we are ever likely to come across in real life.

This response in unsatisfactory for two reasons: firstly, it does not allow us to distinguish between centre embedding and other types of embedding in any principled way: the greater difficulty of centre embedding has no explanation, merely a description in terms of a different arbitrary limitation. Secondly, if the response were taken to be literally advocating a parser with the internal architecture of a finite automaton, there is the objection that the way such a device is naturally conceived of as functioning is by constructing new states and transitions for each new instance of a recursion (for otherwise left brackets could not be correctly matched with right brackets). But this would fail to capture the fact that the 45th recursion of the structure involving 50 right embedded relative clauses, is the 45th instance of the same construction, not the 45th in a series of 50 different constructions. This would be the claim implicit in such a treatment, since each time, new states and transitions would be involved. (This is not the only way that the construction of such an automaton could be approached, of course, and counters or equivalent devices might circumvent this objection: the first one still stands, however.)

A less version of this position is advocated by Reich (1969), who regards it as a fact about competence rather than performance that centre embeddings of a depth of three or more are not readily parsable. However, he regards the parseability of right and left embeddings as evidence that they are not recursive in structure, but iterative (a view for which there is some support from intonational evidence) and thus that syntactic competence can be entirely described by a grammar not exceeding finite state weak or strong generative capacity (1969, p. 271) but able to generate indefinitely many right and left 'embeddings' by means of iteration.

Reich's proposal claims that multiple centre embeddings are straightforwardly ungrammatical rather than unparsable. But this latter claim does not seem to square very comfortably with the observation that increased computational resources (time, or pencil and paper) or some period of deliberate practice can increase performance on these structures quite significantly. On the other hand, no amount of practice or artificial aids can make a clearly ungrammatical sentence like 'on mat cat the sat the' become grammatical. Furthermore, such a view has no explanation for deeper centre embeddings when they do occur: De Roeck *et al.* (1982) provide many examples of centre embeddings up to six deep in written German and English texts, and likewise argue that this supports the claim that these structures are grammatical but unparsable, rather than ungrammatical.

A more elaborate version of Reich's position is taken by Krauwer and Des Tombe (1981). They define a finite state transducer with a limited memory which enables it to parse unlimited right and left embeddings with no loss of grammatical information (given certain assumptions that they make explicit), but a limited number of centre embeddings, and a limited number of mixed right and left embeddings. However, like Reich, Krauwer and Des Tombe explicitly regard the transducer itself as a satisfactory representation of competence rather than as the description of a parsing procedure operating with limited performance resources on

a richer competence grammar. Thus their account falls prey to the same kind of objection as those advanced above against Reich's proposal.

Another related proposal would be to postulate a further element in our syntactic competence operating with the specific purpose of mitigating the effects of recursion in the grammar. This is the position taken by Langendoen (1975), who formulates a set of 'readjustment rules' operating at a late stage in the derivation of a sentence (in a transformational grammar, between syntax and phonology) to convert recursive structures into 'flat' ones capable of being parsed in an iterative, finite state, manner.

In Langendoen and Langsam (1984) a different approach is taken: they define an algorithm to produce from a context-free grammar a finite state transducer producing partial structural descriptions in the form of 'affixed strings': roughly, a version of the 'bracket language' described informally earlier. These transducers are virtually identical to the model advanced in Krauwer and Des Tombe (1981).[3]

In both of these approaches taken by Langendoen, the status of the various formal devices offered so as to make parsing finite state is difficult to interpret. Readjustment rules seem to be definitely an element of competence: the Affixed String transducer is presented as an abstract formal device for preprocessing grammars into finite-state parsable form, and its relationship to human syntactic parsing is not specified. [Elsewhere, (Langendoen and Postal 1984) Langendoen appears to reject the idea that grammatical devices are coherently assessable as psychological theories.]

However, it seems to me that there are fairly strong methodological objections to the claim that readjustment rules, if warranted, should be regarded as an element of competence, or that — if it is legitimate to interpret the later proposals in this way — human linguistic ability includes some device whose task is specifically to preprocess a competence grammar into a form in which it parsable by a finite state device. [These objections also apply to the claims made in Pulman (1983), where metarules, within a Generalized Phrase Structure Grammar framework (Gazdar, 1982) were proposed to capture the effect of Langendoen's treatment. This was the wrong way to look at the phenomenon.] For while data relevant to competence or performance do not come labelled as such in nature, it is fairly clear that the recursive nature of the constructions involved is established by the most straightforward tests of distribution and constituent structure analysis. Thus they have a strong prima facie case to be facts about competence, on any normal construal. To quote Harris (1954, p. 36) these structures really exist in the language 'as much as any scientific structure really obtains in the data which it describes' and thus is also a real property of the grammar describing those structures. The simplest realist assumption following from this is that these structures also really exist in speaker's minds: this is only to maintain that we believe our theory to be true (Chomsky, 1980). But we also want to maintain that the no recursion hypothesis is a true principle of human linguistic performance,[4] and a description of another aspect of speakers' ability (or disability). So if, as we hope to show later, the combination of these two properties is enough to account for all the data accounted for by readjustment rules, then on straightforward grounds of simplicity alone the former account is to be preferred. By keeping a simple account of competence and a simple principle of performance, we let them interact to account for what would otherwise be data demanding a fairly complex treatment.

There is another important drawback of all these approaches which becomes apparent when we consider the assumption mentioned earlier, that semantic interpretation is 'syntax-driven' — i.e. that a syntactic analysis of a sentence provides the necessary level of structure to be the input to the rules which operate compositionally to derive the meaning of a sentence from the meanings of its component words and phrases. For it is by no means obvious that the syntactic structures delivered by a readjustment rule approach, or a finite state description of competence, are such as to be able to coherently support a compositional semantics of the sort that this assumption demands. If the point of parsing a sentence is so as to be able to arrive at its interpretation, as is the unavoidable assumption, this makes the motivation for these devices as an element of either competence or performance very peculiar; what is the point of massaging something into an easily parseable form if the resulting parse trees cannot be interpreted properly?

## 5. WHAT SHOULD A THEORY OF PARSING ACCOUNT FOR?

Other than those we have already discussed, what kind of demands can we place on a theory of parsing, or equivalently, on a characterization of the function $g(i,n)$ described in the passage from Chomsky? At this stage, it seems, there are only a few constraining demands that can realistically be placed on the specification of such a theory. It is plausible to claim that it should have the following properties:

(i) It should provide all the syntactic structures associated with a grammatical input sentence by the grammar in question, (subject to any restrictions on this imposed by the parser itself). While it is clear that many syntactically possible but pragmatically absurd readings of a sentence are never consciously processed by a speaker–hearer, it does not seem to be the business of a syntactic (or semantic) algorithm to decide which of the possibilities present are appropriate in a particular context. Since there may be occasions on which structures which are pragmatically implausible on any normal assumptions may nevertheless be appropriate, we must assume that the parsing algorithm is capable of delivering all possible candidates, at least, at the level of abstraction at which we are viewing the parsing process. This means that we will not allow our description of the parser to be constrained by the necessity of accounting for observations about preferred readings, garden paths, and suchlike. Although it is an undoubted fact that these phenomena, which have been the source of many proposals to be found in the literature about the nature of human parsing, demonstrate something important about the nature of human linguistic comprehension, it is by no means clear that the *a priori* assumption of writers like Kimball (1973, 1975), Frazier and Fodor (1978), Marcus (1980), Church (1980), Ford (1982), Shieber (1983), Schubert (1984), (and many others) that all these phenomena are syntactic is correct. Many of the intuitive judgements on which these proposals are founded are extremely variable and sensitive to different choices of lexical material and changes of contextual assumptions, a fact which has been experimentally verified (Milne, 1982: Crain and Steedman, 1985). But a robust *syntactic* preference should not be affected by non-syntactic properties of a sentence to this extent, and it seems more likely that these phenomena represent the interaction of contextual or discourse effects on comprehension (including those of the so-called null context), as argued by Crain and Steedman (see also Briscoe, 1984; Pulman (forthcoming); and Altmann, 1985). If this is the case, then the claims by Marcus and, more recently, Berwick and Weinberg (1984) that parsing is deterministic do not follow (at least, not from these considerations), even if it may be the case that the overall process of comprehension is deterministic (Briscoe, 1984): and it certainly does not follow that it is the parser itself that determines these various preferences. In fact, once the possibility is acknowledged that such phenomena are non-syntactic, most of the arguments in favour of particular closure strategies, attachment preferences, and so on, need to be treated with a great deal of caution if construed as relevant to the characterization of the parsing mechanism *per se*. The position taken here is agnostic (but pessimistic) concerning the existence of such strategies. If they exist, they can be superimposed on the basic model proposed here without difficulty; if not, then the model is well equipped to deal with non-syntactic guidance of a parse.

