COGNITIVE SCIENCE **14**, 313–353 (1990)

Race-Based Parsing and Syntactic Disambiguation

SUSAN WEBER MCROY AND GRAEME HIRST
University of Toronto

We present a processing model that integrates some important psychological claims about the human sentence-parsing mechanism; namely, that processing is influenced by limitations on working memory and by various syntactic preferences. The model uses time-constraint information to resolve conflicting preferences in a psychologically plausible way. The starting point for this proposal is the Sausage Machine model (Fodor & Frazier, 1980; Frazier & Fodor, 1978). From there, we attempt to overcome the original model's dependence on ad hoc aspects of its grammar, and its omission of verb-frame preferences. We also add mechanisms for lexical disambiguation and semantic processing in parallel with syntactic processing.

1. INTRODUCTION

Natural language sentences can have more than one structural interpretation or meaning because words and grammars for natural languages are ambiguous. However, when people process sentences they rarely notice more than one possible meaning, so ideally, a computational sentence processor should also produce a single interpretation, resolving ambiguities the way a person would.

We are indebted to Barbara Brunson for discussions of her interpretation of current Government-Binding theory into processing strategies, which we employed in the parser.

We also gratefully acknowledge Gerry Altmann, Fernando Pereira, Ben Macias, Stephen Pulman, Elan Dresher, Barbara Brunson, Diane Horton, Mark Catt, Chrysanne DiMarco, and Dan Lyons for their comments on this work, and Janet Dean Fodor and Lyn Frazier for their comments and suggestions about linguistic matters. Special thanks to Timothy Horton for helping us improve the clarity of this article. Many thanks also to Wendy Lehnert and the anonymous reviewers for their constructive suggestions on the content.

We acknowledge the financial support of University of Toronto through awards to the first author (University of Toronto Open Fellowships, Connaught Scholarships, and a Charles Gordon Heyd Fellowship) and the financial support of the Natural Sciences and Engineering Research Council of Canada.

Correspondence and requests for reprints should be sent to Susan Weber McRoy, Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4.

Over the past few years, a variety of human preferences for resolving attachment ambiguities has been identified, and a variety of principles has been proposed for describing them. These principles occasionally apply in the same situations, sometimes agreeing and sometimes disagreeing on what is the preferred attachment. Thus, if one actually tries to use them to build a sentence-processing system, one must either sacrifice some or try to incorporate them all and resolve any conflicts in such a way that redundancies are not a problem. Sacrificing some of them may entail sacrificing some correct parses, whereas the second approach offers a system designer both potentially wider coverage and the freedom to remain neutral on issues not yet resolved by current linguistic theories. In this article, we describe how one might bring together in one system several recent accounts of ambiguity resolution (for words and syntactic structure), and resolve conflicts in a robust and principled way.

The system we propose takes English sentences as input and processes them one word at a time, from left to right, incrementally constructing syntactic and semantic representations. The underlying parsing model is based on the general architecture of Frazier and Fodor's (1978) Sausage Machine model, attempting to take advantage of several of its psychological merits while overcoming its dependence on ad hoc aspects of its grammar. The model extends the basic Sausage Machine proposal by providing an explicit account of the interaction between syntactic and semantic processing that emphasizes the time it takes people to access or compute this information.

Our implementation of the model incorporates independent processing modules called *hypothesizers,* which suggest ways that the parser can combine any two syntactic objects. The hypothesizers encode particular attachment strategies and have access to syntactic and semantic information. To resolve any conflicts among their suggestions, the hypothesizers compute *time costs* and the suggestion with the lowest time cost wins the *race.*

We will now describe in greater detail the psychological principles we integrate. In Section 3 we analyze the original Sausage Machine proposal. Then, we introduce our own race-based approach to parsing in Section 4, showing how it accounts for the psychological principles. Section 5 describes our implementation of the model.

2. THE PSYCHOLOGICAL PRINCIPLES

Among the more important psychological claims about the human sentence-parsing mechanism are that processing is influenced by limitations on working memory and by a number of structural preferences, and that syntactic and semantic processing are performed in parallel.

2.1 Memory Constraints

The constraints on working memory, well described elsewhere (e.g., Klatsky, 1980), appear to have several important effects on parsing. In cases of ambiguity, memory limitations make it unlikely that structure-building decisions are delayed or multiple interpretations are maintained (Marslen-Wilson & Tyler, 1980). Memory limits also cause people to break sentences into more manageable segments for processing, influencing both the amount of effort required to process sentences and the preferred readings of ambiguous sentences. In particular, people's preferred interpretations of ambiguous sentences tend to reflect the limits on working memory. This effect has been called *Local Association* (LA) (Fodor & Frazier, 1980; Frazier & Fodor, 1978).

2.2 Structural Preferences

In addition to structural preferences that arise as a direct result of memory limitations, there appear to be some preferences that arise simply because they make the job of parsing sentences easier or more efficient. The best known of these are *Right Association, Minimal Attachment, Revision as Last Resort,* and *verb-frame preferences.*

Right Association (RA) is the preference for grouping incoming words with adjacent words to their left in a syntactic parse tree, that is, for attaching a new word to the lowest nonterminal node that is on the right-most branch of the current structure (Fodor & Frazier, 1980; Frazier & Fodor, 1978; Kimball, 1973; Wanner, 1980). Whereas LA *defines* the set of locally accessible attachments, RA specifies a *preference* for attachments that would complete the most local syntactic substructure. Thus, RA may reduce the number of attention shifts needed or the number of words that must be kept accessible in order to parse a sentence.

