June 1994

Natural Language Processing

Mark Steedman
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-94-32.

Natural Language Processing

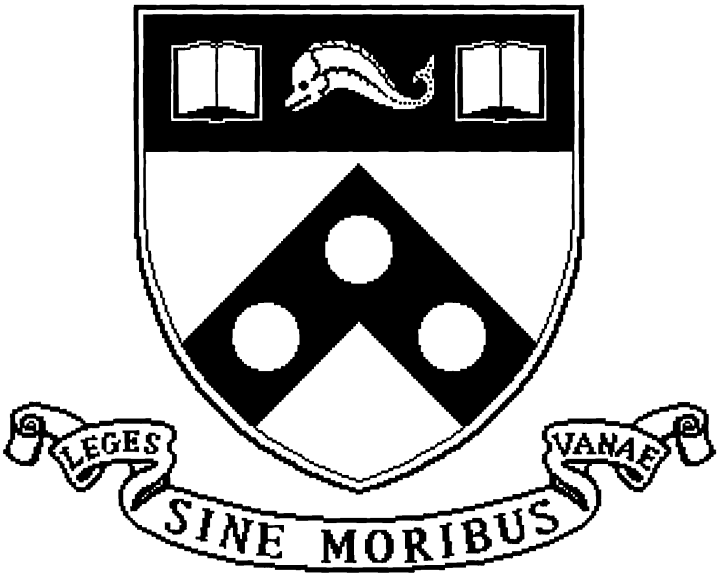
Abstract
The subject of Natural Language Processing can be considered in both broad and narrow senses. In the broad sense, it covers processing issues at all levels of natural language understanding, including speech recognition, syntactic and semantic analysis of sentences, reference to the discourse context (including anaphora, inference of referents, and more extended relations of discourse coherence and narrative structure), conversational inference and implicature, and discourse planning and generation. In the narrower sense, it covers the syntactic and semantic processing *sentences* to deliver semantic objects suitable for referring, inferring, and the like. Of course, the results of inference and reference may under some circumstances play a part in processing in the narrow sense. But the processes that are characteristic of these other modules are not the primary concern.

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-94-32.

Natural Language Processing

MS-CIS-94-32
LINC LAB 270

Mark Steedman



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

June 1994

NATURAL LANGUAGE PROCESSING*

MARK STEEDMAN
University of Pennsylvania

*Thanks to Michael Niv and Mike White for reading and commenting upon the draft. To appear: in M. Boden (ed.), *Handbook of Perception and Cognition, 14: Computational Psychology and Artificial Intelligence*. The work was supported in part by NSF grant nos. IRI90-18513, IRI91-17110, and CISE IIP, CDA 88-22719, DARPA grant no. N00014-90-J-1863, and ARO grant no. DAAL03-89-C0031.

Contents

I: INTRODUCTION

A: SCOPE OF THE STUDY

The subject of Natural Language Processing can be considered in both broad and narrow senses. In the broad sense, it covers processing issues at all levels of natural language understanding, including speech recognition, syntactic and semantic analysis of sentences, reference to the discourse context (including anaphora, inference of referents, and more extended relations of discourse coherence and narrative structure), conversational inference and implicature, and discourse planning and generation. In the narrower sense, it covers the syntactic and semantic processing of *sentences* to deliver semantic objects suitable for referring, inferring, and the like. Of course, the results of inference and reference may under some circumstances play a part in processing in the narrow sense. But the processes that are characteristic of these other modules are not the primary concern.

This chapter is mainly confined to the narrower interpretation of the topic, although it will become apparent that it is impossible to entirely separate it from the broader context. The reader interested in the more global problem is directed to the readings mentioned in the section on "Further Reading", below.

B: THE ANATOMY OF A PROCESSOR

All language processors can be viewed as comprising three elements. The first is a grammar, which defines the legal ways in which constituents may combine, both syntactically and semantically, to yield other constituents. The syntactic class to which the grammar belongs also determines a characteristic automaton, the minimal abstract computer that is capable of discriminating the sentences of the language in question from random strings of words, and assigning structural descriptions to sentences appropriate to their semantics.

The second component of a processor is a non-deterministic algorithm that uses the rules of the grammar to deliver such structural descriptions for a given sentence. Such an algorithm schema determines, for example, whether the rules are used "top-down" or predictively, or "bottom-up", or some mixture of the two, and whether the words of the sentence are examined in order from first to last, or in some less obvious order. However, as the term "non-deterministic" suggests, this component does not itself determine what happens when more than one rule

can apply in a given state of the processor.

This last responsibility devolves to the third component, the oracle, or mechanism for resolving such local processing ambiguities.[1] The oracle decides *which* action should be taken at points in the analysis where the non-deterministic algorithm allows more than one.

Such nondeterminism can arise from two distinct sources. The first source is lexical syntactic ambiguity, as in the case of the English word "bear", which can be either noun, nounphrase, or verb. The second source is structural syntactic ambiguity, which arises when there is more than one way to combine the same lexical categories to yield a legal sentence. For example, a parser for English that has dealt with the words "put the book on the table ..." is likely to be in a state in which the verb "put", the noun-phrase "the book" and the prepositional phrase "on the table" could be combined to yield a complete verb phrase, but where the nounphrase and the prepositional phrase could also be combined to yield a nounphrase "the book on the table". If the sentence ends at that point, as in a, below, then this "local" parsing ambiguity is resolved in favour of the former analysis. However, if the sentence continues as in b, then the latter analysis, in which "on the table" modifies the noun-phrase rather than the verb-phrase, is available, and in fact preferred. (Another analysis is possible, so this sentence is said to be syntactically "globally" ambiguous).

(1) a. Put the book on the table.
b. Put the book on the table in your pocket.

C: The Relevance of Computation

Computer Science provides a rich source of models for theories of all three modules of the human processor, drawing not only on the manner in which similar modules are treated in constructing compilers and interpreters for artificial programming languages, but also work within the Artificial Intelligence paradigm. This work provides both theories and working examples of the way in which syntactic processing, semantic processing, and referential processing can be interleaved and may in very restricted senses interact during processing. This has been made possible by the development of computational systems for knowledge

[1] The division of labour in processing between a non-deterministic algorithm and and an oracle is not always made explicit, particularly in implementations. However, we shall see that all sentence processors can and should be viewed in this way.

representation and inference within AI, which have provided notations for the contextual and domain-specific knowledge involved in linguistic comprehension.

The contribution of AI Knowledge Representations is at least as important as that of compiler theory, for we should be clear at the outset that in many ways programming languages are strikingly unlike their natural counterparts. Programming languages typically have tiny grammars by comparison with human languages. Compilers typically cope with very complex expressions, and are required to be "complete" – that is, to guarantee a correct analysis for any legal expression of the language. Human processors, on the other hand, deal with expressions that are structurally rather simple. And there is every indication that the human parser is in syntactic terms very far from complete. The most obvious evidence for incompleteness is the well-known apparent non-parsability of "center-embedded" sentences such a, below, in comparison with their "right-embedded" relatives like b:

(2) a. The rat the cat the dog bit chased escaped.
b. This is the dog that bit the cat that chased the rat that escaped.

The difficulty of a has widely been supposed to arise from some limitation in the size of working memory – say from a bound on the size of the push-down stack that is characteristic of context-free grammars, and whose use is crucial in the case of center-embedding.[2]

Even more striking evidence of incompleteness arises from certain well-known "garden-path" sentences first noted by Bever. Example a, below, includes a lexical ambiguity which leads the processor so seriously astray that naive subjects are typically unable to identify any analysis for a sentence with which they would otherwise have no trouble, as shown by the syntactically identical b:

(3) a. The horse raced past the barn fell.
b. The horse driven past the barn fell.

II: COMPUTATIONAL THEORIES OF PROCESSING

[2]Interestingly, this very old suggestion has never been really successfully formalised, and recent work by Gibson 1994, in press suggests that the catastrophic effect in a, above, is due to center embedding *within subjects*, rather than center-embedding alone. Niv 1993 cites the increased acceptability of examples like "a book that some Italian I've never heard of wrote will be published next week" (which he attributes to B. Frank) to suggest that the effect stems from an interaction of subjecthood and other discourse factors related to definiteness.

A: THE GRAMMAR

1: COMPETENCE AND PERFORMANCE

The Grammar that we have identified as a component of the processor above is clearly a module of the "Performance" system, rather than of "Competence", in the linguist's sense of those terms. When we contemplate the human processor as a whole, we should be aware that this performance grammar is conceptually distinct from the competence grammar that the linguist provides. In fact there is no logical necessity for the structures that the processor builds to have anything to do with the structures that are implicated by the competence grammar – that is, the structures that are required by the semantics (and the linguist). As Berwick and Weinberg1984, esp. p.78-82 have noted, the processor can in principle parse according to a grammar that is quite different from the one that is most directly related to the semantics, provided that there exists a computable homomorphism mapping the structures of this "covering grammar" onto the structures of the competence grammar. If the homomorphism is simple, so that the computational costs of parsing according to the covering grammar plus the costs of computing the mapping are less than the costs of parsing according to the competence grammar, than there may be a significant practical advantage in this strategem. For this reason, it is quite common for compilers and interpreters to parse according to a weakly equivalent covering grammar, mapping to the "real" grammar as defined by the reference manual via a homomorphism under concatenation on a string representing the derivation under the covering grammar. This strategem has sometimes been used in programming language compilers, when a parsing algorithm that is desirable for reasons of efficiency demands grammars in a normal form that is not adhered to by the grammar in the reference manual. Such a tactic has also been used in at least one early artificial parser for natural languages, in which it was necessary to use a top-down algorithm, which as we shall see below is ill-suited to the left recursive rules which commonly occur in natural grammars.

Nevertheless, considerations of parsimony in the theory of language evolution and language development might also lead us to expect that, as a matter of fact, a close relation will turn out to hold between the competence grammar and the structures dealt with by the psychological processor, and that it will in fact incorporate the competence grammar in a modular fashion. One reason that has been frequently invoked stems from the fact that language development in children is

extremely fast. This speed in turn suggests that it proceeds via the piecemeal addition, substitution and modification of individual rules and categories of competence grammar. However, the addition of, or change to, a rule of competence grammar will not in general correspond to a similarly modular change in a covering grammar. Instead, the entire ensemble of competence rules will typically have to be recompiled into a new covering grammar. Even if we assume that the transformation of one grammar into another is determined by a language-independent algorithm, and can be computed each time at negligible cost, we have still sacrificed parsimony in the theory, and increased the burden of explanation on the theory of evolution. In particular, it is quite unclear why the development of either of the principal components of the theory in isolation should confer any selective advantage. The competence grammar is by assumption unprocessable, and the covering grammar is by assumption uninterpretable. It looks as though they can only evolve as a unified system, together with the translation process. The evolution of such a system is likely to be much harder to explain than that of a more directly competence-based system.