(ii) It should operate from left to right processing each word at a time as fully as possible. Although it is probably true that in many circumstances hearers delay decisions or make anticipatory guesses about the input they are about to receive, these are, we must assume, heuristic strategies which can reasonably be regarded as overlaying the basic parsing procedure, employed in response to various non-linguistic factors. For at any point in a sentence a hearer can tell both what sort of sentence is permitted so far by the rules of his grammar at that point, and what sort is precluded, and also can tell — the complementary ability — what syntactic possibilities are still left open for the remainder of the sentence. This is evidenced by the fact that at any given point in a sentence a hearer can not only recognize ungrammatically (he does not have to wait until the end of the sentence) but can also supply possible and impossible sentence completions, guess at missing words, etc. All of this suggests that the parsing algorithm proceeds one word at a time, processing as fully as possible at each step, in the ideal case. (For experimental evidence bearing on this question, and the preceding point, see Marslen-Wilson and Tyler, 1980.)

(iii) It should make explicit how semantic interpretation is integrated with syntactic analysis in a way consistent with the observations in (i) and (ii), and with the assumption that a compositional semantics is driven by syntax. This is a deceptively modest requirement: it is not generally appreciated how difficult it is to reconcile this assumption with the preceding ones. Consider, for example, the natural way to write a computer program combining syntactic and semantic analysis in a syntax-driven way. Such a system might first parse a sentence, annotating nodes in the resulting tree with an indication of the syntactic rules used. This annotated tree would then be passed to an interpretation routine which applied the appropriate semantic operation to the topmost node (guided by the syntactic information found there, in particular a pointer to the semantic information necessary), calling itself recursively on each subtree to build up the complete interpretation. (Systems operating in more or less this manner are described in Rosenschein and Shieber, 1982; Gawron *et al.*, 1982; and Schubert and Pelletier, 1982.)

Two characteristics of such procedures are in conflict with the 'incremental interpretation' requirements: firstly, only complete constituents are capable of being interpreted, and secondly, interpretation cannot even begin until the end of the constituent is reached. A third characteristic is also troublesome, if we want to interpret such systems as psychological models (in however abstract a sense), for they require the computation of an explicit syntactic structure to precede the process of semantic interpretation. Little psycholinguistic evidence can be adduced, it seems, for the accuracy of such a picture of human parsing, and, as is well known, there is in fact precious little evidence that any purely syntactic level of representation is computed during comprehension at all, despite the obviously central role of syntactic information to the process.

Thus there were several interconnected aims in developing the (fully implemented[5]) parser–interpreter about to be described. The main aim was of course to build a formal model of parsing along the lines suggested by Chomsky's function $g(i,n)$, which embodied the claims made in earlier sections about the nature of parsing, and which was also faithful to the requirements of (i) to (iii). Another aim was to build a parser which satisfied a straightforward version of the 'competence hypothesis' (Chomsky, 1965, p. 9; Bresnan and Kaplan 1982, p. xvii). This program incorporates in a fairly literal form the assumption that grammars have some status, independently of parsers, as mental objects. That is to say, it was assumed that what linguists say about natural language in the form of a grammar (including semantic interpretation rules) is available to the parser–interpreter as some kind of data structure having roughly the form that the linguist's pencil and paper description would suggest, recursion and all.

A further aim was also to demonstrate a serious commitment to the 'incremental interpretation' requirements on a plausible model by getting the parser to build up explicit representations of the meaning of a sentence piece by piece during the course of a parse. Again, this is a harder requirement than is generally realized, given the considerations discussed above, and while most proposals acknowledge its desirability in principle, there are few concrete suggestions as to how it might be done. To my knowledge, the only other work which takes this commitment seriously at the appropriate level of formal detail is that of Ades and Steedman (1982). [Although they are working within an extended categorial grammar framework, and thus their assumptions about the nature of grammatical description and its relationship to parsing are very different, there are some interesting similarities between the structures built up during parsing in both models. In Pulman (forthcoming), I discuss in some detail the similarities and differences between these two approaches.]

For purposes of illustration, I will assume that the underlying grammatical theory involved is some form of Phrase Structure Grammar, where semantic interpreta tion consists of translation, on a rule by rule basis, into a simple form of higher-order logic, derived from the system of Montague's PTQ (Montague, 1973). Neither of these assumptions is crucial: the parsing procedure can be adapted to certain types of transformational grammar, and the associated process of semantic interpretation requires only that the semantic theory can be driven by syntactic structures, and that there is some way of doing function application and composition, or their equivalents. It is unlikely that this rules out any candidates at all.

The procedure is best thought of as a type of shift-reduce algorithm (Aho and Ullman, 1972), though with the ability to deal with incomplete constituents. This facet means that in its mode of operation it is similar to the general family of 'left corner' parsers discussed and illustrated to Johnson-Laird (1983). In the current implementation it operates non-deterministically, finding all possible parses of a sentence with respect to a particular grammar, subject to the limitations on recursion. This is not a realistic feature of the model: if I understood how such a function could be implemented, the main loop of the parser would include a call to a routine which whenever two or more possible parses were encountered, decided between them on the basis of contextual and, presumably, encyclopedic information. Then all but one parse would be discarded or downgraded, and only the most

preferred one pursued.

The driving mechanism of the parser–interpreter maintains an agenda of 'configurations', each representing a particular state of a parse. Within each configuration, complete and incomplete constituents are built on what, somewhat misleadingly, (though honouring tradition), is referred to as a 'stack'. However, although it is referred to as a stack, it is *not* a stack: at any time, more than the topmost item is accessible, and on occasions it is possible to access the bottom most items too. The fact that in general this list behaves like a stack is a reflection of the grammatical operations presupposed, as discussed earlier: different types of grammatical operation would produce different behaviour.

A configuration is a pair consisting of a representation of the state of the stack, and the current position in the input string. The stack is a list of 'entries', each entry representing a wholly or partially recognized constituent, along with its interpretation in terms of a translation into a logical expression. An entry is thus in fact a triple:

⟨category, needed, interpretation⟩

consisting of a category label, indicating what type of constituent is being recognized, a 'needed' list of constituents which must be found before the category is complete, and the interpretation built for the constituent so far. The parser starts off in the initial state, given a sentence, with an empty initial configuration pointing to the first word of the input, and proceeds by trying to produce new ones from that using the basic operations, until either no more alternatives are left, and the parse has failed, or one or more complete parses are produced at the end of the input string.

There are four basic operations which produce a new configuration from an old one. Which one is performed depends on the state of the stack, which is continually checked by the 'main loop' of the mechanism. If there is a choice between two operations, both being eligible to apply, then both are performed, producing two new configurations, possibly leading to two different parses. (This is the point at which the routine described above would operate.)