Minimal Attachment (MA) is the preference for incorporating a new word into a parse tree using the fewest possible new nodes (Fodor & Frazier, 1980; Frazier, 1978; Frazier & Fodor, 1978; Kimball, 1973). Support for MA comes from experiments by Frazier (1978) that show how MA operates in a wide variety of constructs. Discounting controversial examples, which rely on ad hoc aspects of Frazier's grammar, MA accounts for such well-noted phenomena as the preference for analyzing words or phrases as determiners, main verbs, direct objects, or conjuncts, rather than as the start of a complement or relative clause, and the preference for analyzing a clause as main, rather than subordinate.[1]

[1] Apparent conflicts between RA and MA seem to rely on ad hoc asymmetries in Frazier and Fodor's grammar, although MA (and RA) apparently can both be overridden by LA.

Revision as Last Resort (RALR) is the principle that existing structures should not be taken apart and rebuilt unless subsequent words require it, for example, because of semantic incoherence, pragmatic implausibility, or syntactic ill-formedness (Fodor & Frazier, 1980). RALR is not "strict determinism" in the sense of Marcus (1980), because it does not require that structures *never* be undone. Rather, RALR captures the general resistance to unnecessary change that is expected because human parsing is constrained by limited time and memory.

Verb-frame preferences (Connine, Ferreira, Jones, Clifton, & Frazier, 1984) are preferences that a verb has among its allowable arguments. These preferences help guide the initial analyses of sentences and influence their preferred readings (Clifton, Frazier, & Connine, 1984; Fodor, 1978; Ford, Bresnan, & Kaplan, 1982; Mitchell & Holmes, 1985). Verbs may prefer their arguments to be particular phrase structures such as NP, PP, or complement clause, to be particular thematic roles, or to have some particular features, and these preferences may be fixed or may vary with the dynamic context.

2.3 Parallel Processing of Syntax and Semantics

Finally, we note that human sentence processing involves many subtasks that proceed in parallel, such as syntactic and semantic processing. Syntactic and semantic structures are built for each constituent as it is encountered (Marslen-Wilson, 1975; Marslen-Wilson & Tyler, 1980; Marslen-Wilson, Tyler, & Seidenberg, 1978). Semantic processing may lead to priming, thereby influencing subsequent structural decisions. Nevertheless, syntactic preferences appear to persist in the absence of semantic information, and sometimes despite its presence (Ferreira, 1986; Ferreira & Clifton, 1986; Ford, Bresnan, & Kaplan, 1982; Holmes, 1984; Rayner, Carlson, & Frazier, 1983). Thus, it is not the case that semantic information will always be used, let alone used to the exclusion of purely syntactic preferences as suggested by some researchers, for example, Crain and Steedman (1985).

3. THE FRAZIER AND FODOR PROPOSAL

To account for many of the psychological issues described in the previous section, Frazier and Fodor (1978) proposed a sentence-parsing model, which they called the "Sausage Machine" (SM), that is appealing from both a psychological and a computational point of view, despite weaknesses that killed interest in it before it could be fully tested.

3.1 Overview of the Sausage Machine

The SM is a two-stage device whose first stage is limited to looking at the next five or six words of input and building a representation of them. As the first stage completes these representations (phrases or clauses), it passes

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

Figure 1. Sausage Machine grammar.

them on to the second stage, forgetting what it has done as it begins to build the structure for the next set of words. The second stage takes the *packages* as it receives them from the first stage and combines them into a complete parse tree.

The motivation for two stages is the fact that although working memory appears to be very limited in the number of units it can store at once—"seven plus or minus two"—it can apparently store more information in its limited space by "chunking" it or storing it as structured units (Klatsky, 1980). The model differs from earlier two-stage parsing models (see Frazier & Fodor, 1978, p. 292, for a discussion) in that the units produced by its first stage are determined by their size, rather than their syntactic shape, and usually with limited knowledge of the overall structure of the sentence as it is being built.[2]

Processing in both stages proceeds by first allowing the input to trigger grammar rules and then using the partially satisfied rules to generate expectations about what is to follow. Each stage is governed by the principles of RA, MA, and RALR, but the proposal leaves open most engineering issues. However, the model does assume a context-free grammar in which favored attachments use the fewest rules and the fewest levels of intermediate nodes. Fodor and Frazier (1980) suggest that grammar rules be

> accessed in parallel and selected in terms of the outcome of a "race"—the first rule or combination of rules that successfully relates the current lexical item to the phrase marker dominates subsequent processing. (p. 434)

In this context, being "first" simply means "requiring the least number of grammar rules."

For instance, the model assumes its grammar captures the modification of a verb being preferred over the modification of a noun, by representing the latter attachment with two rules, whereas the former requires only one, as in Figure 1. This assumption about the grammar is ad hoc because it makes a distinction that is not required by the theory of context-free grammars. However, given such a grammar, a parser looking for the preferred attachment need only count the number of rules it would have to use for each alternative, and then choose an attachment from among those that require the fewest. Consider Sentence (1) for example:

---

[2] This use of the term *two-stage* is thus also different from that of Weinberg (1987), who described what might rather be called a *two-pass* system: The first pass builds a parse tree, the second establishes binding relations.

(1) Joe read the letter to Mary.

After the first stage has built a syntactic structure for "Joe read the letter" and receives the PP "to Mary," MA says the PP should be attached directly to the VP (which is the preferred reading). To have attached it to the NP would have required accessing the rule NP→NP PP and creating an extra NP node. However, in a longer sentence such as,

(2) Joe read the newspaper article, the card, the telegram, and the letter to Mary.

MA specifies no preference within the first stage because "read" will have been passed on to the second stage by the time the PP is input. In this case, the first stage will be forced to attach it to the NP, which would explain why this is the preferred attachment in (2).