Indeed the first thing we would have to explain is why a covering grammar was necessary in the first place. The reference grammars of programming languages and the competence grammars of natural languages have syntaxes that are ill-suited to parsing with our favourite algorithms because they are constrained from outside by our own requirements. It is we who find Greibach Normal Form tedious, and find grammars with left recursive rules congenial, forcing the use of covering grammars by some artificial processors. It is quite unclear what comparable external force could have the effect of making natural grammars similarly ill-matched to the *natural* sentence processor.[3]

[3]The natural processor could certainly require grammars to be in some normal form. However, provided that the normal form is a class of grammars of the same automata-theoretic power that the semantics of the language requires (and therefore of the same power as the competence grammar), we would expect that normal form to simply be a characteristic of the grammars we observe. In other words, we would view it as a (processing-based) constraint on the form of the competence grammars that actually exist. It is even less easy to believe that a covering grammar could be forced by the sentence processing mechanism being of intrinsically lower automata-theoretic power than the competence grammar. We would first have to ask ourselves how these two systems which by assumption have completely different automata-theoretic character could begin to talk to one another in the first place. We should then have to ask ourselves whether it would not be simpler for evolution to bring the processor up to the automata theoretic level of the competence grammar. After all, such a mechanism has already been evolved once, in the form of the interpreter for the semantics.

For similar reasons, we should expect that the syntactic categories and rules of the competence grammar will stand in the closest possible relationship to the categories and rules of a compositional semantics, in which the interpretation of complex expressions is wholly determined by the interpretation of their component parts, and where the assembly of semantic interpretations can be carried out in lock step with syntactic derivation. Since the sole purpose of syntax is to identify semantic interpretation, anything else seems to pointlessly complicate the problems of evolution and acquisition

We shall therefore in the remainder of the chapter adopt the assumption that the grammar that is used by or implicit in the human sentence processor is the competence grammar itself, a position that Bresnan and Kaplan (1982) have named the "Strong Competence Hypothesis". To adopt this position is not to assume that linguists will have had the foresight to provide their grammars in the form in which we need them to make predictions on the basis of this hypothesis, or even that any of the (disturbingly numerous) alternative formulations are entirely correct. However, the fact that our access to the underlying semantic representations is very limited indeed, and the fact that there are so many possible parsers for each grammar both mean that the linguists' methods for studying competence grammar remain an essential aid. Indeed, in the history of the formal theory of natural grammar, linguists and computationalists have seemed like mountaineers roped together on a mountainside, each in turn helping the other towards the summit.

2. Computational Theories of Grammar

A. The Context-free Core: To talk about this collaboration we need a notation. We will start with a computational notation for grammars called "Definite Clause Grammars" (DCG). This notation is convenient both because it is so close to Phrase Structure Grammar and because it is directly compatible with a useful computational device called "unification", which we can use to build derivations or even interpretations very simply indeed. A DCG for a fragment of English might begin with rules that we can write as follows:

(4)
$$
\begin{array}{rcll}
S : vp\ np1 & \rightarrow & NP_{agr} : np1 & VP_{agr} : vp \\
VP_{agr} : iv & \rightarrow & V_{INTR,agr} : iv & \\
VP_{agr} : tv\ np2 & \rightarrow & V_{TRAN,agr} : tv & NP : np2 \\
NP_{SING} : pn\ e_1 & \rightarrow & PN_{SING} : pn & \\
NP_{agr} : (q\ e_2)(n\ e_2) & \rightarrow & DET_{agr} : q & N_{agr} : n
\end{array}
$$

In this notation, the familiar PS rules such as $S \rightarrow NP\ VP$ are expanded as in

the first rule to include interpretations. Thus an expression of the form $S : vp\ np1$ denotes a grammatical category of syntactic type S (a sentence) and semantic interpretation $vp\ np1$, in which the variable vp is a predicate that is applied to an argument $np1$. Whenever possible, we shall abbreviate such categories as S, etc. In the first rule, both vp and $np1$ also occur as the interpretations of two subconstituents of type NP and VP. Subscripts on symbols like V_{INTR} and V_{TRAN} are a convenient notation for categories that we want to think of as bundles of feature-value pairs, like $\pm$*transitive*. Features with lower case, as in NP_{agr}, are to be read as variables, or feature-value pairs with unbound values. Corresponding unsubscripted categories like NP are to be read as categories whose value on its features is irrelevant to the application of the rule.

We pass over the details of the actual semantic types of variables like iv, tv, pn, and q, except to note we are assuming that they range over higher-order types of the kind familiar from the work of Montague 1974, so that the latter two are varieties of what are called "generalised quantifiers", binding the variables e_n which range over entities in the model or database, and that the former two are even higher types. The important thing to note about this semantics is that it is completely integrated into the syntactic rules, and that interpretations can therefore be assembled simultaneously with syntactic analysis.

Rules written as in 4 leave the string order of subconsituents on the right of a rule, such as NP and VP in the first rule, implicit under an obvious convention in the linear order of those symbols on the page. An alternative notation which makes linear order explicit, but which is in other respects very similar to DCG grammars is that of Augmented Transition Network grammars (ATNs, Woods, 1970), which we can think of as replacing groups of one or more DCG rules by a finite automaton, where each automaton can "call" the others recursively. The earlier DCG rules might be written as in Figure 1.[4]

Like the PS rules that they resemble, the rules in either notation can be applied top-down ("To find a sentence meaning $S : vp\ np1$, find a noun-phrase meaning $np1$ to the left of a verb phrase meaning vp") or bottom-up ("If you find a noun-phrase meaning $np1$ to the left of a verb phrase meaning vp, make them into a sentence meaning $S : vp\ np1$"). However, it is important to remember that both DCGs and ATNs are *grammars* not parsers, and can be fitted up with any

[4]This notation differs considerably from Woods' own in a number of respects. The original ATN grammars packed as many PS rules as possible into each network, so produced "flatter" grammars than this example suggests. In effect they used a covering grammar, to make life easy for the algorithm.

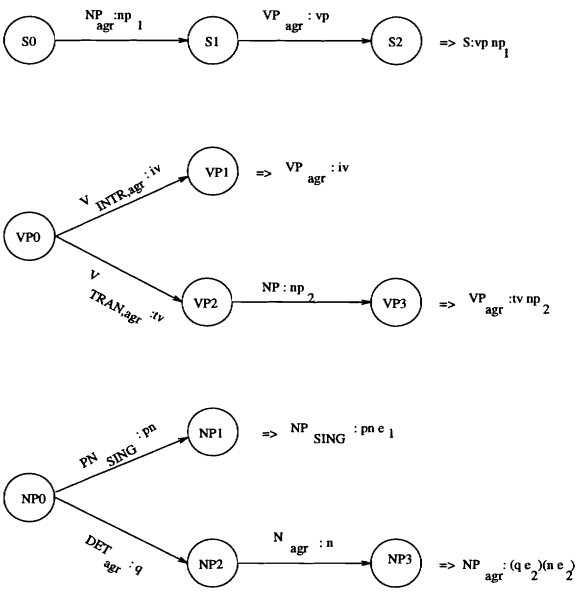


Figure 1: An ATN Grammar

algorithm/oracle combination we like.

Whichever notation we use, and however the algorithm applies the rules, a mechanism called "unification" can be used to ensure that the S that results from a successful analysis of a given string is associated with the appropriate interpretation.[5]

Informally, unification can be regarded as merging or amalgamating terms that are "compatible", and as failing to amalgamate incompatible ones, via an algorithm that "instantiates" variables by substituting expressions for them in one or other of the expressions.[6] For example, the following pairs of terms unify, to yield the

[5]The historical ATN grammars in fact associated explicit "register-changing" rules with transition networks to achieve the same effect of information-passing. Again, the distinction is unimportant for present purposes.

[6]More technically, the result of unifying two compatible terms is the most general term that is an instance of both the original terms.

results shown:[7]

(5)
$$\begin{array}{llll} x & A & \Longrightarrow & A \\ F(GA) & x & \Longrightarrow & F(GA) \\ Fx & F(Gy) & \Longrightarrow & F(Gz) \\ (FA)x & (Fy)y & \Longrightarrow & (FA)A \end{array}$$

The following pairs of terms do not unify:

(6)
$$\begin{array}{llll} A & B & \Longrightarrow & fail \\ Fx & Gy & \Longrightarrow & fail \\ (FA)B & (Fy)y & \Longrightarrow & fail \end{array}$$

For example, suppose the lexicon tells us that the word "Harry" bears the category NP : *HARRY*e_1, and suppose that "walks" has the category VP : *WALKS*.[8]

If we find the word "Harry" to the left of the word "walks", it follows that we can unify this sequence of categories with the right hand side of the first rule in 4. This has the effect of replacing the variable vp by *WALKS* and the variable $np1$ by $HARRY\ e_1$ throughout the rule. The resulting S is therefore associated with the expression *WALKS* (*HARRY* e_1). The derivation, in which the unification does the work of a compositional semantics, can be represented by the usual sort of tree diagram as in Figure 2.[9] In fact, there is a little more to it than this: the variable or undefined value agr on the syntactic categories NP_{agr} and VP_{agr} in the rule also got bound to the value $3SING$ by the unification. Had the subject and verb born different values on the agreement feature, as in the following illegal string, the unification, and hence the whole derivation, would have failed:

(7) *Harry walk

Thus we have the rudiments of an account of linguistic agreement. Of course, it isn't a very *good* account. In a more complete fragment of English we would

[7]It should be noticed that the unification of two variables is a "new" variable, distinct from either. It should also be noticed that nodes in trees, terms, and variables are, under this interpretation, pointers to data structures, and unification makes two pointers point to the identical data structure. Strictly, therefore, interpretation structures are directed acyclic graphs, rather than the trees that the present notation suggests.

[8]The distinguished variable e_1 ranges over individuals in the discourse model, which we can think of as a database of facts.

[9]As in the case of syntactic categories like NP_{3SING}, we assume that symbols like $WALKS$ in interpretations are merely place-holders for more complex terms separating tense, aspect, verb-sense etc.