'Shift' takes the next word from the input and creates a new stack entry for it (for each lexical entry it has in the dictionary). For example, given a lexical entry like:

every: Lexical Category=Det, Interpretation=$\lambda P\lambda Q$(for-all x.Px → Qx)

Shift produces a stack entry like:

⟨Det, nil, $\lambda P\lambda Q$(for-all x.Px → Qx)⟩

Since lexical categories are always complete the second 'needed' element in a stack entry created by Shift will always be empty. Having created a new stack entry, Shift records a new configuration with that entry on top of the stack, and an input pointer updated to the succeeding word in the input sentence. (In what follows, the interpretation of non-logical words is assumed to be the associated constant, as is customary: thus the interpretation of 'man' will be represented as 'MAN'.)

'Invoke-rule' can only apply when there is a completed entry on top of the stack, representing a constituent that has been completely processed and fully interpreted. Essentially, it checks the rules in the grammar to see whether the category represented by that entry could begin some higher level constituent. (A one-word lookahead is incorporated for efficiency in the actual implementation. Within the Phrase Structure Grammar framework this is sufficient to rule out the vast majority of 'false starts' or temporary ambiguities that a naive parsing algorithm encounters, most notably those concerning conjunction and optional modifiers. This suggests that at least this modest degree of lookahead may be available to the human parsing mechanism, but I have not explored the question in any detail.)

If Invoke-rule succeeds in matching a category of an entry with the first member of the right hand side of a rule, and the rule is consistent with the lookahead word, it creates a new entry from them. Logically speaking, this process happens as follows: assume, for illustration, an entry of the form:

⟨Det, nil, EVERY⟩

(where the interpretation of 'every' might actually be as above) and a example rule of the form:

NP → Det N; DET (N)

where the part after the semi-colon is the semantic component, stating that the meaning of the whole NP is derived by applying the meaning of the determiner to the meaning of the noun. The Det entry matches the beginning of the right-hand side of the rule and so could begin an NP constituent. Now assume a function, call it Abstract, which when applied to a rule of this form produces from its right hand side and semantic component the result of lambda-abstracting over all the right hand side symbols (in the order specified in the rule) which appear in the semantic component. Thus Abstract applied to the rule above would produce:

λdetλn [det (n)]

If applied to a rule like:

S → NP VP; VP(NP)

it would produce

λnpλvp [ vp (np)]

This is simply a more literal rendering of what the rule actually says, in fact: making explicit the fact that the items occurring in the semantic part of the rule are to be interpreted as variables — the meaning of the sentence is found by taking the meanings of the NP and the VP (parsed in that order) and applying the latter to the former as function to argument.

When Invoke-rule has matched an entry to a rule it produces a new entry where the category is the left hand side of the rule, the 'needed' list is all but the first of the right-hand side, and the interpretation is the result of applying Abstract to the rule, and then applying the expression resulting from this operation, as a function (which it will always be), to the interpretation of the original entry. In the example above the result of all this would be:

⟨NP, N, λn[ EVERY (n)]⟩

In other words, the new interpretation is simply that of the whole rule with that of the existing entry put in the appropriate place. In general, the interpretation of an incomplete constituent is that it is a function expecting to find the needed items as arguments: there is in this respect a parallelism between the needed field and the interpretation, the latter being in effect a semantic version of the former.

'Combine', as you might expect, combines a complete entry on top of the stack with an incomplete one below it, if the category label of the former matches the first 'needed' item of the latter. For example, if the stack contained an entry like the one just described with a complete entry on top:

⟨N, nil, MAN⟩
⟨NP, N, λn [EVERY (n)]⟩

then Combine would produce a new entry with the category of the incomplete one, the remainder, if any, of the needed list, and an interpretation which is the result of applying that of the incomplete entry to that of the complete one. Here the result would be:

⟨NP, nil, EVERY (MAN)⟩

when beta-reduction (lambda conversion) of the lambda expressions has taken place. The result of this operation is a complete constituent, in this instance, although this need not be the case. If the interpretation of the bottom-most item had been a function expecting, say, two arguments, the new interpretation would be another function still expecting one argument. In general, if the needed field is not nil, the new interpretation will always reflect this.

These three operations are in fact sufficient to allow the parser to operate as a practical parsing device. If it is run as stated so far, it will derive all possible parses allowed for a sentence by a particular grammar, and will deliver the semantic interpretations corresponding to them. Other than the fact that these interpretations are built up incrementally, the parser operates in most respects like the systems we referred to above: it is a full context-free parser, and will handle left, right, and centre-embeddings to arbitrary depth, stacking up constituents in a way which is extremely profligate of memory allocation, consuming space which grows in proportion to the length of the input sentence.

A further operation is thus also necessary if we are to maintain consistency with our original assumptions. If our model is to be finite state in the sense intended, then some mechanism must be deployed to restrict the amount of syntactic memory used to some relatively constant level. 'Clear' is intended to do this, and to correspond to the intuition that once a complete or completable representation of a proposition has been built up, the syntactic information needed to do this is no longer required, under normal circumstances. For example, if in parsing a sentence we know that we have begun on the final constituent, we can be sure that, if this is grammatical, the whole thing will be grammatical: the items we have already parsed cannot affect the outcome at this point. Thus the syntactic information associated with the earlier stage of parsing of the sentence is no longer necessary.

Clear operates when: (i) the stack grows beyond some limit (determined by various factors); (ii) two adjacent incomplete items on the stack are projections of V (i.e. VP or S); and (iii) the top one of the two potentially contains everything needed to complete the one underneath it.

When the conditions for Clear are met, the effect is that the interpretation of the bottommost entry is composed with that of the one above it, the bottom one then being erased completely. For example, in a situation like:

-top-
. . .
. . .
⟨VP, S, λs [HOPES (s)]⟩
⟨S, VP, λvp [vp(SOME(MAN))]⟩
-bottom-

where the next-to-bottom entry is of the type that the one underneath is looking for, the result of Clear is that the stack will contain:

-top-
. . .
. . .
⟨S,S,λx [λvp {vp (SOME (MAN))} [λs {HOPES (s)}(x)]]⟩
-bottom-

When this 'S' eventually finds the S it is looking for, the interpretation will reduce to what we would have had more directly if Clear had not operated.

The first condition represents the claim that Clear is, ultimately, motivated by short term memory restrictions: when a certain threshold is exceeded, Clear operates to remove purely syntactic information. Notice that the intended interpretation of Clear is as a purely syntactic limitation: it is obvious that no such limitation at the short term level applies to semantic memory. This is modelled here by the fact that Clear guarantees that the amount of syntactic information that will appear on the stack at any point in a parse is limited by whatever the threshold operating is, whereas the semantic interpretations which may be found under the same circumstances can grow to an arbitrary size.[6]

The second condition is intended to capture the intuition that it is the main predicate of a sentence which when encountered provides enough information to be able to continue parsing safely after that point with no reference to anything before. When a verb is encountered, the number and type of (obligatory) arguments that should appear in its complement will be known. (Some complications to this clause are necessary to deal with adjectives and nouns which subcategorize for particular complements, but these details will be ignored here.) In common with many recent proposals, it is assumed that all optional (post) modifiers are introduced by left-recursive rules: NP → NP PP, VP → VP PP, etc.

The third condition above corresponds to the obvious truth that you can only get rid of syntactic information when it is safe to do so, and that 'selective forgetting' is not possible: either all the syntactic information relevant to the earlier portion of the sentence is discarded, or none of it is. Otherwise, the claim, and the later explanations which depend on it, would be vacuous.