3.2 The Appeal of the Sausage Machine

The SM is appealing from a psychological point of view, because it accounts for most of the issues discussed in Section 2 and appears to allow for the incorporation of verb-frame preferences and parallel semantic processing. The limited view of the first stage causes LA because nonlocal attachments are not seen and so are never considered. MA, RA, and RALR, although not strictly required by the SM model, make the SM more efficient in a psychologically plausible way. In particular, abiding by MA and RALR means that the SM will access a minimum number of rules and build a minimum number of structures. Similarly, because a parser following RA will always consider attaching a new word to the nonterminal node nearest the one it is already working on, RA reduces attention shifts between constituents in the SM. RA also tends to increase the size of first-stage packages by encouraging the first stage to incorporate a new word in the current package rather than the next one, hence reducing the strain on the second stage. The limited view of the first stage also explains the existence of attachments that abide by MA or RALR with respect to a package, but not the sentence as a whole. Finally, the SM provides a nice account of sentence complexity, including differences in the difficulty of certain center-embedded sentences.[3]

---

[3] For example, people find it easier to parse center embeddings as in Sentence (i) than ones as in Sentence (ii) (Fodor & Frazier, 1980; Frazier & Fodor, 1978; Wanner, 1980).

(i) The very beautiful young woman the man the girl loved met on a cruise ship in Maine died of cholera in 1972.

(ii) Women men girls love meet die.

Sentence (ii) is short enough that the first stage will try to handle it all at once. In contrast, the long noun phrases in Sentence (i) would force the SM to process them separately. Processing them separately may decrease the processing load and also prevent them from initially being misparsed as a simple conjoined structure, as predicted by MA (in agreement with the experimental results of Blumenthal, 1966).

From a computational standpoint, the SM is appealing because of its parallelism and its attempt to make the best use of time and information available to it. While the first stage is doing simple segmentations, the second stage can be resolving major attachments, and conceivably locating fillers for gaps, revising structures on the basis of semantic information, or adding nodes to keep track of obligatorily parallel constructions. Its processing strategy allows it to exploit expectations without needing to retract hypotheses as often as purely top-down systems.

3.3 Problems with the Sausage Machine

Despite its psychological and computational appeal, the SM model as originally described has at least one obvious flaw and several critical omissions. (See also Warner's, 1980, general critique.) To begin with, the SM's account of MA is much too dependent on ad hoc details of the grammar that Frazier and Fodor (1978) have assumed; specifically, it relies upon structures not generally supported by current linguistic theories and makes attachment decisions based on the exact number of nodes in these structures. (Church, 1980; Cottrell, 1988; Hirst, 1987; and Schubert, 1984, also express this criticism.) In addition, the original SM proposal omits important low-level details. More seriously, the model says nothing about how verb-frame information might be used, how revision would work, how semantics might fit in, or what effect time constraints or priming might have. Any realistic parsing model must account for these issues.

4. A RACE MODEL OF PARSING

Having presented the principles to be integrated, and the strengths and weaknesses of an earlier model that attempted to account for many of them, we are now ready to describe our own parsing model from a theoretical point of view. We will then describe our implementation of the model in Section 5.

4.1 Overview of the Race Model

The general framework of the sentence processor being proposed here is a two-stage parser inspired by the Sausage Machine (described in Section 3) and its account of memory constraints, RA, MA, and RALR. Our sentence processor extends the SM model in that it also accounts for time constraints, verb-frame preferences, and the interaction between syntactic and semantic processing, and specifies a revision mechanism. In addition, the model incorporates a more principled approach to grammar, derived from Chomsky's (1981) Government-Binding theory, and remains independent of aspects not constrained by the underlying grammatical theory, following only a generalized statement of Minimal Attachment (cf. Section 4.5.2).

4.2 Top-Level Processing

As in the original Sausage Machine model, processing in our parser proceeds primarily according to the expectations of syntax. The first stage incorporates each word, as it is received, into the current fragment of syntactic representation according to expectations of that fragment, selectional requirements of the new word, structural constraints and preferences, semantic information, and knowledge of possible revisions. The first-stage fragments are limited in size so as to respect the limits on the capacity of working memory. When a fragment approaches the capacity of working memory, the first stage looks for a good place to end it (e.g., a punctuation mark or a grammatical function word such as a preposition), passes the fragment to the second-stage processor, and begins a new fragment. The second-stage processor, running in parallel with the first stage, then incorporates each fragment it receives into a syntactic representation for the entire sentence according to exactly the same attachment criteria that the first stage uses. The two stages differ in operation only as a result of their working on different units; for example, only the first stage has to retrieve the words from the lexicon and initiate lexical disambiguation processes.

4.3 Underlying Grammar

The grammatical theory employed by the model is derived from Government Binding (GB) (Abney, 1987; Chomsky, 1981; Cowper, 1987; see Section 5.2). Our choice of GB was largely a matter of convenience, and is not intended to imply a belief in any special psychological status for the theory. Rather, our concern was that we use *some* well-known, well-founded theory in order that the implementation not be crucially reliant upon any ad hoc aspects of the grammar. In other words, we did not want a GB parser, but rather a parser that used GB. (See McRoy, 1987, for a discussion of how the parser differs from the GB parsers of Abney & Cole, 1985; Dorr, 1987; and Wehrli, 1984.)

Phrase structures and attachments are determined according to the grammar's syntactic structures (described in Section 5.2), expectations created by selectional information (e.g., constraints on arguments) stored in the lexicon, plus a limited set of phrase structure rules to handle modifiers.

4.4 Memory Constraints

In previous models, human memory limitations have been incorporated either by limiting the number of unattached syntactic or semantic objects that the parser can see (Church, 1980; Marcus, 1980; Pulman, 1986) or by limiting the number of ambiguities it may represent (Schubert, 1984), in either case without regard to the size of the objects. Schubert's model (1984) also incorporates "expectation potentials" that decay with distance, giving him something akin to LA.