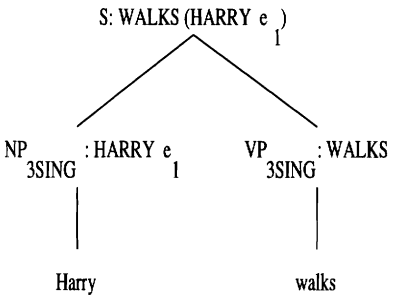


Figure 2: A DCG derivation

want to further unpack the feature agr and its values like $3SING$ into a bundle made up of a number of features like num (number) and per (person). Linguists would probably point out that some phenomena of agreement appear to be most parsimoniously described in terms of *disjunction* of feature-value pairs, a fact which may suggest we need a more refined representation altogether. However, these refinements are of less interest for present purposes than the observation that we have here the beginnings of a very simple account of a phenomenon that in the early days of generative grammar was thought to require Aspects-style transformations, a very powerful class of rule indeed.

The interpretation that is delivered by the above derivation is closely related to the derivation tree itself. The observation generalises to other derivations that are permitted. This fact suggests that what we have so far developed is merely a computationally convenient form of context free grammar, "slaved" via unification to a device that builds interpretation structures as derivation proceeds, an idea that seems to originate with Kuno in the sixties. This is a helpful way to think about the processor, but some caution is in order. We have so far been quite vague about the unification mechanism, and in particular about the types of values that features may bear. In particular, if we allow the values to be lists, then we shall change the automata-theoretic power of the grammar (which may of course be what we want). Completely unrestricted feature systems can also threaten the tractability of the parsing problem (cf. Barton, Berwick and Ristad 1987).

With this notation in place, we can begin to look at more complex constructions. It is convenient to collect these into three classes. The "bounded" constructions, which include many phenomena that were originally thought to require transformations, are those which relate argument positions *within a single tensed clause*. These constructions are to be contrasted with "unbounded" constructions, such as

relativisation, which can relate elements of sentences across one or more clause boundaries. The unbounded constructions can be divided into "well behaved" constructions, like relativisation itself, and "less-well-behaved" constructions, notably including coordination. The computational implications of this last class are a topic in their own right, and will only be very briefly touched on below. It is the first two classes that have received most attention, both from linguists and from computational linguists.

B. BOUNDED CONSTRUCTIONS: All languages of the world appear to include devices, usually morphologically marked, which affect the mapping between semantic predicate argument relations and linear order and/or surface case of the corresponding expressions in the sentence. An example in English is afforded by the contrast between active and passive morphology, which produces clauses in which the semantic object is respectively realised as accusative or sentence final, and nominative, or sentence initial.

(8) a. I like Ike.
b. Ike is liked.

Others relate the same surface argument expression to more than one semantic argument role, and sometimes to arguments of different verbs standing in a semantically subordinating relation, as in morphological reflexivisation, or various raising or control constructions, as in the following English example:

(9) I persuaded Ike to leave.

The languages of the world show very striking similarities in the range and type of such constructions that they offer. To capture and explain these regularities is a major goal of contemporary linguistic research.

All of these constructions share an important distinguishing property called "boundedness". That is to say that the semantic arguments whose relations to surface grammar these constructions determine must either be involved in the same proposition (as in the case of the passive and morphological reflexivisation), or in a proposition and an *immediately* subordinate complement proposition, as in the last example.

The bounded and structure-preserving properties of these constructions immediately suggest that they should be handled *in the context-free base component of the grammar*, as proposed in the ATN framework by Woods, 1970. Many modern theories of syntax make a related assumption, which goes by the name of the "Base

Generation Hypothesis" (cf. Brame 1978). In the DCG notation we can capture the idea as follows:[10]

(10) $VP : ocv\,(vp\,y)\,y \;\rightarrow\; V_{oc} : ocv \quad NP : y \quad VP_{to-inf} : vp$

Another very natural way to interpreting Base Generation is to capture the bounded constructions *in the lexicon* – that is, in the subcategorisation frames for object control verbs and the like. This is the tactic that is adopted in Categorial Grammar (CG, Oehrle et al. 1988), Lexical-functional Grammar (LFG, Bresnan 1982), certain versions of Tree-Adjoining Grammar (TAG, Joshi et al. 1991), and Head-driven Phrase-structure Grammar (HPSG Pollard and Sag 1994).

A DCG grammar expanded in this way continues to be closely related to Woods' ATN analysis of the bounded constructions, with unification again doing the work of "Register Modification". The fact that both the ATN and the DCG work exclusively at the level of argument structure or the interpretations of immediate constituents goes a long way towards explaining the bounded character of these constructions. This benefit of the computational approaches has become standard, and is implicit in nearly all linguistic theories of the constructions, including GPSG, HPSG, LFG, Lexicalised TAGs, and certain versions of Government-Binding theory (GB, Chomsky 1981). For this reason, we shall have little more to say about the bounded constructions here, except to note that this is one place where computational linguistics has directly influenced mainstream linguistics and psycholinguistics (see Bresnan 1978 for further discussion of the relation between linguistic accounts and the ATN.)

C. UNBOUNDED CONSTRUCTIONS: The languages of the world show similar consistency in respect of another family of constructions. These are known as "unbounded", because they involve dependencies at the level of the interpretation or argument structure between expressions which may be separated by unboundedly many intervening elements. Examples in English are relativisation, wh-question formation, topicalisation and, arguably, right node raising, exemplified in 11 below:

(11) a. People that . . . I like.
b. People, . . . I like!
c. I like, and . . . you merely tolerate, people who own cats.

[10] As usual, the interpretation here is highly simplified, and all the real work is being done by the translation *ocv* of the object control verb. A linguist would foresee a problem for the Binding theory in this particular representation, which we pass over here in the interests of brevity.

For each of these expressions, an infinite number of further well formed expressions of the same type can be generated by recursively inserting instances of the string "O'Grady says that..." in place of "...". It follows that the verb *like* can be unboundedly distant from the head noun *people* of its semantic object argument. While there are a number of patterns of relativisation across languages, including cases where the relative pronoun is either *in situ* with the verb, or is omitted entirely, in all languages such constructions involve an unbounded dependency.[11]

These constructions seem quite different from those that we encountered in the last section. We cannot build the argument structures needed for semantic interpretation merely by identifying elements in translations of immediate constituents in a context-free rule, since in general the elements related by the dependency cannot be elements belonging to a single CF rule.

All approaches to the unbounded dependency exhibited by constructions like the relative take the form of a context-free core, augmented by some extra apparatus for handling the unbounded dependencies themselves. Many of the interesting contrasts between the theories concern the automata-theoretic power that such extensions implicate. Although we have no very clear information concerning an upper bound on the power of human grammar, mechanisms of lesser power than a Turing machine clearly have considerable theoretical interest, if only to show where they break down.

Linguistic and computational theories of this "weakly non-context-free" kind have broadly fallen into two categories. The first type can be characterised as using the context-free base to determine an argument structure for an entire complex sentence, then using additional apparatus to establish long range dependencies in one fell swoop. An example is *Aspects*-style transformational grammar, which introduced tree-to-tree rules including variables in order to transform arbitrarily deeply embedded trees with *wh*-items *in situ* into trees with those items fronted. The other type of mechanism established long-range dependencies by "trickling" pointers or indices down a path through the derivation tree connecting the two dependent elements.

Aspects-style rules themselves were quickly identified as implicating full Turing machine power (Peters and Ritchie 1973). They also failed to explain a number of asymmetries in extractability of different arguments, of which a striking example is the "Fixed Subject Constraint", which describes the fact that in English and

[11] Some of these possibilities are actually exhibited in restricted forms in English, in pronoun-free relatives and multiple *wh*-questions.

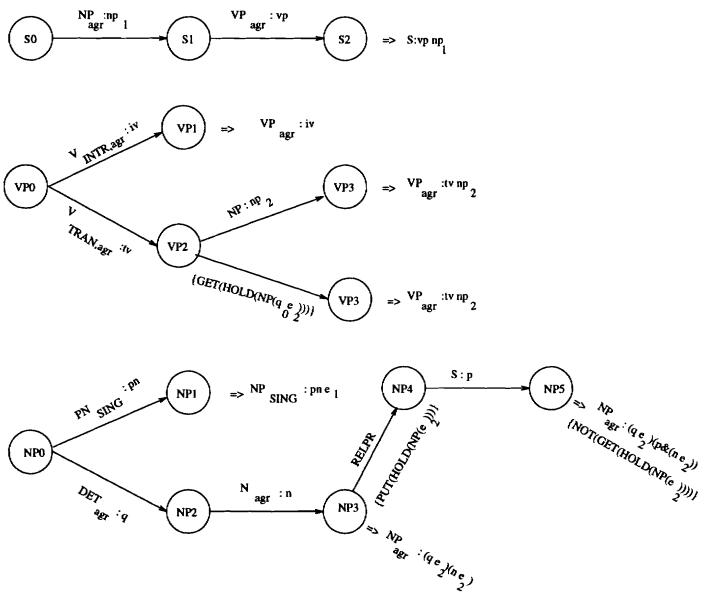


Figure 3: The ATN HOLD mechanism

many other SVO languages, subjects cannot unboundedly extract, unlike other arguments, as illustrated in the following example:[12]

(12) a. A man who(m) I think that Ike likes.
b. *A man who(m) I think that likes Ike.

For both of these reasons, there was considerable interest in certain computational versions of the swoop mechanism that appeared to be more constrained. Following early work by Thorne and Bobrow, the idea was most elegantly formulated by Woods 1970 for the ATN.

Woods' ATN allowed certain kinds of register-changing *side-effects* to be associated with state-transitions. Most of Woods' register-changing operations have up till now been subsumed under the unification mechanism. However, to handle long-range dependency, we shall associate such actions with a number of transitions that we shall add to the NP and VP nets in Figure 1. These actions will transfer special terms or markers into and out of into a special globally accessible register or store called HOLD, extending the grammar as in Figure 3. The actions

[12]Subjects of bare complements, as in "a man who(m) I think likes Ike" are exceptional in this respect.

concerning the HOLD register are enclosed in braces. They allow the grammar to achieve the same effect as a swoop transformational rule. To derive a relative clause, you simply use the ordinary rules of context free grammar, except that whenever you encounter a relative pronoun, you make it one end of a potentially unbounded NP dependency, expressed as a note in the HOLD register, and whenever a verb needs an NP argument, it has the option of satisfying the need for that argument and making it the other end of an unbounded dependency by retrieving an NP from HOLD.[13]

Part of the attraction of the ATN HOLD mechanism is that it offers a way to think about constraints on long range dependencies. For example, because we do not include the option of the subject being obtained from the HOLD register, we capture the Fixed Subject Condition illustrated in 12.[14]

Furthermore, the fact that sentences like 13, a below involve more than one dependency, and that those dependencies nest, as revealed by b, can be explained as arising from limitations on the HOLD store – perhaps that it is a push-down stack, as suggested by Woods 1973:[15]

(13) a. Which violin$_1$ is this sonata$_2$ easy to play$_2$ on$_1$
b. *Which sonata$_1$ is this violin$_2$ easy to play$_1$ on$_2$

The importance of this latter observation is considerable. When one sees a stack, one immediately thinks of a characteristic automaton, such as the push-down automaton (PDA) that is characteristic of context-free grammars. Since a push-down store is already implicit in the context-free rules, and since adding a further stack to mediate long range dependencies would in principle jump us up to full Turing machine power, the intriguing possibility is that the stack involved in long-range dependencies is in some sense the *same* stack involved in context-free rules, as proposed by Ades and Steedman 1982, p.522, and as further specified in the work of Joshi et al. discussed below.