The exact nature of the threshold for Clear is quite complex: presumably in a realistic model it would reflect many different factors, both linguistic and non-linguistic, which can affect transient memory load capabilities. Here, we simply capture some of this by a crude limit on the number of stack entries that can appear at a time: a limit of something like six seems to get things about right in practice. What 'about right' means will emerge below.

Here is a trace of the parser to show how all these operations work together. The meanings of the individual lexical items have been suppressed in the interests of readability. For simplicity, we will assume that Clear can operate when there are just two items on the stack. It is important to stress that under the assumptions just sketched, Clear would *not* operate in parsing such a simple sentence.

Sample grammar:

S → NP VP ; VP (NP)
VP → V NP ; V (NP)
NP → Det N ; Det (N)

Input: the cat caught a mouse

Shift:
⟨Det, nil, THE⟩

Invoke:
⟨NP, N, λn [THE (n)]⟩

Shift:
⟨N, nil, CAT⟩
⟨NP, N, λn [THE (n)]⟩

Combine:
⟨NP, N, THE(CAT)⟩

Invoke:
⟨S, VP, λvp [ vp (THE (CAT))]⟩

Shift:
⟨V, nil, CAUGHT⟩
⟨S, VP, λvp [ vp (THE (CAT))]⟩

Invoke:
⟨VP, NP, λnp [CAUGHT (np)]⟩
⟨S, VP, λvp [vp (THE (CAT))]⟩

Clear:
⟨S, NP, λx [λvp {vp (THE (CAT))}[λnp {CAUGHT (np)} (x)]]⟩

Shift:
⟨Det, nil, A⟩
⟨S, NP, λx [λvp {vp (THE (CAT))}[λnp {CAUGHT (np)} (x)]]⟩

Invoke:
⟨NP, N, λn [A (n)]⟩
⟨S, NP, λx[λvp {vp (THE (CAT))}[λnp {CAUGHT (np)} (x)]]⟩

Shift:
⟨N, nil, MOUSE⟩
⟨NP, N, λn[A (n)]⟩
⟨S, NP, λx[λvp{vp (THE (CAT))}[λnp {CAUGHT (np)} (x)]]⟩

Combine:
⟨NP, nil, A(MOUSE)⟩
⟨S, NP, λx[λvp{vp(THE (CAT))}[λnp{CAUGHT(np)}(x)]]⟩

Combine:
⟨S, nil, λx[λvp{vp(THE (CAT))}[λnp{CAUGHT(np)}(x)]](A MOUSE))⟩

At this point the parse is complete, and the complex interpretation beta-reduces to:
[CAUGHT (A (MOUSE))](THE (CAT))

The resulting interpretation is exactly what would have been obtained by a 'classical' system operating as described earlier.

## 6. PARSING RECURSIVE CONSTRUCTIONS NON-RECURSIVELY

It should be clear that our procedure provides a direct model of the process of incremental interpretation on the assumption that at least a central part of the meaning of a sentence is given by a translation into a logical form of this kind. As soon as a word is encountered on a left to right pass through an input sentence, it is integrated into the logical form being built up. At every stage, this logical form contains as much information about the meaning of the sentence so far as can be derived from what has been processed. It should also be clear that the partial interpretations, though of course not yet complete and thus not yet necessarily expressing a complete proposition, are perfectly meaningful objects both intuitively and formally speaking (within the higher-order logic assumed here, functions are just terms like any other): they can be used to perform inferences, be the antecedent for anaphora or ellipsis, be integrated with the context so as to assess and discard alternative interpretations corresponding to different parsings, and in general perform any of the functions we expect the meaning of a sentence or sentence fragment to be able to do.

The procedure also remains faithful to the assumption of the compositional, syntax-driven nature of semantics. The parser assumes that semantic information can be associated with syntactic rules in some way [though it is not ruled out that some extra aspects of interpretation may need to be computed by separate procedures: for example, identification of variables for the purposes of indicating coreference; cases of wide scope of quantifier phrases in syntactic narrow scope positions, etc. See Cooper (1983) for relevant discussion]. Once the rule in question has been identified by Invoke-rule, the semantic information involved is extracted and used to form the next stack entry. The syntactic information is also used to form expectations about what constituents must come next, (although it is conceivable that if semantic type is entirely predictable from syntactic category and vice versa this information is actually redundant). No other mechanisms for linking syntax with semantics are required, at least as far as these basic elements of the interpretation of a sentence are concerned.

An important thing to notice is that all of this is achieved without building any explicit syntax trees during the course of parsing a sentence. Syntactic information is of course used to build up the interpretation and to guide the parse, but this does not result in the construction of an independent syntactic level of representation. While it is true that in some sense trees are implicit in the sequence of operations of the parser, this is an inevitable consequence of the fact that the rules used themselves define trees, and as we shall see, even in this weak sense the tree structures implicit for certain types of recursive construction are not isomorphic to those which would be defined by the grammar.

One way of describing this aspect of the operation of the parser is as embodying the intuition often expressed (most often in the oral tradition, though explicit in Isard, 1974), that syntax is a 'control structure' for semantics. That is to say, the role of syntax in the process of language comprehension is as a set of tacit instructions for building representations of meaning, rather than as a level of representation with construction procedures of its own. The truth of this is of course ultimately a question for experimental investigation, but this way of looking at syntax has the merit of being consistent both with the widespread agreement among linguists that syntax plays a central role in language understanding, and with the already mentioned failure of psycholinguists to find any very strong evidence that purely syntactic representations (of any richness) are computed at any stage during normal comprehension.

We argued earlier, following Miller and Isard (1964), that the human parsing mechanism is fundamentally incapable of operating recursively. To be more precise: if (in the worst case) the parser encounters an instance of a construction in the course of trying to parse an earlier instance of it, the record of the earlier instance will be erased and 'forgotten', causing confusion in those cases where the information is needed to complete a parse successfully, as in the centre embedding cases. Clearly this is not absolute: some instances of centre embedding can be found to a depth of 4 or 5, but for simplicity we will assume that there is some small fixed limit, L.

The present procedure implements the no recursion restriction quite literally: if Invoke-rule attempts to put on the stack an incomplete constituent of category X when there are already L instances of such incomplete Xs on the stack, then the earliest instance is erased before Invoke-rule can succeed. The interesting and striking thing about this restriction is that as stated, it applies to all types of recursion, and thus might be expected to result in parsing failures not just for centre embedded examples of a depth greater than L, but for left and right recursions deeper than L too. However, this does not happen: the basic operations of the parser in fact conspire to bring it about that both left and right recursions can be parsed, the former fully, and the latter to just the extent, apparently, that is needed to be able to provide them with an appropriate interpretation. Thus a perfectly general and simple restriction can be imposed, rather than some version qualified so as to distinguish between different types of recursion. Furthermore, this is achieved without any need to change the original form of the grammar in any way: no preprocessing, readjustment rules, or conversion to transducers is necessary.

The simplest case is that of left recursion, which we will illustrate with an artificial example grammar:

A → A a; A(a)
A → a; a

When processing a string 'aaa. . .', the parser operates as in the following trace ('B' is the interpretation of 'a'):

Shift:
⟨a, nil, B⟩

Invoke:
⟨A, nil, B⟩

Invoke:
⟨A, a, λa ⟨B(a)]⟩

Shift:
⟨a, nil, B⟩
⟨A, a, λa [B(a)]⟩

Combine:
⟨A, nil, B(B)⟩

Invoke:
⟨A, a, λa [B(B(a))]⟩

At this point it is evident that we have entered a cycle of operations: Invoke begins a new A constituent, having just completed one; that is combined with a following 'a' to form another A, and so on. At no point in this cycle is there ever more than one occurrence of an incomplete A constituent on the stack, and so there is never any situation in which the recursion limitation would come into effect. In other words, (like any left corner mechanism), this parser can process unbounded left recursion without the stack growing beyond a constant depth.