Our sentence processor's account of memory constraints is carried over from that of the SM proposal. As in the original SM model, the first stage of our parser has a capacity of approximately seven words and builds a structure for these words without the benefit of remembering previous structures it has constructed, thus incorporating a strict account of the memory limitation data. Thus, like the SM, the model exhibits and explains LA, because nonlocal attachments are beyond the view of the first stage. Our model also resolves structural ambiguities immediately (as described below), so that attachment decisions are not delayed and multiple interpretations are not maintained.

4.5 Race-Based Decision Making

At any point in the parsing of a sentence the grammar may permit more than one attachment of a particular word or fragment. The two top-level parsing stages depend on decision mechanisms which determine how to proceed with a given piece of input. These are modelled in a psychologically plausible way, designed to capture the preferences of Section 2, in an elegant, unified framework based on time constraints.

Presumably the human parsing mechanism has the advantage of parallelism, and it might be expected that at any point in a parse alternative possibilities might be processed in parallel, with faster simpler processing dominating slower more complex alternatives. A central aspect of our model is this sort of "processing race." Since memory constraints make it unlikely that many parallel interpretations are maintained throughout the course of parsing (Section 2.1), we suggest that whenever there is some input to be assimilated there are processing races to determine how to proceed, and that the fastest, simplest conjecture will generally be adopted. In our conception, these races stop as soon as an interpretation incorporating the new input is established, so that any ambiguity is resolved on the spot before further input is processed. In addition to matching our expectations for parallelism, choosing the fastest conjecture sits well with the time-constrained realities of real-time language processing.

Describing our model at a neural or connectionist level might emphasize aspects of parallelism more directly; however, our implementation will be a predominantly serial model because such a system is easier to design, modify, and control. Moreover, in a serial implementation, specific decisions about the underlying machinery will not cloud the principles we endeavor to capture. To simulate the effects of parallelism, our implementation will calculate *time costs* for alternate conjectures, reflecting the time and difficulty of building and executing them (to be discussed in Section 5.1.2). The "winner" of the race, in this system, will be that with the lowest time cost.

Though there do not exist data to determine absolute time costs from actual human performance, we can at least intuit various comparative relationships that abide by the principles of Section 2:

- Lexical information is faster to identify than syntactic information, presumably because lexical retrieval is easier than tree traversal.
- Simpler attachments are faster to build than more complex ones, because it takes more time to build more structure.
- More local hypotheses are faster to recognize than more distant ones, because it takes more time to traverse more of a parse tree.
- Syntactic information is faster to access than semantic information, because tree traversal is easier than knowledge-base inference (cf. Ferreira, 1986; Ferreira & Clifton, 1986).
- Possibilities that have been primed, either by semantics or syntax, are faster to identify and build than unprimed ones (Frazer, Taft, Roeper, Clifton, & Ehrlich, 1984; Hirst, 1987).
- Possibilities that are more highly expected are faster to identify than ones that are less expected.

Surprisingly, this simple time-cost ranking of operations accounts for nearly all the preferences we looked at in Section 2. A characterization of the preferences in terms of these operations, combined with the above characterization of operations in terms of time-cost constraints, will enable us to characterize the preferences in terms of simple time-cost constraints. Thus, by basing its decisions on competing time costs, the model is able to provide a unified account of the preference principles. In the remainder of this section, we shall analyze the model's account of the individual principles and their relationship to time costs.

4.5.1 Verb-Frame Preferences. In previous models, lexical preferences, such as verb-frame preferences for arguments, have been incorporated by simply noting in the lexicon whether or not a particular argument is "preferred" (Shieber, 1983; Wilks, Huang, & Fass, 1985). Simply marking the existence of a preference does not allow for different preference strengths among different verbs or their arguments, and hence, these systems will occasionally err when all possible attachments have been marked as preferred. Schubert (1984) proposes a system in which preferences may be compared, but the details (such as whether it uses syntactic or thematic preferences) are left unclear.

In the proposed model, verb-frame preferences are taken to be thematic in nature; that is, they link semantic objects, not syntactic ones (cf. Wilks et al., 1985). These preferences vary in strength; the stronger the preference for an argument, the more highly expected it is, and hence the lower the time cost of an attachment that begins that argument. Verb-frame preferences differ from semantic or pragmatic preferences because they are assumed to be part of the lexicon rather than being retrieved or inferred from the knowledge base. Though we have said that lexical hypotheses will be faster than syntactic or semantic ones, some verb-frame preferences will be sufficiently weak to allow

other preferences to predominate. The exact level of strength where this change of dominance occurs must be determined experimentally, so we will consider only clear-cut cases (e.g., see the normative results of Clifton et al., 1984; Connine et al., 1984).

4.5.2 Minimal Attachment. MA has been defined as a preference for attachments that require adding the fewest nodes to attach a new word to the current structure; but, as we saw above, this definition is too dependent on assumptions about the underlying grammar. In addition, this definition can also make the system's account of the interactions among MA and other preferences implausibly rigid (Church, 1980; Wanner, 1980; Wilks et al., 1985) or force the system to delay attachment decisions (Shieber, 1983), which is also implausible. In contrast, Cottrell (1988) presents a spreading activation system in which structures involving fewer nodes become activated faster because it takes less time for the activation to spread through the representation. The structure gains strength with each new word that is added to it, eventually killing off its lagging (nonminimal) competitors. This formulation still requires that the grammar writer make sure that preferred structures involve fewer nodes, but it provides a natural explanation of why a structure having fewer nodes, and hence MA, is to be preferred.