Of course, to claim this much is not the same as claiming that natural languages are context-free. We know from work by Aho 1969 that a "nested stack" automaton, equipped with a single stack whose contents may themselves be stacks,

[13]A further check that the index has indeed been removed from HOLD is included on exit from the complex NP, in order to prevent overgeneralisation to examples like **A man that I like Ike.*

[14]A linguist would notice, however, that nothing in the present theory explains why this constraint appears to conspire with other aspects of word-order, cross-linguistically. Nor have we revealed how bare complement subject extraction is allowed.

[15]The example originates with Janet Fodor.

is of greater than context free power (but of lesser power than context sensitive grammars). However, it does suggest that we should try to account for unbounded dependencies in much the same way we accounted for bounded ones, by putting as much of the work as possible into the base rules themselves. This insight can be seen as underlying the second group of mechanisms for unbounded dependencies, in which a similar kind of pointer or index is "trickled" down the sentence during derivation, as opposed to being established in one fell swoop.

Within mainstream linguistics, this kind of explanation can be seen as surfacing in proposals by Bresnan and Chomsky which resulted in the "comp-to-comp movement" account of unbounded dependencies including relativisation. According to this theory, unbounded dependencies were the sum of a series of local dependencies between argument positions and the complementiser position in individual clauses. Thus transformational "movement" was restricted to bounded movement.

Since we have seen that bounded dependencies can be captured within generalised PS rules, it should be clear that it is a comparitively short step to grammars which eliminate movement entirely, and bring unbounded dependencies under the Base Generation Hypothesis. The proposal takes an extreme form in the Generalised Phrase Structure Grammars (GPSG) of Gazdar et al. 1985. The GPSG treatment can be included in our DCG rules by associating a feature-value pair SLASH, equivalent to a local HOLD register. In the original version of the theory, the value of this feature was simply a pointer identifying a unique long range dependency, and the theory was therefore weakly context free. The original base set of rules for each category such as S are included again with an empty slash feature. It is convenient to write such categories simply as S, NP, VP, etc, and to write the corresponding categories in which the SLASH pointer is of type NP as S/NP, NP/NP, VP/NP, etc. For every old style rule defining "non-slash" categories, there may be one or more rules which specify how an S/NP with non-empty SLASH feature passes that feature to its offspring. For example, we might introduce the following additional rules, among others:[16]

(14)
$$\begin{array}{lcllll}
S : vp\ np1/NP : x & \rightarrow & NP_{agr} : np1 & VP_{agr} : tv\ x/NP : x & & \\
VP_{agr} : tv\ np_2/NP : x & \rightarrow & V_{TRAN,agr} : tv & NP : np2/NP : x & & \\
NP_{agr} : (q\ e_2)(p\&(n\ e_2)) & \rightarrow & DET_{agr} : q & N_{agr} : n & RELPRO & S : p/NP : q_0\ e_2
\end{array}$$

The category NP/NP corresponds to the linguists notion of a trace or empty NP. We have again slyly captured the constraint upon extracting subjects by not

[16]In the original version of the theory, such extra rules were induced via "metarules".

including a rule of the form $S/NP \rightarrow NP/NP\ VP$. The multiple unbounded dependencies exhibited in sentences 13 could be handled within this type of GPSG, by a combination of the techniques we have seen for bounded and unbounded dependencies. However, the multiple intersecting unbounded dependencies that occur in Scandinavian languages and the arbitrarily many verb-argument dependencies that can intercalate in Germanic infinitival complements provide evidence that natural languages cannot be contained within the class of CF languages (see Gazdar 1988 for discussion). A number of other grammar formalisms with a computational orientation were developed in response to such observations, including Tree-Adjoining Grammars (TAG, Joshi et al 1991) and Combinatory Categorial Grammars (CCG, Ades and Steedman, 1982), and Gazdar's proposal to permit SLASH features to be stacks (cf. Gazdar 1988, Pollard and Sag 1994). Joshi et al. 1991 showed that a number of these grammars were weakly equivalent to indexed grammars, and that under certain limiting assumptions to *linear* indexed grammars, in which only a single stack-valued feature is involved.

D. LESS WELL-BEHAVED CONSTRUCTIONS: Before leaving the topic of grammar, it should be remarked that there remain a number of constructions that are much less well-behaved with respect to both linguistic and computational theories of the kind we have discussed so far. Coordination, Parentheticalisation, and Intonational Phrasing, all appear to operate upon a very wide range of fragments that bear very little resemblance to traditional constituents, while remaining subject to very strong and apparently syntactic constraints. The seriousness of the problems which examples like the following present for both the theory of competence grammar and the possibilities of efficient processing are an enduring problem that should not be underestimated:

(15) a. I will buy, and may read, your latest novel.
b. Give Dead-eye Dick a sugar stick, and Mexican Pete, a bun
c. Harry likes Adlai, and Mike, Ike

The lack of truly explanatory coverage of such examples suggests that there may be more to say about the computational nature of the competence grammar than we have been able to consider here.[17]

[17] I address this question elsewhere – see Further Reading, below.

B: THE ALGORITHM

1: NATURAL AND UNNATURAL ALGORITHMS

The parsing algorithms that have been developed for the purpose of compiling and interpreting programming languages come in a bewildering variety, each distinguished from the others on a number of parameters. The parameters determine the order in which the space of parser-states is searched, the manner in which the rules of the grammar are applied (top-down, bottom-up, or a mixture of the two), and in the use of various auxiliary "tables" or "charts" to increase efficiency.

The extent to which we can distinguish these devices empirically as models of the human parser is very limited, partly because of residual uncertainties about the nature of the competence grammar itself, but mostly because of their considerable dependency upon the third module of the processor, the oracle. We have already noted that these algorithms have been developed for languages very unlike our own, with no global ambiguity and very limited nondeterminism, so this result is perhaps not surprising. Nevertheless, these algorithms remain the sole source for models of the human algorithm, and it is important to understand the consequences of variation along each of these dimensions.

The appeal of the Strong Competence Hypothesis lies in the observation that it is evolutionarily and developmentally simpler to keep to a minimum the apparatus that the parser demands over and above the competence grammar itself. A parsing algorithm that requires a great deal of "extra" apparatus – particularly when that apparatus is language-specific – therefore tends to be less plausible as a candidate. However, the algorithm must, given a natural competence grammar and the assistance of an oracle, be capable of parsing efficiently. An algorithm for which natural grammars appear to induce inefficiency, or for which we find it hard to identify an effective oracle, will also appear implausible. We shall find that at first glance all the algorithms score rather badly for psychological pausibility on one or other of these criteria.

All of the algorithms that are discussed below were in the first instance defined for context-free grammars and the associated push-down automaton, since that is the effective level of the grammar for most programming languages. Indeed, the desire to exploit these algorithms, and therefore to keep to a minimum the additional apparatus required to deal with the apparently non-context-free properties of natural languages, was a major impulse behind the development of the

computational alternatives to grammatical transformations discussed in the earlier section on competence grammar.[18]

2. Parsing as Search

We can view the workings of the nondeterministic algorithm as a search problem, in which the task is to find all paths through a maze to the exit(s) corresponding to succesful parser terminations. The choice points in the maze where more than one path can be taken correspond to states of the parser in which the grammar allows more than one continuation, and the transitions between parser states correspond to actions such as reading another word or applying a rule of grammar.

The complexity of a parsing algorithm – that is, the relation that the number of computational operations and the amount of working memory required bear to the length of the sentence under analysis – depends on the characteristics of this search space. The maze may be built on a chain of "islands", so that all paths pass through certain "bridges" or "bottlenecks" corresponding to the same state, so that all overall paths share the same analysis or analyses on each island. If so the algorithm may be able to "share" these analyses. If the paths through the maze never cross then this saving will be impossible. Some paths through the maze are dead ends. Some paths are endless, leading neither to success nor to a dead end. These the parser should try to avoid.

If paths never cross, the search space can be viewed as a tree. There are two basic methods of searching trees. If the tree has b branches at each branching point, we will refer to b as the "branching factor". (More realistically, since the degree of non-determinism at a branch-point may vary, we should think of the branching factor b as the *average* number of branches that do not immediately succeed or fail.)

One way to search the tree is to work "breadth first", starting at the root, looking at the first branches to see if they lead to success states, or if they are dead ends, and otherwise iterating over all the successor states. This search clearly has exponential costs both in terms of the number of alternatives that must be examined (and hence time), and in terms of the memory we need to keep track of the b^n alternative states on the "frontier" of the search.

[18]The "weakly non-context free" grammars discussed at the and of that section have a characteristic automaton called an "extended push-down automaton". The classic algorithms discussed below, such as Earley's, generalise quite directly to this automaton.

Another way is to work "depth-first", choosing one branch from the root (say the leftmost, or the "best" under some criterion), and then choosing from its successors by the same criterion. Whenever you reach a success- or failure- point, you back up to the last choice-point, and take the *next* -leftmost (or -best, or whatever) alternative. This regime also takes exponential time in the depth of the search (although its memory requirements are only order n, because it only needs to keep the n states on the path back to the root).

Clearly, the greater the nondeterminism induced by the grammar, the more complex the maze and the larger the number of wrong alternatives leading to blind alleys.

We have already noted that a huge degree of nondeterminism appears to be characteristic of all natural languages. A lower bound estimate of what we are up against can be obtained by observing that the branching factor b must be at least as great as the average lexical ambiguity, and the depth n of the search tree must be at least as great as the number of words in the sentence. (This lower bound is over-optimistic, since it takes no account of structural ambiguity). It follows that exhaustive search is likely to be unacceptably costly in computational terms. Nor should we expect parallelisation of our search algorithm to solve this problem. For any fixed n we can in principle make the search linear in n by providing b^n processors. But this number is also exponential in n, and is impracticably large for the range involved here.