Centre embeddings of a depth greater than L will not be parsed correctly by this procedure, however. To see how this works out in detail we will assume these simple rules (derived from Gazdar, 1982) for relatives:

NP → NP REL
REL → NP VP
REL → (NP) S/NP

S/NP → NP VP/NP
VP/NP → V NP/NP
NP/NP → trace

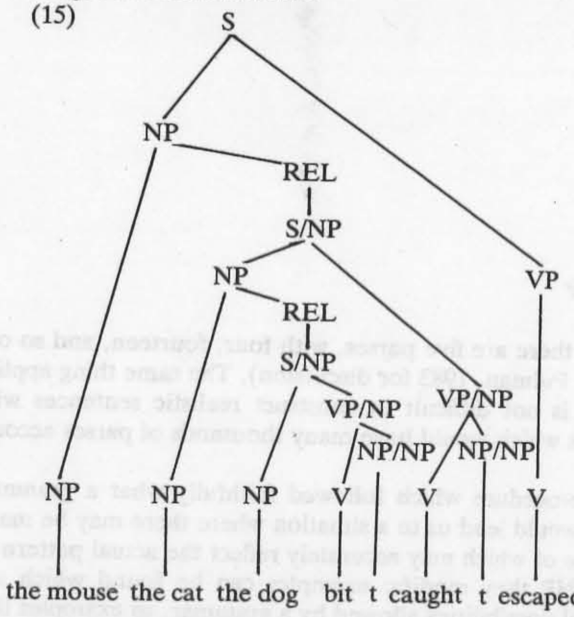
We will make several simplifying assumptions at this point, in the interests of clarity of exposition. Firstly, notice that we are assuming that relative clauses are a distinct constituent from S. This is a distinction made in some but not all linguistic theories: it is not a crucial one for our purposes. Secondly, we will assume that no recursion at all is allowed. Thus no more than one instance of a particular incomplete constituent can appear on the stack at any one time. (However, notice that what distinguishes constituents from each other in this respect is the rule that builds them, not the category label 'NP', etc., which may be shared. In this simple grammatical framework, each rule defines a different type of constituent for the purposes of the recursion limitation.) Finally, rather than build the incremental semantic interpretations yielded by the parser we will display the partial tree (in labelled bracket form) that a more conventional parser might give: the point can be made just as easily this way, and trees are considerably more 'human-readable' than the complex lambda expressions actually built by the parser.

For a sentence like:

(14) The mouse the cat the dog bit caught escaped

we ought to build a tree like:

(15)



Things proceed as follows, ignoring some obvious steps:

(i) ⟨NP, nil, [NP the mouse]⟩
(ii) ⟨NP, REL, [NP[NP the mouse][REL. . .]]⟩
(iii) ⟨NP, nil, [NP the cat]⟩
⟨NP, REL, [NP[NP the mouse][REL. . .]]⟩

At this point, if we are to find the correct interpretation or build the appropriate parse tree Invoke must recognize the NP 'the cat' as the beginning of another relative clause, and place on the stack an entry like:

⟨NP, REL, [NP[NP the cat][REL. . .]]⟩

But of course this will violate the recursion restriction, for there is already an [NP, REL. . .] on the stack. Let us assume that this earlier one is thus erased, or at least rendered inaccessible to the parsing procedure in some way. Things now proceed — again ignoring obvious details — until we have recognized the sentence as far as the word 'bit':

(iv) ⟨NP, nil, [NP[NP the cat][REL[S/NP[NP the dog][VP/NP bit]]]]⟩

At this point the procedure runs into trouble. If the parser merely continues with 'caught' it will be stuck: 'caught' in its own is not a complete VP, for it is missing an object. So a possible parse in which what is on the stack is the subject of 'caught' will fail. But there is no other option available for it. In order to treat 'caught' correctly, the parser needs to know not only that it is a VP/NP, which it will be able to do, but also that this constituent is of part of a REL, and thus can legitimately have a missing object. But this of course is precisely the information that is no longer available to it, for the REL entry which would have signalled this has been erased. Thus no S/NP constituent is predicted, and the VP/NP 'caught' cannot be integrated with the rest of the sentence. (The reader is invited to check that with the basic operations available to it, and the grammatical rules given, the parser cannot proceed beyond this point coherently.)

It is reassuring to remember that it is exactly at this point — after the first verb of the sequence stacked up — where both intuitive and experimental evidence (Miller and Isard, 1964) suggest the onset of difficulty with these constructions. Our parsing procedure seems to run into trouble at exactly the same point that people do when they are processing these centre embedded constructions. It is interesting too that the model as described will apparently parse sentences like:

The cat that the rat that the dog bit squealed

although not being able to assign a fully coherent interpretation to it. We should therefore hope to find that speakers likewise do not experience as much of a *syntactic* problem with such sentences. Intuitively, this is so: hearers are not conscious, when the sentence is first heard, of anything like the degree of difficulty of a sentence like (32), and it is often only when they stop to reflect that the incomplete nature of the sentence becomes apparent to them. It would be interesting to see if this intuition can be experimentally supported.

Turning now to right recursion, there are two cases of interest. With multiple sentential complementation like

(16) Joe thought that Bill expected that Mary knew . . .

we might expect that exactly the same thing would happen as in the centre embedding cases. At the point at which the deepest sentences are encountered, the earliest S constituents are still incomplete, and thus the recursion restriction ought to mean that they are erased from the stack. However, this is where the motivation for the Clear operation will become apparent, for the operation of Clear means that the recursion limit may be avoided in these constructions. We will illustrate this again with the simplifying assumption that no recursion at all is permitted, and that Clear operates when there are two appropriate items on the stack. Thus whenever we have a stack of the form:

⟨VP, S, β⟩
⟨S, VP, α⟩

Clear will erase the bottom entry leaving:

⟨S, S, λx[α{β(x)}]⟩

Whenever there is a stack of the form:
⟨S, VP, β⟩
⟨VP, S, α⟩
Clear will likewise produce:
⟨VP, VP, λx[α{β(x)}]⟩
Thus neither recursive category will ever have more than one instance on the stack at a time. As in the earlier illustrative examples, the process of function composition means that, when the final constituent is encountered, the whole complex logical expression reduces down to exactly what we would have had under the 'classical' view: the difference here is that we do not depend on the whole syntactic tree being explicitly constructed first in order to get the correct results. Notice, incidentally, that this final process of beta-reduction is not part of the process of assembling the meaning: the meaning is fully assembled already, even in the form of a complex unreduced lambda expression. The reduced form is merely a more easily readable logically equivalent version.

There are several qualifications which have to made to the above simplified picture to arrive at a more satisfactory general account. Since the motivation for Clear is partly via considerations of short-term memory load, in a more realistic model some extra parameter to reflect this transient load should clearly be involved such that Clear only operates when a certain threshold is exceeded, and where this threshold reflects something not limited to a crude count of stack entries or any similar measure, but can take into account non-syntactic constraints on memory and computation.