Thus, we use a more general statement of MA: The attachment that can be computed most quickly (i.e., with lowest time cost) is the one that will be favored. At the beginning of Section 4.5, we claimed that the stronger an expectation for attaching a syntactic object, the lower the time cost of identifying the attachment, and the smaller the amount of additional structure necessary for attaching an object, the lower the time cost of executing the attachment. Thus, in the proposed model, to find the most minimal attachment of a constituent C is to find (in decreasing order of preference):

1. A place along the right edge of the current parse tree where C is expected (or a place where C expects the current parse tree);
2. A place along the right edge of the current parse tree where C is allowed;
3. A place along the right edge of the parse tree where C could be attached by first proposing some intermediate structure such as a phrase or a clause.

Note that the third choice really entails an ordered list of attachment possibilities in which expected structures are more minimal than merely allowed ones, and structures requiring fewer additional nodes are more minimal than ones that require more.

4.5.3 Right Association. RA specifies that given any ambiguity among possible minimal attachments, the one that links a word to adjacent words is favored over an attachment that links a word to words that are farther

away. To account for this preference, the proposed model associates a time cost with a more distant connection, because accessing more distant words (e.g., by tree traversal) takes longer. Similar results have been accomplished by considering attachments in right-to-left order in a list (or top-to-bottom order in a stack) (Church, 1980; Pulman, 1986; Shieber, 1983; Wanner, 1980; Wilks et al., 1985).

4.5.4 Semantic Preferences. In the proposed model, semantic interpretations are to be computed incrementally on a word-by-word basis, in parallel with syntax. Semantic processing interacts with syntactic processing only at certain points, for example, to resolve an ambiguity in attaching a phrase or a clause.

Here, "semantic" preferences are those that would create a description of an entity in the knowledge base or similar to one in the knowledge base. For instance, the semantic representation for a structure resulting from an attachment of a modifier is compared with objects the systems knows about and those it has seen in the past. Similarity to an object the system knows about (i.e., an object in the knowledge base) is taken as evidence in support of the attachment. Consequently, the time cost of the corresponding attachment is lower than that of an unexpected modifier, although still higher than that of a syntactically expected or strong thematically expected attachment. The expectation for an object is further increased if the object matches an object the system has encountered recently, that is, if the object has been "primed" Section 4.5.5).

To date, only relatively few systems build semantic interpretations incrementally on a word-by-word basis as they are computing syntactic structures, as our model does. One such system, inspired by Montague semantics (Dowty, Wall, & Peters, 1981; Montague, 1973) and described by Hirst (1987, 1988b), executes a semantic interpretation rule for each syntactic structure-building rule. The parser has access to a module ("The Semantic Enquiry Desk") that determines the semantically preferred syntactic structure-building operation. The module rates the proposed syntactic structures according to a fixed ranking scheme that combines lexical preferences for certain arguments as described in Ford et al. (1982) and semantic preferences as described in Crain and Steedman (1985). Pulman (1986) also presents a Montague-inspired method of semantic processing, but uses syntax only as a control structure for building semantic structures; that is, there are two sets of rules, but only the semantic rules produce representations. Huang and Guthrie (1985) describe a system, based on Semantic Definite Clause Grammars (Huang, 1985), that attempts to satisfy syntactic and semantic (selectional) constraint predicates in parallel.

In contrast to these synchronous systems, Cottrell's (1988) connectionist parser computes syntactic and semantic representations in parallel, using in-

dependent networks that may interact asynchronously. In one network, case roles, such as Agent, Object, and Beneficiary, are assigned to conceptual objects specified by the noun phrases of the sentence, and, in a separate network, syntactic roles, such as subject, predicate, and head, are assigned to syntactic objects, such as NP, VP, and Noun. Although Cottrell has not yet linked these two networks, he proposes to do so by means of binding nodes so that the pattern of activation in one network can influence, by means of support, the activation in the other network. Since the two processes are independent, one may run faster than the other, so it is possible that a syntactic interpretation might succeed on the basis of semantic strength alone or vice versa. In general, however, it is expected that one network would influence the other only when preferences within that network are too weak to resolve an ambiguity on their own.

4.5.5 Priming. In the proposed model, priming preactivates certain structures, making computations that access these structures run a little faster. For instance, at the lexical level, the disambiguation process always checks for preactivation of a possible word sense before seeing if other factors, such as selectional constraints, will help limit the number of choices. If one meaning has been sufficiently primed, all others will be eliminated right away. At the attachment level, similarity to an object the system has seen in the past is taken as evidence in support of the attachment (cf. Section 4.5.4).

4.5.6 Revision. Since structural ambiguities are to be resolved as soon as they arise, the model also specifies a mechanism to handle any mistakes recognized later. We assume that other than the most difficult errors, which *should* be irrecoverable, most errors are very common and cause the human parser very little difficulty because rules have been learned to correct them. Conventional backtracking approaches ignore what is known about the failure and waste the effort already invested in the current parse, making them psychologically implausible because of the time and memory constraints on human parsing. Instead, the proposed model revises structures by finding a revision rule that matches the current input and the current state of the parse and then attempting to apply that rule. This view is supported by Frazier and Rayner (1982) who found that when people discover that they have made an error in parsing, often they immediately regress back to the location of the ambiguity resolved by the current input. Such revision rules can be like the following schema for attaching a new node to an existing structure:

> If there is a phrase XP with daughter $X_i, \ldots, X_j$ along the right edge of the parse tree, but it was also possible to have attached $X_i, \ldots, X_j$ into the parse tree by supposing a phrase YP that allows both $X_i, \ldots, X_j$ and the new node, then replace XP by YP and attach $X_i, \ldots, X_j$ and the new node.

Although our model assumes such a revision mechanism, we have not concentrated on the details, and the implementation to be described performs only a few simple revisions of the style described above to attach postmodifying phrases.