As with all arguments from worst-case complexity, caution is in order here. The effective branching factor b may be smaller than we think, say because the search space has islands in the sense mentioned earlier. Moreover, n is quite small for natural sentences – on the order of 20 for written text. We shall see that within such bounds an exponential algorithm may perform better than a polynomial one. Nevertheless, even before we start to consider particular algorithms in detail, it is likely that we shall find ourselves in the same position as large search problems like chess-playing – that is, of having to adopt heuristic search strategies to narrow the effective branching factor. In the case of sentence processing, the heuristics must identify with high reliability the alternative at each point that will in fact lead to an analysis. While some less favoured alternatives may be stored for later use in case of failure, we may be forced to adopt a "best-first" version of depth-first search. Such methods are likely to be incomplete – that is, there may well be sentences that are allowed by the grammar, but for which our algorithms will fail to deliver analyses, or alternative analyses for ambiguous sentences that they will fail to detect.

This will come as no surprise. We have already noted that human processors are rarely aware of global ambiguities. We have also seen in the case of subject center-embedded sentences like 2 and garden-path sentences like 3 that both the psychological algorithm and the psychological oracle are incomplete. The significance of this observation for the nondeterministic algorithm is that individual algorithms differ in where they make the exponential costs show up, and in what particular problems they leave for the oracle to solve.

3: LEFT-TO-RIGHT PROCESSING

A further dimension of variation for the algorithm concerns the order in which the words in the string are examined. In nearly all cases they are examined in order of utterance from first to last, usually referred to as "left-to-right".[19] While this regime has an obvious intuitive appeal, it is worth noting that regardless of other details, one wants the parser to examine the most informative items earliest, in order to limit uncertainty, and that it seems to be an empirical fact about English and other languages that leftmost boundaries are much less ambiguously marked than right.

There is one exception to the standard left-to-right regime in modern natural language processors. Under a similar rationale of starting with unambiguous information, in tasks involving text derived via speech processing or other sources of noise and error, it is sometimes expedient to begin anywhere that yields a comparatively unambiguous analysis. This regime is know as "Island-driven" parsing.

4: SIMPLE BOTTOM-UP VS. SIMPLE TOP-DOWN

A. SIMPLE BOTTOM-UP: The simplest bottom-up nondeterministic algorithm is often referred to as the "Shift-reduce" algorithm, because it is defined in terms of two processes of "shifting", or pushing a category onto the stack of the push-down automaton, and "reducing", or combining categories on top of the stack according to a grammatical rule.[20] In its most general form, the algorithm can be defined in terms of the stack of the PDA (which initally is empty), and a buffer (which

[19]This term of course assumes an English orthography. We shall follow this lamentably chauvinistic terminology without comment from now on, because the alternatives are so cumbersome.

[20]Confusingly, the term "bottom-up" is sometimes used to refer to what is discussed below as the "left-corner" algorithm, while the term "shift-reduce" is sometimes used to refer to a particular kind of deterministic shift-reduce algorithm discussed below as the $LR(k)$ algorithm.

initially contains the words of the sentence in order),as follows:

(16) While the buffer and the stack are non-empty, EITHER:

1. If the topmost items on the stack match the right-hand side of a production rule, remove them from the stack, REDUCE them according to the rule, and place the result on top of the stack, OR:
2. remove the next word from the buffer, and SHIFT its grammatical category to the top of the stack.

Any empirical predictions from such a minimal algorithm must wait for the discussion of possible oracles that will determine which of these actions is preferred is states when both are possible, but we can make the following general remarks.

First, we should note that the non-determinism of this algorithm is of three distinct kinds, which we can refer to as SHIFT/SHIFT conflicts (which arise when a lexical item has more than one grammatical category), SHIFT/REDUCE conflicts, and REDUCE/REDUCE conflicts (when more than one rule can apply).

Second, the simple bottom-up parser is entirely non-predictive. This threatens efficiency, for a reason that will become obvious from considering the following sentence:

(17) The men walk

Any grammar of English must reflect the fact that "men" can (like most common nouns) be an NP as well as a noun. This fact introduces a SHIFT/SHIFT conflict when the word "men" is encountered, so that unless the oracle rules otherwise, the parser is likely to build the useless sentence "Men walk", despite the fact that there is no possible continuation of the words "the men walk" that could require an S at this point. Since the VP may be arbitrarily complex, this redundant analysis is likely to proliferate, threatening the usual exponential consequences. Of course, the oracle may indeed be smart enough to prevent this, and the chart-based techniques discussed below can also be applied, but it is a generic problem with the algorithm. Nevertheless, purely as a nondeterministic algorithm, it retains the appeal of requiring the absolute minimum of extra apparatus, over and above the grammar and the corresponding automaton.

B. SIMPLE TOP-DOWN: The simple top-down or "recursive descent" algorithm is entirely predictive. Starting with the start symbol S, the algorithm non-deterministically picks a rule that expands that symbol, say rule 1 of the fragment

4, $S \rightarrow NP\ VP$. For each non-terminal symbol in the expansion, the algorithm recursively calls itself on that symbol. In order to exploit the standard left-to-right order of processing discussed in section 2 above, it is usual to carry out this recursion "leftmost first" – in this case, to process the NP before the VP. For each terminal or preterminal symbol, a lexical item of the right category must be present at the appropriate point in the string. The algorithm uses the stack of the PDA to keep track of the recursion.

This process is conveniently described in terms of a dot notation, which captures the notion of a partially parsed expansion. Thus when we first call the algorithm recursively to parse the NP in the above expansion, we stack the following "dotted rule", so that when the algorithm returns from parsing the NP, it knows that it must parse the VP:

(18) $S \rightarrow NP\ .\ VP$

When it returns successfully from a recursion, it removes this symbol from the top of the stack and checks whether the dot is at the right hand end. If it is, the constituent is complete, and it returns. If it isn't, it moves the dot on past the next symbol, replaces the dotted rule on top of the stack, and recurses. That is, in this case it stacks the following dotted rule and goes off to look for a VP:

(19) $S \rightarrow NP\ VP\ .$

The left-to-right version of this algorithm can be informally stated as follows:

(20) To parse a category of type C:

1. Look at the category W of the next word in the string. If W=C, exit.
2. Choose a rule that expands C, and recursively parse the elements of the expansion in order from leftmost to rightmost, using a dotted rule and the stack to keep track.

This is the nondetermistic algorithm that was used in early implementations of ATN parsers.

The top-down algorithm minimises the wasteful construction of spurious constituents that can never take part in any derivation. For example, under the standard leftmost-first regime, since the rules of the grammar of English never predict S immediately succeeding a Determiner, the algorithm never builds the spurious S "Men walk" in sentence 17.

However, there are also countervailing disadvantages to the algorithm. Precisely because it is predictive, the search space that it must find its way through

is effectively the entire space permitted by the grammar, including states corresponding to analyses for which there is no support whatever in the string itself. Of course this search space is narrowed considerable once the first words of the sentence have been accounted for. But it makes life more difficult for the oracle, and in the absence of the chart-based techniques discussed later the recursive descent parser is exponential in the worst case.

C: MIXED BOTTOM-UP TOP-DOWN ALGORITHMS: Because of the respective disadvantages of pure top-down and bottom up algorithms, there has been considerable attention to a group of algorithms which attempt to combine the advantages of the two.

The simplest left-to-right version of the idea is the "Left Corner" algorithm. It can be thought of as starting bottom up, with a word or a constituent that is actually present in the sentence, and then using those rules of which that constituent is the leftmost daughter to recursively left-corner parse further constituents to the right. It can be informally stated as follows:

(21) To parse a category of type C:

1. Look at the category W of the next word in the string.
2. If W=C, exit.
3. Choose a rule that expands any category N such that the leftmost element of the expansion is W.
4. Recursively parse the remaining elements of the expansion in order from leftmost to rightmost, using a stacked dotted rule to keep track.
5. Assign the category N to W and go to 2

This elegant algorithm combines the "data-driven" advantage of the bottom up algorithm with the "predictive" virtues of the recursive descent algorithm.

D. CHART PARSING: All three nondeterministic algorithms described in the previous section are, in the absence of an effective oracle, technically computationally intractable. That is to say that if all we do to make them deterministic is to equip them with a "backtracking" memory for all choice-points, and an apparatus for failing back to these choice-points to try other alternatives, then all of them have worst-case exponential costs.

Like all worst-case computational complexity results, this one should be treated with some caution. Whether the worst-case is encountered in practice depends on

a number of factors, including the particular grammar involved. In particular, the left-corner algorithm can perform better than the non-exponential CKY algorithm considered below for quite practically useful grammar fragments (Slocum 1981).

Nevertheless, it is important to know that the complexity of the search problem can be greatly reduced, because of an important property of the parsing search space that has not been discussed here so far.

Natural language sentences are typically made up of a number of "islands", corresponding to substrings whose analysis or analyses are independent of those on the neighbouring islands, and of the sentence as a whole. To take an example from Marcus 1980, when a processor has encountered only the words "Have the students who missed the exam . . . ", it cannot know whether the word "have" is an auxilliary, as in the interrogative a, below, or a main verb, as in the imperative b:

(22) a. Have the students who missed the exam taken the make-up?
b. Have the students who missed the exam take the make-up!

It can only resolve this question when the next word, "take" or "taken", is processed.

A predictive algorithm that happens to have chosen the wrong rule for expansion in analysing one or the other sentence up to this point should be able to take advantage of the work that it has done in parsing the NP "the students who missed the exam", since its analysis is identical in both cases. But blind backtracking back to the state where the wrong choice was made will lose this information. Similar arguments apply, mutatis mutandis, to a blindly backtracking shift-reduce parser.

To take advantage of this characteristic of the search space, we must record the fact that a noun-phrase has been found spanning words two to seven of the sentence, together with the analysis. Any other analysis requiring a noun-phrase starting at position 2 can then simply use the result of the earlier analysis.

The data structure in which this information is stored is called a "chart". All of the above exponential algorithms can be made to parse in worst case time n^3 by the use of a chart. Many of the most efficient techniques that are known for computational natural language systems are of this kind, including the Cocke/Kasami/Younger algorithm (which is bottom-up), and Earley's 1970 algorithm, which is a refinement of the left-corner technique. This latter algorithm gains particular efficiency by the use of an "active" chart, in which not only complete constituents are stored, but also *incomplete* constituents, corresponding to the "dotted rules" used in the predictive algorithms.