A second respect in which more detail needs to be added concerns the connection between the conditions on Clear, and the limitation on recursion. The current model implies that they are connected, in the sense that with a recursion limit of one, even a sentence like

(17) John expected that Bill would leave

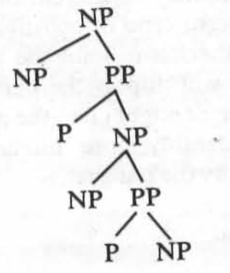
could not be parsed unless Clear had operated, thus implying that Clear should in reality operate under the unrealistically severe conditions we have been assuming for illustration. A recursion limit of one is in fact a reasonable first approximation to the behaviour of speakers when faced with centre embeddings (under the analysis assumed above — obviously the limit has to be stated differently if REL is identified with S), but it seems unlikely that Clear is so restricted. It does not seem very plausible to maintain that such a short sentence as (17) imposes any very great strain on syntactic short-term memory, and even if were to try to endorse such a claim it would be faced with unwelcome empirical consequences elsewhere: if operating under such a severe restriction Clear would actually prevent VP conjunctions from being parsed at all, for by the time the conjunction is reached, the only constituent left on the stack is labelled as an S, not a VP, and so Invoke-rule cannot find an appropriate candidate to continue.

Perhaps a better way to think about it is to imagine Clear operating whenever it can, though delayed by some margin behind the input. Thus things lower down the stack would be Cleared, while constituents higher up need not be. This would mean that unless the first conjunct of a conjunction was so long and complex that Clearing had begun to operate within it, all conjuncts would be parsed normally. This is what the rough and ready limit on stack depth is intended to model: with a limit of about six, and a recursion limit of about three, the parser seems to produce about the right kind of behaviour (where Clearing takes place as low as possible on

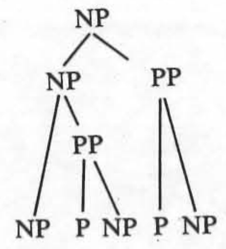
the stack): i.e. centre embeddings of degree two fill the stack, but can (just) be parsed; those of greater than two cannot be parsed appropriately; simple right embeddings like (17) are parsed without Clearing or the recursion limit being exceeded, but those of degree three or more trigger Clear before the recursion limit is reached. In the centre embedding cases, of course, the conditions for Clearing are never met.

Not all cases of right recursion need be 'rescued' by Clear, however. Consider the case of multiple PP or REL modifiers, the former introduced by rules like:
NP → NP PP
PP → P NP
Given a sequence of the form 'NP P NP P NP', there are two possible parses:

(18)



(19)



Given three PPs in a row there are five parses, with four, fourteen, and so on, (see Church and Patil, 1982; Pulman, 1983 for discussion). The same thing applies to conjunctions as well: it is not difficult to construct realistic sentences with multiple conjunctions or PPs which would have many thousands of parses according to rules like those above.

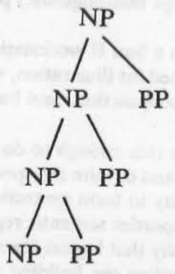
Clearly then, a parsing procedure which followed faithfully what a grammar might say about such cases would lead us to a situation where there may be many distinct parse trees, only one of which may accurately reflect the actual pattern of attachment of PPs to the NP they modify: examples can be found which are consistent with all the logical possibilities allowed by a grammar, as examples like these suggest:

(20a) The house in the woods by the river
(b) The book on rock climbing by the writer from Scotland
(c) The bird in the tree near the flowerbed with a red beak

It is clear that there are several non-syntactic factors affecting the parsing of optional modifiers like these. Intonation can force a particular attachment (this 'attachment' may not need to be made syntactically at all, of course), some syntactically possible attachments could be ruled out on pragmatic grounds, and so on. In the case where a definite attachment is not suggested by intonation we can think of the parsing model as 'consulting the context' whenever a choice arises as described earlier, making an attachment on this basis, and then continuing. Thus the combinatorial explosion suggested by the grammar will never actually happen, for each prepositional phrase (or relative clause) will only give rise to one parse, and the multiplicative effect of a string of modifiers will not arise.

It is interesting to point out, however, that even if the context was not consulted to decide on the correct attachment, the parser would still not deliver all the parses assigned to a string of modifiers by the grammar. In fact, assuming a recursion limit of one, there is only one 'parse' of such structures that will succeed. The parsing procedure will process these cases in a way which corresponds to a left branching or stacked analysis:

(21)



Centre embedded parsings of these (which will always be possible) would exceed the recursion limit, and so will right embedded parses, since Clear — applying only to projections of V, recall — cannot rescue the structures.

It has often been noted (Chomsky, 1965, p. 13ff.; Reich, 1969; Langendoen, 1975) that structures involving a longish string of modifiers like these, even where their interpretation is clearly that of a right embedded structure, typically have intonation contours that do not respect this syntactic structure. Multiple right embedded relative clauses appear to consist intonationally of a string of non-embedded sentences:

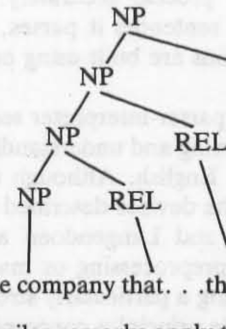
(22a) [This is the man][who helped me write that book][that nobody ever reads]
(b) [This is [the man [who helped me write [that book [that nobody ever reads]]]]]

as in (22a), rather than having a contour which groups words according to the syntactically appropriate bracketing in (22b). (This sentence is actually short enough to receive the 'correct' contour with some planning.) The breaks appear between the head NP of the relative and its modifying clause, thus splitting the constituent into two, intonationally. Similar effects can be found for sequences of PPs like those above.

In this context, it is surely significant that the stacked 'parse' which the operation of the procedure mimics is actually the one which corresponds exactly to these unexpected intonation patterns:

(23a) [That's the company][that manufactures the drugs][that have the side effects][that made her come out in a rash]

(b)



Similar remarks apply to the intonational peculiarities of the multiple possessive determiner examples: these determiners usually have something like a list intonation

(24) [John's][mother's][friend's]doctor. . .

rather than having a single contour over the whole NP, and are often felt to be 'awkward' in some intuitive way in this respect. Notice that here, the sequence of 'intonational' constituents corresponds exactly to the sequence of complete non-recursive 'Det' constituents built on top of the stack while parsing such examples.

The exact interpretation of the implications of these intonational facts for a model of parsing is rather complex, admittedly. Let us make the simplest assumption suggested by the competence hypothesis: that the structures over which intonation contours are computed during production are isomorphic to those implicit while parsing. If this is so, then the intonational facts are, as we have seen, consistent with our model, and provide no support for the iterative treatment of these constructions suggested by the authors discussed earlier.

The 'stacking' treatment of right recursive modifiers is of course affected by exactly how much recursion is possible (though the treatment of left recursion is unaffected by this). Even if several different paths are pursued, though, the incremental interpretation made available will enable the decision as to which is the appropriate path to pursue to be taken as soon as possible, thus allowing the others to be discarded. Thus either way the operation of the parsing procedure is faithful to the obvious fact that these constructions are on the whole completely unproblematic as far as human parsing is concerned.

## 7. CONCLUSIONS

We began with two hypotheses about the nature of human parsing: that it has the characteristics of a strict finite state device, and that it has particular difficulties with recursion. While related, these are separate hypotheses. We have also discussed various further requirements that can be placed on models of parsing, if they are to capture the most basic properties of human sentence comprehension: in particular the requirement that syntactic analysis be integrated with semantic and contextual interpretation in a way consistent with the observed ability of humans to understand sentences incrementally while hearing them. Something else that we have assumed, more as a methodological stance than anything, is the 'competence hypothesis' implicit in the quotations from Chomsky with which we began: that a competence grammar may be literally included with a model of performance.

We have described a computer program which captures all these properties.

Because of the 'Clear' operation, and the recursion restriction, it operates with strictly finite state resources, although using a grammar of richer power, and thus fails to parse certain types of construction correctly. These are exactly the constructions that people find difficult or impossible to process accurately. It provides, incrementally, semantic interpretations for the sentences it parses, to exactly the extent, we have argued, that such interpretations are built using only the information contained in a grammar.