To satisfy the RALR principle, the model associates a greater time cost with rebuilding an existing structure than simply adding to it, making revisions less favored. More difficult revisions will be more time consuming (and hence less favored) than simpler revisions. For example, a revision that entails simply adding a recursive level of structure between two existing nodes will be easier than a revision that requires moving a piece of existing structure, and both of these revisions will be much easier than revisions that require several operations such as adding structure, moving a piece of existing structure, and inserting a trace.[4]

5. AN IMPLEMENTATION OF RACE-BASED PARSING

The task of the sentence processor is to take English sentences, which may contain lexical or structural ambiguities, and process them a word at a time from left to right, incrementally building up unambiguous syntactic and semantic representations. From the discussion in the previous sections, it is evident that the correct representation will depend upon the interaction of a variety of factors, such as memory constraints, lexical, syntactic, and semantic expectations, structural preferences, and priming. We have thus proposed a two-stage processing model that gives a unified account of the interaction among expectations, preferences, and priming.

In our implementation of the model, we simulate the high degree of parallelism of human information processing in a predominantly serial system, encoding the principles using time-cost functions so as to "race" alternative interpretations. Independent processing modules called *hypothesizers* suggest alternative attachments and compute the corresponding time costs. Our implementation, which we now describe, demonstrates the workability of this approach.

Since our primary interest has been the race aspects of the model, we have limited our project to generating syntactic and semantic representations of

[4] In our discussion of MA, attachment classes were defined rather broadly to decrease the system's dependence on any subtleties of the grammar that have not been psychologically verified. Strictly speaking, incorporating RALR introduces some dependence on ad hoc features of the grammar because some grammars may construct a direct attachment where others may require a recursive level of structure. The solution is to make adding a recursive level of structure no more costly, that is, no slower, than a direct attachment, unless there is independent evidence to the contrary, such as a known difference in expectedness. Thus, the model's account of RALR captures the major generalization that movement and significant restructuring are possible, but that they are so costly that they will not be considered unless there is no simpler possibility.

simple sentences and do not take up problems concerning the identification of complex morphological structure, compounds, or binding. However, even this simplified system has required the integration of a variety of processing modules and knowledge sources (see Figure 2, p. 328).

The sentence processor is divided into two stages, running in parallel. Each stage processes its input one unit at a time (words or packages, respectively), combining the units as they are encountered. The first stage is limited to looking at five to seven words at a time, and, as it turns out, the second stage normally only looks at five to seven packages or less. As the first stage completes packages of words (including both a semantic and a syntactic representation) it passes them to the second stage. Each stage calls the attachment processor to incorporate the input units into the current syntactic and semantic structures. The attachment processor, in turn, relies on the hypothesizers to suggest attachments and on the grammar application routines to execute them. The hypothesizers themselves rely on consultant routines that provide information about grammatical phrase structures, the current contents of the knowledge base, and thematic arguments. The grammar application routines, which execute the attachments, also see to it that the semantic interpreter, Absity (discussed in Section 5.5), executes corresponding semantic operations. Not shown in Figure 2 are the lexical disambiguation processes, Polaroid Words (PWs; discussed in Section 5.4), which run in parallel with the system. These processes are triggered whenever a word is input or any associated PWs change (Hirst, 1987, 1988b; Hirst & Charniak, 1982); changes may also result from attachment decisions.

Whereas control flows downward through the system, time-cost information flows upward from the consultant routines to the attachment processor. The consultant routines supply the hypothesizers with time-cost factors necessary to compute the time cost of proposed attachments. The hypothesizers use the factors as arguments to their time-cost functions. These functions have been coded so as to preserve the time constraints correctly, thereby producing the desired principles. Thus, once a time cost is computed, it is returned, along with a description of the corresponding attachment, to the attachment processor. The attachment processor then performs the attachment with the lowest time cost.

We devote the remainder of Section 5 to describing the system's components and how they fit together. At the end of the section we present a simple parsing example.

5.1 Syntactic Processing

As mentioned above, the controlling processes for each stage simply cycle through their input, calling the attachment processor for each item. The first stage extracts each constituent from the sentence to be processed, produces syntactic and semantic structures, and passes them to the second stage as it completes them. The second stage thus receives these structures as input