$\mathcal{O}(n^3)$ performance can still be impracticably expensive, especially if worst case is also average case, as it is in practice for the CKY algorithm, and if n can get large, as it does in compiling programming languages, so there is some interest in briefly considering a further class of algorithm that can have computational costs *linear* in n for a subset of grammars. So far we have only considered general-purpose nondeterministic algorithms. But we can in principle consider more grammar-specific algorithms that tell us more about the action(s) that should be taken in particular states. If the grammar allows us to make this machine so specific that it is entirely deterministic, then we shall have eliminated the oracle entirely. Of course, we cannot by definition do this for a truly nondeterministic grammar. But if the apparent nondeterminism in the grammar happens to be resolvable by looking ahead at a bounded number of immediately adjacent symbols in the string, then such such a deterministic algorithm exists. One particular variety of deterministic parser of this kind is called the $LR(k)$ parser, because it does a Left-to-right scan, delivers a Rightmost derivation, and uses a k symbol look-ahead. Since it is convenient to express the control for this algorithm as a matrix indexed by a) the state, b) the topmost element on the push-down stack, and c) the remaining input, such controls are refered to as "LR tables". Since such tables can be quite complicated, there is intense interest among computer scientists in the fact that there are known techniques for constructing such deterministic algorithms automatically, from the original grammar. (In fact, such techniques provide the basis for one of the most widely used methods for automatically generating compilers, and many programming languages are as a result in the class $LR(1)$ – that is, all nondetermism is resolvable by a one-symbol lookahead.

We know that natural languages are not $LR(k)$ languages from examples like the last example 22, as Marcus 1980 pointed out. That is, after the word "have" in that example, there is a non-determinism between an analysis according to which it is an auxiliary followed by a subject "the students . . . ", and one where it is a main verb followed by an object. This non-determinism is resolved by the word "take" or "taken". But because of the embedding property of English, the noun-phrase may be indefinitiely large – "the students", "the students who missed the exam", "the students who missed the exam that I set in the morning", and so on. There is therefore no bound upon the size k of the lookahead that this construction would require.

Nevertheless, we shall see below that natural lexical ambiguity does in fact look as if it might be resolvable by examining strictly bounded number of neighbouring symbols, and it is also the case that the techniques for automatically building LR tables deliver coherent (albeit nondeterministic) results for non-LR grammars, a fact that has led to a number of generalisations of the technique. It is therefore interesting to ask whether some modification of the idea can be applied to natural languages.

Marcus 1980 describes an LR-like parser with three distinctive characteristics. First, he allowed lookahead of up to three items – that is, $k = 3$. Second, in order to handle the nondeterminism just discussed, which manifestly does not cause human processors any noticeable difficulty, he allowed the lookahead window to include NPs (but no other constituent types), as well as words. Third, he assumed that the equivalent of the LR table was inexact. The residual nondeterminism was then to be resolved by default.[21] Marcus then claimed that while English itself remained a non-deterministic language, the *parser* was only complete with respect to the deterministic fragment thus defined. (This also is a technique that has been used in LR-based compilers). On occasion the defaults would engender the classical garden path effects.

These moves restore the oracle to a position of importance, and in some sense compromise the original attraction of the LR algorithm. Nevertheless, deterministic parsing remains an attractive idea for practical applications, particularly where large volume is at a premium, and where the work of the oracle can be deferred by building "underspecified" syntactic derivations, as in the approach known as "D-theoretic" parsing (Marcus et al. 1983).

6. Summary: The Psychological Algorithm

Where among this maze of parameters is the psychological algorithm situated? Is it top-down or bottom up, or a mixture of the two? Does it use a chart, or an LR table, or not?

Evolutionary considerations prejudice us against any theory of parsing that requires large amounts of extra apparatus, over and above the grammar and the corresponding automaton, for reasons similar to those that led us to reject the possibility of parsing by covering grammar. To that extent the bottom-up algorithm continues to exercise a considerable appeal. However, it passes a considerable

[21] Certain other residual sources of nondeterminism – in particular, noun-noun compounding – were to be resolved by other mechanisms discussed below.

burden on to the oracle.

By contrast, the predictive algorithms such as Left-Corner, which ask less of the oracle, import extra mechanisms such as dotted rules to control the predictive search. This may not be too bad: such mechanisms are likely to be quite generally needed for a variety of hierarchically organised behaviours such as planned actions, and even for such lowly motor tasks as such as getting objects out of boxes and reaching around obstacles.

Among the predictive algorithms, the simple top-down recursive descent algorithm has a strike against it that makes it perhaps the least psychologically plausible of all. Grammars that include left-recursive rules of the form $A \rightarrow A\ B$ introduce a danger of infinite recursion in the leftmost-first version of the algorithm, as the parser predicts an A, which predicts an A, which predicts .... (Nor can we get round this with some other parsing regime, such as rightmost-first, since *some* kind of recursion will cause similar versions for *all* recursive descent regimes.) Guarding against this possibility complicates the algorithm and its operations in ways that seriously stretch psychological credibility.

For similar reasons, the LR parser, which also complicates the algorithm by introducing a language-specific LR table, seems to be implausible in evolutionary and developmental terms, since like a covering grammar, it needs to be recomputed every time a modification is made to the grammar. When we come to discuss the oracle below, we shall see that there are other ways in which the consequences of lexical nondeterminism can be minimised, and that the structural nondeterminism that causes LR tables for natural languages to be inexact is the major problem for the oracle.

The psychological relevance of the chart is also questionable. The fact that naive subjects and everyday language users are so rarely aware of syntactic ambiguity, together with the existence of garden path sentences, makes it unlikely that human language processors have access to such a powerful memory device, and suggests that they depend upon an oracle that usually gets it right first time. Most of the psychological theories discussed below make this assumption.

Unfortunately, until we know more about that oracle, just about the only further conclusion we can state with any confidence is that the grammar itself does seem to favour algorithms that go from left-to-right, because of the tendency to mark left boundaries. (This is probably a major reason why simple left-corner parsing works as well as it does). This conclusion hardly comes as a surprise. In other respects, natural grammars look surprisingly *un*helpfully designed for *all* of these algorithms. For example, for all left-to-right algorithms, processing right-

branching structures causes the stack to grow, while left-branching structures keep the stack small. (In the case of bottom-up algorithms, the items on the stack are unattached constituents. In the case of top-down algorithms, they are dotted rules, or the equivalent.) In view of the earlier observation that limitations on center-embedding seem likely to arise from stack limitations, one might therefore have expected natural grammars to minimise this burden by favouring left-branching constructions. In fact, however, right branching appears to be widespread among the languages of the world, according to the usual linguistic analyses. We shall return to this curious fact below.

C: THE ORACLE

1: STOCHASTIC TECHNIQUES.

One of the most important and useful techniques for reducing nondeterminism, at least at the level of the lexicon, is also one of the oldest, with origins in Information Theory that predate Artificial Intelligence and even Generative Grammar itself.

It has been known since the work of Shannon and Weaver in the early fifties that natural language text can be modelled to a close approximation by stochastic finite-state Markov processes. Of course, this fact tells us nothing about the grammar that in part determines the stochastic behaviour, as Chomsky 1957, p.17 influentially pointed out. But it is a consequence of this fact that the part of speech corresponding to any given word in a sentence is in principle highly predictable solely on the basis of the parts of speech of the two or three preceding words, solely on the basis of the stochastic model, and without benefit of higher-level analysis.

This observation does not in itself solve the problem, for the finite automata in question are large, and the sheer volume of distributional information that they depend upon is gigantic. For this reason, stochastic and probablistic techniques played very little part in the early development of the computational natural language processing techniques discussed above. However, it was always likely that such techniques would eventually prove useful in drastically reducing lexical ambiguity (cf. Chomsky 1957, p.17, n.4). Two developments at the start of the eighties allowed this possibility to be realised. The first was that computing machinery became fast enough, big enough in terms of memory, and cheap enough, to actually try what had become at the time a rather unfashionable approach. The second crucial development was the invention of some quite new algorithms

for constructing such "hidden" Markov models automatically, via re-estimation procedures, on the basis of exposure to fairly large amounts of text annotated with part-of-speech tags by hand, whose assembly was also crucial to this development. With the use of such techniques, it is possible to disambiguate part-of-speech with a reliability of around 97% for text where all words are known (Brill 1994).

These figures should be kept in perspective. They mean that roughly every other sentence will include a word for which the preferred part of speech is incorrect. It should also not be forgotten that choosing part-of-speech on the basis of basic or unigram frequency alone yields around 93% correct disambiguation. However, the efficiency of all of the algorithms considered above can be greatly increased by using part-of-speech disambiguation as a component of the oracle in an "n-best" parser architecture, whereby the parser only considers a small number of alternative part-of-speech categories for ambiguous words, in the order of their likelihood according to the model down to some threshold. For example, it is likely that such a model would make the simple bottom up algorithm more efficient by excluding the possibility of the word "men" being a noun phrase rather than a noun in the earlier example 17 "The men walk", thereby avoiding the creation of a redundant S "men walk". That is, it is likely that SHIFT/SHIFT conflicts can be largely eliminated for the shift-reduce algorithm by this method.

Once the Markov model is trained, it is computationally extremely cheap, because it is a finite state device. It is therefore likely that such stochastic oracles and the n-best architecture will be standard in large coverage natural language processing programs in future. It will be interesting in this period to see how a number of proposals to generalise the techniques will fare, including proposals to disambiguate word senses as well as part-of -speech, and a number of computationally less resource-intensive rule-based training methods. There are also a number of interesting proposals to embed such mechanisms in neural networks and other "connectionist" devices, whose computational implications go beyond the scope of the present essay. Although there have also been related proposals for stochastic Context-free Grammars, it seems much less likely that these techniques will cope with the other variety of syntactic nondeterminism, arising from structural ambiguity in examples like 1 "put the book on the table in your pocket", and that some of the other techniques discussed below will be needed as well.

The psychological relevance of stochastic lexical disambiguation is less clear. On the one hand, there is plenty of evidence that humans and other animals are sensitive to statistical regularities across extended periods. While it seems somewhat less likely that they can compute higher-order statistics of the kind

required to train Hidden Markov Models, some of the rule-based alternatives may be more psychologically plausible. Nevertheless, it should be borne in mind that the high redundancy of real text which these techniques exploit is in actual fact a by-product of the grammar and the content of the text. The possibility remains open that it is at these higher levels that humans filter out nondeterminism.

2: STRATEGY-BASED TECHNIQUES

One important group of proposals for the psychological oracle come from the psychologists themselves. It is pleasant to report that they have been enthusiastically taken up by computational linguists attempting to increase efficiency for natural language parsers. These proposals originate in work by Fodor, Bever and Garrett, 1974, who proposed that the garden path effect of sentences like 3, "The horse raced past the barn fell" arose from the use by the parser of the "Canonical Sentoid Strategy", which esentially said that if the parser ever encountered an NP followed by what could be a tensed verb, then it should act on the assumption that both elements are the subject and tensed verb of a "sentoid" or underlying clause. The claim was that this strategy, which resolves the lexical nondeterminism of the word "raced", is usually beneficial, but in this case commits the processor to a blind alley.