The recursion limitation and the basic operations of the parser-interpreter seem to combine to provide a fairly satisfactory model of the parsing and understanding of the different types of recursive constructions found in English. Although the overall behaviour of the procedure approximates that of the devices described by Krauwer and Des Tombe (1981), Langendoen (1975), and Langendoen and Langsam (1984), it does so in a way which involves no preprocessing or manipulation of the grammatical formalism at all, thus embodying a particularly strong version of the competence hypothesis. In this model, simple principles concerning the nature of the human parsing device interact with a straightforward description of linguistic competence to produce an apparently accurate abstract model of some of the central phenomena of parsing performance. This is essentially the relation between grammar and parser implicit in Chomsky's description of the function $g(i,n)$, and the present model can therefore be taken as a first approximation to the characterization of that function.

## NOTES

Versions of parts of this paper have been given as talks at the Universities of Essex, Cambridge, Sussex, Reading, and York since about 1982 and an earlier version of the parser was described in a paper for the second European conference of the Association for Computational Linguistics which appears in the proceedings (Pulman, 1985). I am grateful to the audiences on these occasions for their comments and to the following people for their comments, criticisms and encouragement: Ted Briscoe, Eva Ejerhed, Gerald Gazdar, Steve Harlow, Steve Isard, David Reibel, Geoffrey Sampson, Aaron Sloman, Mark Steedman, Graeme Ritchie and Yorick Wilks. Phil Johson-Laird, Charles Clifton and an anonymous referee also made many valuable suggestions concerning both form and content.

1. The usual meanings of these terms is as follows: where X and Y are variables over non-terminal symbols, and $\alpha$ and $\beta$ are variables over strings of terminal and non-terminal symbols, we say that a grammar is 'nesting' if for some X and Y, $X = > \alpha Y\beta$, ($\alpha$ and $\beta$ non-null), (i.e. there is a derivation leading from X to $\alpha Y\beta$), and that it is 'self-embedding' if for some X, $X =>X\beta$. In the case where $\alpha$ is null we have 'left-embedding' (left recursion) and where $\beta$ is null we have 'right-embedding' (right recursion). Where neither are null we have 'centre-embedding'. (Note that in Chomsky (1963, pp. 394ff.) the term 'self-embedding' is taken to mean centre embedding only.)

2. Alternative explanations for centre embedding difficulties are advanced by, among others. Fodor *et al*. (1973), Kimball (1973), and Frazier and Fodor (1978). These proposals are discussed and criticised in Pulman (forthcoming).

3. Proposals similar in spirit are advanced by Ejerhed (1982), and Ejerhed and Church (1983), who advocate a model of syntax in which recursion has been eliminated, thus guaranteeing finite-state-ness, and in which some non-finite-state phenomena (nesting and intersecting dependencies) are handled semantically.

4. Although so far we have been concerned only with recursion in syntax, it seems reasonably plausible to assume that semantic recursion provides an equivalent degree of difficulty. To take a simple example, multiple negation, either explicit or implicit, seems very awkward to process when beyond a factor of two or three:

He doesn't not not like the picture
You can't fail not to dislike it

On the assumption that the semantic rule for negation operates on a whole proposition, then such multiple negatives, while arguably not involving syntactic recursion, would involve semantic recursion. Similarly, multiple comparatives are notoriously difficult to process accurately and easily. If (a) and (b) are grammatical, then in most frameworks (c) and (d) will also automatically be grammatical:

(a) This building is taller than that one
(b) This building is taller than it is wide
(c) This building is taller than it is wider than that one
(d) This building is taller than it is longer than it is wider than that one

(c) is grammatical and means something like 'the height of the former building exceeds the extent to which its width exceeds that of the latter' But the sentence is awkward, to say the least, and (d) requires some conscious effort to arrive at an appropriate reading. These examples involve only right embedding and so the difficulty cannot be traced to this. In a semantics for comparatives such as that presented in Klein (1982), however, the rule for interpreting such adjective phrases would have to be applied recursively in order to produce an appropriate interpretation. Interestingly, examples like (d) invite a conjunction type misanalysis 'taller and longer and wider than' exactly analogous to what is observed in the case of centre embeddings (Blumenthal, 1966): if the no recursion hypothesis applies at the semantic as well as the syntactic level then this parallelism is accounted for.

Recursion difficulties may also explain the well-known awkwardness of self-reference. Exactly the same phenomenon of recursive application seems to be involved here: evaluating the meaning, reference or truth of some expression involves referring to the expression currently being evaluated. Not all examples of self reference involve conscious difficulty of processing, but all evoke a stylistic effect, at the very least, as in the joke philosophy essay on 'Is this a question?' (you are supposed to write, 'Yes, if this is an answer'), but the difficulty most ordinary mortals encounter in even understanding, at first acquaintance, an expression like 'the set of all sets that are not members of themselves' suggests that the no recursion principle may apply here too, independently of any further conceptual difficulties associated with logical paradoxes. In sum, therefore, there seems to be some evidence for maintaining the no recursion principle as a general principle of linguistic (and perhaps non-linguistic) performance, rather than being specific to the syntactic processor.

5. The program is written in Franz Lisp under Berkeley 4.2 Unix on a Sun II workstation. It uses a grammar with syntactic and semantic coverage of all the constructions used for illustration, various types of 'wh' constructions, as well as some types of phrasal and sentential conjunction, and has access to a lexicon of about 3000 words.

6. Isn't this cheating? Aren't we merely transducing into a language rich enough to do what we are denying the parser the ability to do? Two points: we are, at this point (and despite the speculations in a preceding footnote) making claims only about restrictions on the ability to form syntactic representations (i.e. stack entries) only. This claim is unaffected by whatever properties semantic representations per se turn out to have. Secondly, it is in any case assumed provisionally that logical forms of the type built here are ultimately dispensable: they are merely a way of modelling the building up of a truth conditional interpretation via translation into a disambiguated language. The logical forms are proxies for the model-theoretic interpretations which can be read off from them, and as such are theoretically dispensable, and not a language or level of representation in their own right.

## REFERENCES

Ades, A.E. and Steedman, M.J. (1982). On the order of words. *Linguistics and Philosophy 4*, 517–558.

Aho, A.V. and Ullman, J.D. (1972). *The Theory of Parsing, Translation, and Compiling*, Vol I. Prentice-Hall, Englewood Cliffs, N.J.

Altmann, G. (1985), The resolution of local syntactic ambiguity by the human sentence processing mechanism. In: *Proceedings of the Second European meeting of the Association for Computational Linguistics*, ACL.

Bach, E. (1977). Comments on the paper by Chomsky. In: *Formal Syntax*, P.W. Culicover, T. Wasow and A. Akmajian (Eds). Academic Press, New York.

Berwick, R.C. and Weinberg, A.S. (1984). *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition*. MIT Press, Cambridge, Mass.

Blumenthal, A.L. (1966). Observations with self-embedded sentences. *Psychonomic Science 6*, 453–454.

Bresnan, J. (Ed.) (1982). *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge Mass.

Bresnan, J. and Kaplan, R. (1982). Introduction to Bresnan (Ed.) (1982).

Bresnan, J., Kaplan, R., Peters, S. and Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry 13*, 613–635.

Briscoe, E.J. (1984). Towards an understanding of spoken speech comprehension: the interactive determinism hypothesis. Ph.D. Diss., Dept of Linguistics, University of Cambridge.