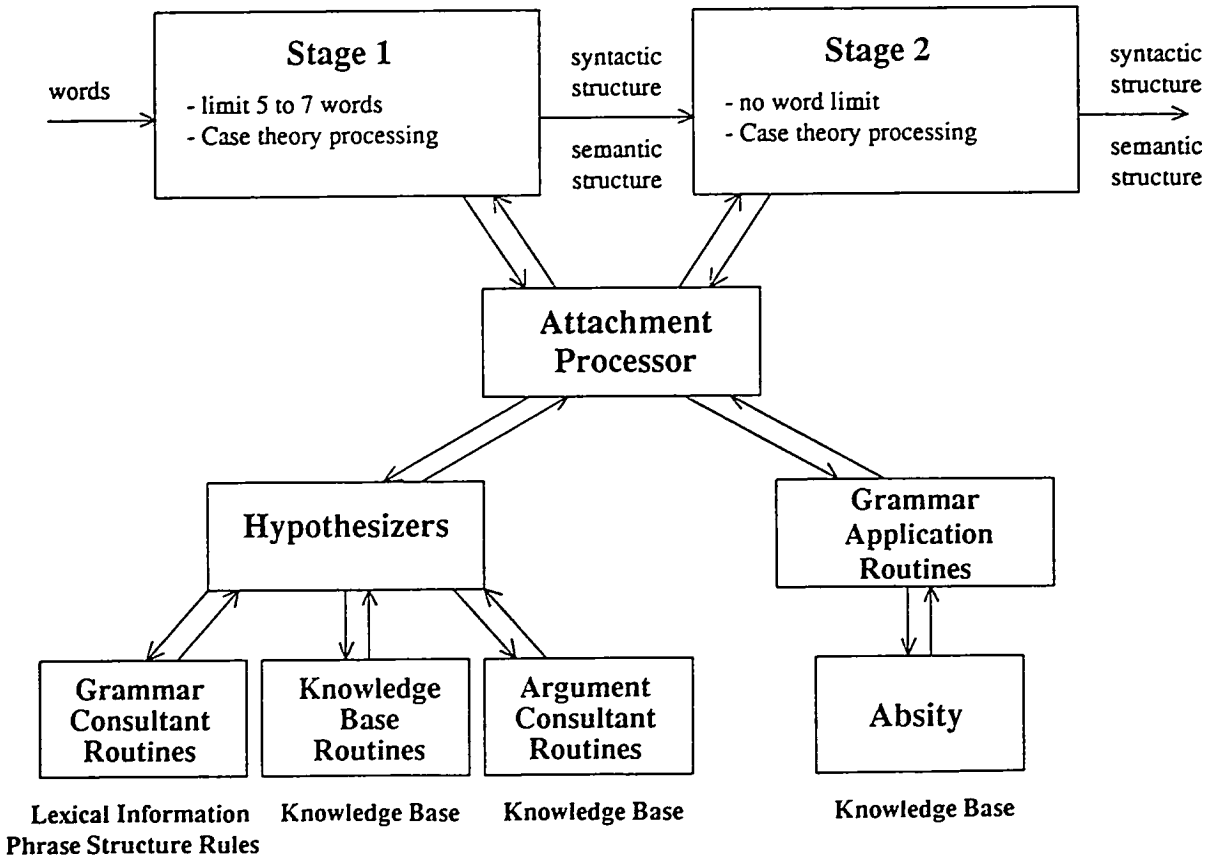

Figure 2. System overview.

and produces a syntactic and semantic representation for the entire sentence. The algorithms of each stage are shown below:

First Stage Loop:

1. Get the next word. Retrieve its lexical entry and start a PW process (see Section 5.4) for it.
2. Call the attachment processor to identify, evaluate, and execute the optimal attachment to incorporate the word into the current package.
3. Apply Case theory (see Section 5.2.1) to add any expectations.[5]
4. If the package is reaching capacity and this is a good break point (i.e., just before any punctuation mark, complement marker, preposition, conjunction, verb, or INFL that is *not* the last word of the sentence), or capacity has been met, close the package and pass it to the second stage.

Second Stage Loop:

1. Call the attachment processor to identify, evaluate, and execute the optimal attachment to incorporate the next package into the current structure.
2. Apply Case theory to add any expectations.

5.1.1 The Attachment Processor. Each of the parsing stages calls upon the attachment processor to combine structures. The attachment processor considers all possible attachments, *racing* them against each other using a form of simulated parallelism. In particular, after the attachments are identified by serially calling each of the applicable hypothesizers, the attachment processor selects the single attachment with the lowest time cost. Then, the processor executes the winning attachment, calling the specified grammar-application routine. The attachment algorithm is shown below:

Attachment Algorithm:

1. Determine the applicable hypothesizers on the basis of the syntactic category of the item to be attached to the current structure.
2. Activate the applicable hypothesizers. For each, get the possibilities it identifies and their associated time costs.
3. Find the one with the lowest time cost and apply it, executing the specified call to a grammar-application routine.

Since the attachment processor only looks for the attachment with the lowest time cost, multiple votes for the same attachment are not significant. Thus, it does not matter if different hypothesizers or underlying psycholin-

[5] These expectations will be useful for identifying a possible MA preference.

guistic theories account for the same underlying preference; the system is not vulnerable to being misled by a majority vote due to redundancy.

5.1.2 The Hypothesizers. The hypothesizers are a set of independent processing modules that identify possible attachments and generate corresponding calls to grammar-application routines that may be executed by the attachment processor. The hypothesizers also annotate each attachment specification with a time cost corresponding to the time necessary to identify and produce it. Each hypothesizer suggests as many attachments as it recognizes. This process is not unconstrained, because the two-stage model enforces local decision making, controlling the number of possible attachments to be considered, while still allowing for larger conjoined or complemented structures to be handled correctly. Hypothesizers are not intended to be psychologically plausible in themselves, but rather to be a mechanism for implementing the psychological constraints in a predominantly serial model.

Hypothesizers have access only to what there is to combine and to routines that encode or access attachment knowledge, which may be syntactic, contextual (i.e., from the knowledge base), or thematic. For example, a hypothesizer may ask an argument consultant routine how strongly, if at all, one item is expecting another item as an argument. Grammatical information may also be compiled into a hypothesizer itself. Note that it is unimportant whether individual hypothesizers encode general principles, specific subsets of general principles, or even special cases, permitting one to make the choices that are easiest to program or most efficient to run. Also, one does not have to decide open questions such as whether a given verb's argument expectations are thematic or syntactic; since the system is redundancy-tolerant, such expectations can be encoded both ways.

Our implementation has seven hypothesizers, some very general, and some quite specific, which we will now describe.

1. A hypothesizer that checks for structural expectations, via the grammar consultant routines, and suggests the corresponding attachments. An item is said to have *structural expectations* if it requires a particular syntactic structure to precede or follow it. In our grammar, an item may require a specifier to its left and some number of arguments to its right. Such requirements are listed in the lexicon (to be discussed in Section 5.2.1) and in partially completed syntactic structures, where they are found by the grammar consultants. Thus, the hypothesizer will suggest an attachment when an item expects the item following it as an argument, or when an item expects the item preceding it as a specifier. These will be minimal attachments, so their time costs will be very low.
2. A highly specific hypothesizer that recognizes the attachment possible when the new input is a verb and the current package contains only a

word that might be a plural noun, but is still ambiguous (so the system has no outstanding expectations). In this case, the first word is treated as the subject of a sentence with the next word as the beginning of a verb phrase. Such an attachment is possible because there is a chain of expectations linking a verb to a plural noun; assuming the noun interpretation of the first word, the plural noun morphology licenses a null determiner to form a determiner phrase (DP), the verbal morphology licenses INFL, and INFL selects a DP specifier and a verb-phrase argument (VP). This is less rapid than attachments of the previous hypothesizer.[6]