The Canonical Sentoid Strategy was only one of a number of such heuristics that supposedly guided the parser through the search space. To a modern eye, rules like this look uncomfortably as if they duplicate the rules of the grammar itself – such as rule 1 of the ATN 1 – and in fact later strategy-based approaches all attempt to separate the heuristic and grammatical components, via the assumption of more "surfacey" theories of grammar, including the ATN, rather than the framework of the Standard Theory assumed by Fodor et al.

One important early modification of the strategy idea was presented by Kimball 1973 who proposed seven very general principles of "surface structure parsing". Kimball was primarily a linguist, and was influential in the development of the Base Generation Hypothesis concerning the competence grammar. He was therefore in a position to make a much cleaner separation between the grammar, the algorithm, and the oracle than Fodor et al. had been.

Many of Kimball's principles are properly regarded as properties of grammar or the algorithm, rather than the oracle. But two of them seemed to capture a large number of phenomena concerning the resolution of structural ambiguity, and had the attraction of really looking like properties of a *parser* rather than misplaced

rules of grammar. In fact Kimball offered a number of conjectures as to *why* the parser might have these properties, in terms of comparitive structural complexity, time to completion, and so on. Close relatives of these two principles turn up in a number of proposals which are often called "Two Factor" theories of the oracle. Kimball called them "Right Association" and "Closure," but it will be convenient to refer to the latter as "Minimal Attachment," the name by which a more specific but related principle goes in Lyn Frazier's theory, for it is in Frazier's 1978 work that these two principles reached their most refined form.

The detailed formulation of these principles need not concern us here, since our principal concern is with their computational interpretation. It will suffice to say that the first principle, Right Association, captures the fact that in sentences like the following, the prefered analysis seems to be the one where the VP modifier "last Tuesday" is predicated of Nixon's death rather than Bill's announcement.

(23) Bill said Nixon died last Tuesday.

Minimal Attachment, on the other hand, captures the fact that in the following sentence there is a preference for the analysis where the modifier "for Susan" modifies carrying the package rather than the package:

(24) Tom carried the package for Susan.

It is this principle that captures the fact that we go wrong in Bever's garden path sentences (although both Kimball and Frazier need further apparatus to explain why recovery is impossible). That is to say that in the following fragment of the sentence 3, exactly the same factors favour the attachment of the VP "raced past the barn" to the NP as subject, rather than as a Past-participial modifier attached to the NP itself.

Following Frazier's work, which initiated a considerable body of experimental work supporting the reality of these effects, a number of computational accounts attempted to capture them in terms of simple properties of parsers. Among these, Wanner 1980 showed that Right Association could be captured in the ATN in terms of a global ordering of transition types equivalent to rule ordering in a more conventional grammar. Perhaps the most elegant computational account was provided by Pereira 1985, who showed that Right Association was equivalent in a shift-reduce parser to resolving all SHIFT/REDUCE conflicts in favour of shifting, and resolving all REDUCE/REDUCE conflicts in favour of the reduction that removes most nodes from the stack. (Pereira's parser was in fact an LR(1)

parser with a not-fully-deterministic LR table, which these two heuristics were used to render fully deterministic.)

However, at the very time these elegant computational accounts were being put forward, the very basis of the effects that they were to explain was being called into question by certain other experimental approaches with close links to computational models.

3: Lexical Preferences

First, it was immediately noticeable that effects like Minimal Attachment were sensitive to the particular lexical items involved. Thus, Minimal Attachment makes the right prediction in a, below, but in b, it makes the wrong prediction:

(25) a. The woman positioned the dress on the rack.
b. The woman wanted the dress on the rack.

One effect of the move to Base Generation by Bresnan and others had been to emphasise Lexicalism in the theory of grammar, and to include more information concerning subcategorisation or function-argument relations into the lexical entries for verbs. This proposal, originally motivated by purely linguistic considerations, suggested a natural way to capture the phenomenon in the performance theory, via differential ordering of the lexical entries determining the various subcategorisation possibilities for different verbs in LFG (Ford et al. in 1982). Such an ordering on lexical entries might also be produced dynamically, as a result of an n-gram based lexical disambiguator of the kind described above, with the lexical categories taking the part of the part-of-speech tags. (This approach is currently being pursued in n-best parsers for other varieties of lexicalist grammars, including TAG and CCG.)

The lexicalist account of attachment preferences contributes a major piece to solution of the puzzle, and has been further refined by Pritchett 1992, Trueswell et al., 1994, and Gibson 1994, in press. However, it was also clear that similar effects inconsistent with Minimal Attachment could be found for the *same* verb in combination with different arguments, suggesting that something more was involved.

4: Incremental Semantic Filtering

In his first identification of the garden-path sentences, Bever noted that the effect was sensitive to content. Thus while a, below, is a standard garden path, in b, the effect is greatly reduced:

(26) a. The doctor sent for the patient arrived
b. The flowers sent for the patient arrived

The observation suggests that human processors can take into account the relative plausibility of doctors versus flowers as agents of the action of sending for something or someone. In fact, it suggests that they can take account of this plausibility information early on in the course of processing, *even before the spurious clause is complete*. (Otherwise we would have no explanation for the persistence of the garden-path effect in case a). This and a number of other experimental phenomena led Marslen-Wilson et al. 1978 to argue that the primary source of disambiguating information drawn upon by the oracle was semantic.

This argument for semantically "interactive" parsing aroused surprisingly strong opposition among psychologists at the time. The reason was greatly to do with doubt and confusion concerning the computational feasibility of the proposal. This is surprising, for there was already available a very vivid existence proof of something like the necessary mechanism, embodied in the well-known natural language understanding program of Winograd 1972, which interpreted questions and commands in a simple simulated world of toy blocks on a table, using a dynamically changing discourse model.

Winograd proposed, not only that the parser should pay attention to semantic requirements such as animacy that verbs like "send for" imposed upon their subjects, but also that attachment ambiguities of the kind found in his domain in instructions like the following should be resolved simply by adopting whichever analysis successfully referred to an entity in the world:

(27) Put the block in the box on the table

The program resolved this ambiguity on the basis of whether there was in the discourse model a block unique by virtue of being the only block or a recently mentioned block, plus a box similarly unique by virtue of being on the table, or whether instead there was a block unique by virtue of being in a unique box, plus a unique table. Thus Winograd's proposal was that the oracle worked by semantically "filtering" the alternatives proposed by the parser.

This definition of semantically interactive parsing is often called "weak" interaction, because it assumes that the grammar and the algorithm propose well-formed analyses entirely autonomously, and that the oracle merely disposes among the alternatives, killing off or interrupting those analyses that are either semantically incoherent (flowers being unqualified to send for things) or referentially unsuccessful (there being no block in the box).

Part of the resistance to Marslen-Wilson's proposal arose from a confusion of the weakly interactive or filtering processor with an alternative proposal for "strong" interaction, according to which an autonomously produced interpretation could supposedly manipulate the syntactic component itself, activating or suppressing rules, and thereby determining in advance which analysis was built. The sense in which it is possible to build semantic analyses independently of syntax, and the question of what this bought the processor if it still had to build the syntactic analysis to check its predictions were never very clearly resolved, and it is not clear whether any truly strongly interactive processor was ever built. Such a model was strenuously (and in the view of the present author, correctly) opposed by Fodor 1983, on the grounds that it violated the modularity criterion – in effect, that it really did not qualify as an explanation at all. However, Fodor himself pointed out that the weak or filtering interaction was entirely modular (1983, see p.78 and 135).

Crain and Steedman (1985) argued that some version of the weakly interactive oracle proposed by Winograd could account not only for attachment preferences, but also for the phenomenon of garden pathing, or unrecoverably incorrect attachment preferences. They made the following further assumptions about the processor.

First, they argued from minimal pairs like 26 that filtering had to occur very early in the analysis, well before the end of the clause. It followed that semantic interpretations had to be constructed in parallel with syntactic analysis, much as the earlier DCG notation suggests, and that partial interpretations, corresponding to syntactically incomplete fragments such as *the flowers sent for* ..., must be available.

Second, they argued that, unlike syntactic well-formedness, semantic and referential anomaly was *relative* rather than all-or-none. It followed that, unlike structural strategy-based parsers, the weakly interactive processor had to produce all partial analyses at a choice point, complete with partial interpretations, and then reject or interrupt all but the best before continuing the "best-first" search.

Third, they claimed that when sentences are processed in isolation and out of context, as in the typical psycholinguistic experiment of the day, the oracle chooses that analysis whose (semantic and referential) pragmatic presuppositions were easiest to "accomodate", or add to the discourse model. In the case of garden path examples like "The horse raced past the barn fell", they argued that the single presupposition that there was a unique horse was easier to accomodate than the presupposition that there were many horses, one of which was distinguished by

the property that a hitherto unknown agent caused it to race somewhere. They presented experimental evidence that attachment preferences were under the control of referential context, by prefacing minimal pairs of locally attachment-ambiguous target sentences by contexts which either established two women respectively with and without a distinguishing property, or one woman with that property.

(28) CONTEXTS:

1. A psychologist was counselling two women. He was worried about one of them, but not about the other.

2. A psychologist was counselling a man and a woman. He was worried about one of them, but not about the other.

TARGETS:

1. The psychologist told the woman that he was having trouble with *her husband.*

2. The psychologist told the woman that he was having trouble with *to visit him again.*

Both target sentences have a local ambiguity at the word *that*, which is only resolved when the italicised words are encountered. Minimal attachment would predict that the second target would always cause a garden path. In fact however this garden path effect is eliminated when the sentence is preceded by the first context, which satisfies the presupposition of the relative clause analysis. And a garden path effect is induced in the first target when it is preceded by the same context, because by the same token it fails to support the presupposition that there is a unique woman. These authors also show that certain predictions concerning the effect of definiteness on garden paths in the null context. The experiments were repeated and extended with improved materials by Altmann (Altmann and Steedman 1988).

There is continuing disagreement on the question of what *other* mechanisms may also be involved in the oracle. Perhaps the most interesting questions for the computer scientists to ask the psychologists is whether there is a psychological role for the low level stochastic mechanisms that have proved so effective in practical applications, and whether there is a residual role for structure-based strategies, or whether human processors work excusively at the level of semantics to achieve the same filtering effect. However, there is now fairly general agreement that the weak interaction plays an important role.