Caplan, D. (1972). Clause boundaries and recognition latencies for words in sentences. *Perception and*

*Psychophysics 12*, 73–76.
Chomsky, N. (1959) On Certain formal properties of grammars. *Information and Control 2*, 136–167.
Chomsky, N. (1964). On the notion 'rule of grammar'. In: *The Structure of Language: Readings in the Philosophy of Language*, J.A. Fodor and J.J. Katz (Eds). Prentice Hall, N.J.
Chomsky, N. (1963). Formal properties of grammars. In: *Handbook of Mathematical Psychology*, Vol II, R. Luce, R. Bush and E. Galanter (Eds). John Wiley, New York.
Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass.
Chomsky, N. (1980). *Rules and Representations*. Basil Blackwell, Oxford.
Chomsky, N. and Miller, G. (1963). Finitary models of language users. In: *Handbook of Mathematical Psychology*, Vol II, R. Luce, R. Bush and E. Galanter (Eds). John Wiley, New York.
Church, K.W. (1980). On memory limitations in natural language processing. MIT Technical report (MIT/,LCS/TR-245)
Church, K.W. and Patil, R. (1982). Coping with syntactic ambiguity. *American Journal of Computational Linguistics 8*, 139–149.
Cooper, R. (1983). *Quantification and Syntactic Theory*. D. Reidel, Dordrecht:
Crain, S. and Steedman, M.J. (1985). On not being led up the garden path: the use of context by the psychological parser. In: *Natural Language Parsing: Psycholinguistic, Theoretical, and Computa tional Perspecitves*, A. Zwicky, L. Kartunnen, and D. Dowty (Eds). Cambridge University Press, Cambridge.
De Roeck, A. *et al.* (1982). A myth about centre-embedding. *Lingua 58*, 327–340.
Ejerhed, E. (1982). The processing of unbounded dependencies in Swedish. In: *Readings on Unbounded Dependencies in Scandinavian Languages: Umea Studies in the Humanities 43*, E. Engdahl and E. Ejerhed (Eds). University of Umea.
Ejerhed, E. and Church, K.W. (1983). Finite state parsing In: *Papers from the 7th Scandinavian Conference of Linguistics, Publication no 9*, F. Karlsson (Ed.) University of Helsinki, Dept of Linguistics, pp. 410–432.
Fodor J.A., Bever T.G. and Garret, M.F. (1974). *The Psychology of Language*. McGraw-Hill, New York.
Fodor, J.D. (1978). Parsing strategies and constraints on transformations. *Linguistic Inquiry 9*, 427–473.
Frazier, L. (1979). *On Comprehending Sentences: Syntactic Parsing Strategies*. Indiana University Linguistics Club, Bloomington, Ind.
Frazier, L. and Fodor J.D. (1978). The sausage machine: a new two stage parsing model. *Cognition 6*, 291–325.
Ford, M., Bresnan, J. and Kaplan, R.M. (1982). A competence-based theory of syntactic closure.In: *The Mental Representation of Grammatical Relations*, Bresnan, J. (Ed). MIT Press, Cambridge, Mass.
Gawron, J.M. *et al.* (1982). The GPSG linguistics system. *Proceedings of the 20th Annual Meeting*. Association for Computational Linguistics.
Gazdar, G. (1982), Phrase structure grammar. In: *The Nature of Syntactic Representation*. P. Jacobson and G. Pullum (Eds). D. Reidel, Dordrecht, pp. 131–186.
Gazdar, G., Klein, E., Pullum, G. and Sag, I. (1985). *Generalised Phrase Structure Grammar*. Basil Blackewll, Oxford.
Harris, Z. (1954). Distributional structure. Reprinted in: *The Structure of Language*, J.J. Katz and J.A. Fodor (Eds). Prentice-Hall, N.J. (1964).
Isard, S. (1974). What would you have done if . . . .? *Theoretical Linguistics 1*, 233–256.
Johnson-Laird, P.N. (1983). *Mental Models*. Cambridge University Press, Cambridge.
Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition 2*, 15–47.
Kimball, J. (1975). Predictive analysis and over-the-top parsing. In: *Syntax and Semantics*, Vol. 4 J. Kimball (Ed.) Academic Press, New York.
Klein, E. (1982). The interpretation of adjectival comparatives. *Journal of Linguistics 18*, 113–136.
Krauwer, S. and Des Tombe, L. (1981). Transducers and grammars as theories of language. *Theoretical Linguistics 8*, 173–202.
Langendoen, D.T. (1975). Finite state parsing of phrase structure languages and the status of readjustment rules in grammar. *Linguistic Inquiry 6*, 533–554.
Langendoen, D.T. and Postal, P. (1984). *The Vastness of Natural Languages*. Basil Blackwell, Oxford.
Langendoen, D.T. and Langsam, Y. (1984). The representation of constituent structures for finite state parsing. In: *Proceedings of Coling 84*. Association for Computational Linguistics.
Maling, J. and Zaenen, A. (1982). Scandinavian extraction phenomena. In: Jacobson and Pullum (Eds).

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, Mass.
Marr, D. (1977). Artificial intelligence : a personal view. Reprinted in: *Mind Design*, J. Haugeland (Ed.) Bradford Books, MIT Press, Cambridge, Mass: pp. 129–142.
Marslen-Wilson, W.D. (1975) Sentence perception as an interactive parallel process. *Science 189*, 226–228.
Marslen-Wilson, W.D. and Tyler, L.K. (1980). The temporal structure of spoken language understanding. *Cognition 8*, 1–71.
Miller, G. and Isard, S.D. (1964). Free recall of self embedded English sentences. *Information and Control 7*, 292–303.
Milne, R.W. (1982). Predicting garden path sentences. *Cognitive Science 6*, 349–373.
Montague, R. (1973). The proper treatment of quantification in ordinary English. In: *Approaches to Natural Language*, J. Hintikka *et al.* (Eds). D. Reidel, Dordrecht.
Pulman, S.G. (1983). Generalised phrase structure grammar, Earley's algorithm, adn the minimisation of recursion. In: *Automatic Natural Language Parsing*, K. Sparck Jones and Y. Wilks (Eds). Ellis Horwood, Chichester.
Pulman, S. G. (1985). A parser that doesn't. *Proceedings of the Second European Meeting of the Association for Computational Linguistics*, ACL.
Pulman, S.G. (forthcoming). Computational models of parsing. In: *Progress in the Psychology of Language*, Vol. 2, A. Ellis (Ed). Lawrence Erlbaum.
Reich, P. (1969). The finiteness of natural language. Reprinted in: *Syntactic Theory I*, F. Householder (Ed.). Penguin, Harmondsworth.
Ritchie, G. (1980) *Computational Grammar*. Harvester Press, Hassocks.
Rosenschein, S.J. and Shieber, S.M. (1982). Translating English into logical form. *Proceedings of the 20th Annual Meeting of the, Association for Computational Linguistics*, ACL.
Schlesinger, I.M. (1968). *Sentence Structure and the Reading Process*. Mouton, The Hague.
Schubert, L.K. (1984). On Parsing Preferences. In: *Coling 84: Proceedings of the 10th International Conference on Computational Linguistics*. Computational Linguistics, Stanford, Calif.
Schubert, L.K. and Pelletier, F.J. (1982). From English to logic: context free computation of 'conventional' logical translation. *American Journal of Computational Linguistics 8*, 27–44.
Shieber, S. (1983). Sentence disambiguation by a shift-reduce parsing technique. *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, ACL.
Steedman, M.J. (1983). On the generality of the nested dependency constraint and the reason for an exception in Dutch. *Linguistics 21–1*, 35–66.
Tyler, L.K. and Marslen-Wilson, W.D. (1977). The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behaviour 16*, 683–692.
Wanner, E. (1980). The ATN and the Sausage Machine: which one is baloney? *Cognition 8*, 209–225.
Woods, W.A. (1970). Transition network grammars for natural language analysis. *Communications of the ACM 13*, 591–606.
Woods, W.A. (1973). An experimental parsing system for transition network grammars. In: *Natural Language Processing*, R. Rustin (Ed). Algorithmics Press, New York.