III: CONCLUSION

Many open questions remain concerning the exact computational nature of all three modules of the psychological processor. The explanatory coverage of the theory of competence grammar remains incomplete. The choice among the various available algorithms depends upon the further identification of the oracle, and to some extent upon further questions concerning the computational architecture itself. The range of resources that the oracle itself draws upon, remains an open question for further research.

Nevertheless, the results and notations of computer science make it it possible to see that everything we know experimentally about the psychological sentence processor is compatible with the expectation it will eventually be seen to an extremely simple, explanatory, and modular device in both computational and evolutionary terms. The working models that computational linguistics offers provide a proof of concept for systems involving extremely surfacey grammars, in which syntactic composition and semantic composition are very closely related indeed, and in which such grammars can be used directly by algorithms that may use very little more than the minimal automaton characteristic of the class of grammars in question and the simplest language independent algorithm, working under the guidance of an oracle that exploits to the full the weak interaction that such semantically transparent grammars allow.

A number of puzzles remain. One is that the predominantly right branching structure evinced by natural languages according to most theories of grammar still appears to maximise the working memory requirements for left-to-right parsers. Another is that the same predominance of right-branching seems to require some extra apparatus for the weakly-interactive oracle to work. Right branching grammars like the one introduced at 4 are immediately compatible with incremental interpretation *of constituents*. However, we have seen that interpretations for left-prefixes like "The flowers sent for ...," which are not constituents under such grammars, appear nevertheless to be available to the parser. While interpretations for non-constituents can be built under any grammar, doing so requires extra structures that are not available from the grammar, and thus compromises the strictest interpretation of the Strong Competence Hypothesis.[22] It is likely that computational linguistics will continue to play an important part in research towards a

[22]I address this question elsewhere – see Further Reading, below.

resolution of these further problems.

FURTHER READING

I have assumed familiarity above with the basic ideas of Generative Grammar. Most introductory texts on linguistics, psycholinguistics, and computational linguistics cover this material. I have also assumed a nodding aquaintance with Formal Language Theory, in particular the Chomsky hierarchy. This and much other useful material is covered by Partee et al, 1990. The standard reference on the subject, Harrison 1978, is more technical. Allen 1987 is an excellent introductory text to the broader field of Computational Linguistics, and the indispensible collection edited by Grosz et al. 1986 gathers a number of key research papers in this broader area, including several of those cited above. The best gentle introduction to the Computer Scientists view of Algorithmics in general is Harel's 1987.

A trustworthy guide to the characteristics of a number of alternative parsing regimes is Kay 1980, from whom I have borrowed the general tri-modular view of the processor, generalizing his notion of an "agenda" to the notion of the oracle presented above. The grammars, algorithms, and oracles decribed above are all very readily implementable in the programming language Prolog, and the elegant text by Pereira and Shieber 1987 provides all the help that is needed. The most complete accessible account of the ATN, including many important features not discussed here, is Woods 1973. The question of the automata theoretic power required for Natural Languages is helpfully discussed by Gazdar 1988, and Joshi et al., 1991. The important collection edited by Dowty et al. 1985 brings together a number of computational and psycholinguistic papers, including several discussed above. I have not attempted to survey the huge and rapidly changing experimental psycholinguistic literature on this topic here.

The present paper is a companion to my "Computational Aspects of Grammar", in which the question of the nature of the competence grammar itself, and its relation to the problem of incremental semantic interpretation and weakly interactive parsing under a very strict reading of the Strong Competence Hypothesis, are investigated in greater depth. Both papers are intended to be read independently, and as a consequence certain sections concerning notation and the theory of natural language grammar are common to both.

ACKNOWLEDGMENTS

Thanks to Michael Niv and Mike White for reading and commenting upon the draft. The work was supported in part by NSF grant nos. IRI90-18513, IRI91-17110, and CISE IIP, CDA 88-22719, DARPA grant no. N00014-90-J-1863, and ARO grant no. DAAL03-89-C0031.

BIBLIOGRAPHY

Ades, A. & M. Steedman. (1982) On the order of words, *Linguistics & Philosophy*. 4, 517-558.

Allen, J. (1987) *Natural language understanding*, Menlo Park CA: Benjamin Cummings.

Aho, A. (1969) Nested-stack automata. *Journal of the Association for Computing Machinery,* **16**, 383-406.

Aho, A. & S. Johnson. (1974) LR parsing. *Computing Surveys,* **6**, 99-124.

Altmann, G. & M. Steedman. (1988) Interaction with context during human sentence processing *Cognition*. 30, 191-238

Barton, G.E, R. Berwick, & E. Ristad. (1987) *Computational complexity and natural language*. Cambridge MA: MIT Press.

Berwick, R. & A. Weinberg. (1984) *The grammatical basis of linguistic performance*. Cambridge MA: MIT Press.

Brame, M. (1978) *Base generated syntax*. Noit Amrofer, Seattle WA.

Bresnan, J. (1978) A realistic transformational grammar, in M. Halle, J. Bresnan, & G. Miller (eds.), *Linguistic structure and psychological reality*. Cambridge MA: MIT Press.

Bresnan, J. (ed.). (1982) *The mental representation of grammatical relations*. Cambridge MA: MIT Press.

Brill, E. (1994) A report of recent progress in transformation-based error driven learning, *Proceedings, ARPA Human Language Technology Workshop*, Plainsboro, NJ, March 1994. Palo Alto CA: Morgan-Kaufmann. (In press)

Chomsky, N. (1957) *Syntactic structures*. Mouton, the Hague.

Chomsky, N. (1981) *Lectures on government and binding*. Dordrecht: Foris.

Crain, S, & M. Steedman. (1985) On not being led up the garden path: the use of context by the psychological parser, in Dowty et al. (1985), 320-358.

Dowty, D., L. Kartunnen, & A. Zwicky, (eds.). (1985) *Natural language parsing: psychological, computational and theoretical perspectives.* ACL Studies in Natural Language Processing, Cambridge University Press.

Earley, J. (1970) An efficient context-free parsing algorithm, *Communications of the Association for Computing Machinery,* **13,**, 94-102.

Fodor, J. (1983) *The modularity of mind.* Cambridge MA: MIT Press. Cambridge MA.

Fodor, J, T. Bever, & M. Garrett. (1974) *The psychology of language.* McGraw-Hill, New York NY.

Ford, M, J. Bresnan, & R. Kaplan. (1982) A competence-based theory of syntactic closure, in Joan Bresnan (ed. *The mental representation of grammatical relations.* Cambridge MA: MIT Press.

Frazier, L. & J. Fodor. (1978) The sausage machine: a new two-stage parsing model, *Cognition*, 6, 291-325.

Gazdar, G. (1988) Applicability of indexed grammars to natural languages', in U. Reyle, & C. Rohrer, (eds.), *Natural language parsing and linguistic theories*, Dordrecht: Reidel, 69-94.

Gazdar, G., E. Klein, G. Pullum, & I. Sag. (1985) *Generalised phrase structure grammar*, Blackwell, Oxford.

Gibson, E. (1994, in press) *Memory limitations and linguistic processing breakdown*, Cambridge MA: MIT Press, (to appear).

Grosz, B., K. Sparck Jones, & B. Webber. (1986) Readings in natural language processing. Palo Alto CA: Morgan-Kaufmann.

Harel, D. (1987) *Algorithmics: the spirit of computing*, Readin MA: Addison-Wesley.

Harrison, M. (1978) *Introduction to formal language theory*, Reading, MA: Addison-Wesley.

Joshi, A., K. Vijay-Shanker, & D. Weir. (1991) The convergence of mildly context-sensitive formalisms, in P. Sells, S. Shieber & T. Wasow, (eds.), *Processing of linguistic structure*, Cambridge MA: MIT Press. 31-81.

Joshi, A. & Y. Schabes (1992) Tree adjoining grammars and lexicalized grammars, in M. Nivat & M. Podelski (eds.), *Definability and recognizability of sets of trees.* Princeton: Elsevier.

Kay, M. (1980) Algorithm schemata and data structures in syntactic processing, CSL-80-12, Xerox PARC. (Reprinted in Grosz et al. 1986).

Kimball, J. (1973) Seven principles of surface structure parsing in natural language, *Cognition*, 2, 15–47.

Marcus, M. (1980) *A theory of syntactic recognition for natural language*. Cambridge MA: MIT Press.

Marcus, M, D. Hindle, & M. Fleck. (1983) D-theory: talking about talking about trees, *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge Mass, June, 1983, 129-136.

Marslen-Wilson, W, L. Tyler & M. Seidenberg. (1978) The semantic control of sentence segmentation, in W.J.M. Levelt & G. Flores d'Arcais (eds.), *Studies in the perception of language*. New York NY: Wiley.

Montague, R, (1974) Formal philosophy: papers of Richard Montague, ed. by R.H. Thomason, New Haven: Yale University Press.

Niv, M. (1993) Resolution of syntactic ambiguity: the case of new subjects, in *Proceedings of the 15th Annual Meeting of the Cognitive Science Society*, Boulder CO, June 1993, XXX-XXX.

Oehrle, R., E. Bach & D. Wheeler, (eds), *Categorial grammars and natural language structures*. Dordrecht: Reidel.

Partee, B., A. ter Meulen, & R. Wall. (1990) *Mathematical Methods in Linguistics*, Dordrecht: Kluwer.

Pereira, F. (1985) A new characterisation of attachment preferences, in Dowty et al. (1985), 307-319.

Pereira, F. & S. Shieber. (1987) *Prolog and natutal language understanding*. Chicago: CSLI/University of Chicago Press.

Peters, S. & R. Ritchie. (1973) On the generative power of transformational grammars, *Information Science,* **6**, 49-83.

Pollard, C. & I. Sag. (1994) *Head-driven phrase structure grammar*. Chicago: CSLI/University of Chicago Press.

Pritchett, B. (1992) *Grammatical competence and parsing performance*, Chicago: University of Chicago Press.

Slocum, J. (1981) A practical comparison of parsing strategies, *Proceedings of the Nineteenth Annual Meeting of the Association for Computational Linguistics*, Stanford CA, June 1981, 1-6.

Trueswell, J. C., Tanenhaus, M. K., & Garnsey, S. M. (1994) Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory & Language*, in press.

Wanner, E. (1980) The ATN and the Sausage Machine: which one is baloney? *Cognition*, 8, 209-225.

Winograd, T. (1972)*Understanding natural language.* Edinburgh University Press.

Woods, W. (1970) Transition network grammars for natural language analysis, *Communications of the Association for Computing Machinery,* **3**, 591-606. (Reprinted in Grosz et al. 1986).

Woods, W. (1973) An experimental parsing system for transition network grammars. *Natural language processing*, Courant Computer Science Symposium 8, ed. by R. Rustin, 111-154. New York: Algorithmics Press.