3. A hypothesizer that detects, through the grammar consultant routines and the knowledge base, when there can be an attachment of a postmodifier, and suggests these attachments. An item X allows item Y as a postmodifier if, first, there is a phrase structure rule linking Y to a projection of X as a postmodifier (e.g., $\overline{X} \rightarrow \overline{X}\ Y$), and secondly, X modified by Y describes an entity that is known either to exist or possibly exist (Hirst, 1987, 1988b). These attachments have higher time costs than expected attachments. They are also subject to RA and hence will require increasing time with increasing distance. However, if there is priming of the object described by the postmodified phrase, the time cost of these attachments will be reduced, because priming reduces the time needed to check existence.
4. A hypothesizer that detects, through the grammar consultant routines, when there can be an attachment of a phrase that begins with a premodifier and suggests these attachments. An item Y may be attached to item X if a projection of X expects an argument Z and there is a phrase structure rule linking Y to Z as a premodifier (e.g., $\overline{Z} \rightarrow Y\overline{Z}$). This is less rapid than either an expected attachment or an attachment of a postmodifier because it is not expected and requires building additional intermediate structure.
5. A hypothesizer that checks for thematic expectations, through the argument consultant routines, and suggests the corresponding attachments. Thematic expectations correspond to verb-frame preferences. Thus, these attachments may be more strongly favored than other attachments when there is no priming. The time costs of these attachments reflect the fact that more highly expected items are somehow easier to access, or more readily assumed to exist, than less expected items. For highly expected items, these thematically expected attachments will be equally as fast as structurally expected attachments.

---

[6] Note that, technically speaking, this hypothesizer is redundant; the system has a separate lexical disambiguation system, and Hypothesizer 7 will add the morphological elements necessary to join the two items. However, this seems to exaggerate the difficulty of this attachment. As with the case of revision, we claim that when an operation is not as difficult as it "should be," there must be some additional rule that has been learned. This hypothesizer represents such a rule.

6. A hypothesizer that detects when a verb can be attached to an apparently complete sentence, and suggests building the corresponding structure. This hypothesizer encodes a simple revision rule that allows one to reinterpret part of the current structure as a reduced relative clause modifying the subject of the sentence, and then make the new verb the main verb of the sentence. This is a nonminimal attachment involving some restructuring, and hence will have a relatively high time cost, unless there is some sort of priming of the object described by the newly modified subject. (This sort of priming has not yet been implemented.)
7. A hypothesizer that detects, through the grammar consultant routines, when an item can be linked to the existing structure by adding a missing morphological element whose expectations would link all three together, and suggests the corresponding operations. For example, morphological elements such as null determiners and INFL (the head of the clause) are *licensed* by affixes on nouns and verbs (plural marking and tense marking, respectively). This type of attachment requires recognition of the possible licensing, creation of a token to represent the subatomic element, and then attachment of both the token and the item that the system was trying to attach in the first place.[7] This hypothesizer thus requires a great deal of time, making it favored only when there are no other possibilities detected.

As mentioned earlier, the time-cost function of each hypothesizer preserves the time-cost constraints underlying the preference principles. These constraints establish an ordering among the suggestions made by an individual hypothesizer, but only a partial ordering over all suggestions. We expect them to provide only a partial ordering over all possible suggestions because of the lack of experimental data concerning the comparative costs of structures that do not compete. It is important to note, however, that we really only need to be concerned with getting the decision correct, that the attachment with the lowest time cost is, in fact, the one favored by the principles. Note that if two attachments receive the same winning time cost, then either they represent the same attachment (indicating a harmless redundancy in the system) or there has been a mistake made in the design of the underlying functions, or there is an unresolvable (genuine) ambiguity, for example, deliberate wordplay. However, since competitions involve a small number of attachments, time costs can still be limited to a small number of integer values.

Although it is sometimes convenient to think of time costs as measures that induce an ordering on hypothesizers, it is also somewhat misleading, as

[7] Note that it is *not* sufficient to assume that there is a front-end morphological analyzer that recognizes INFL in the verbal morphology and "undoes" the effects of affix-hopping, as in Abney and Cole (1985), because the existence of INFL is dependent upon the syntactic context; there must be a prior expectation for INFL.

h1 < h6
h1 < h2 < h7
h1 ≤ h5 < h3 < h4 (modified object unprimed)
h1 ≤ h3 < h5 < h4 (modified object primed)

Figure 3. Ranking of values of time-cost functions, by hypothesizer.

some hypothesizers produce multiple suggestions with different time costs, including some that depend on dynamic factors. Time costs are numbers that indicate the difficulty of identifying and constructing an individual attachment using a particular strategy, relative to competing attachments. By using a ranking instead of an absolute measure, we can proceed even though knowledge of the human grammar and parsing device is still incomplete, although it also means that we cannot add or otherwise combine time costs to compare entire sentences. With this caveat, we summarize in Figure 3 the relationships among the hypothesizers with respect to values of their time-cost functions. Again, since not all hypothesizers compete against each other, a complete ranking among hypothesizers is neither meaningful nor possible. For example, Hypothesizers 6 and 7 are never simultaneously active. We also assume that Hypothesizers 6 and 7 do not compete with Hypothesizers 3, 4, or 5, and that Hypothesizer 2 does not compete with Hypothesizers 4, 5, or 6. Since Hypothesizer 2 is rarely active, it does not normally compete with Hypothesizer 3, but when it does, h2<h3, except when the postmodifier attachment has been primed.

5.2 The Grammar and Lexicon

The system relies on a set of grammar routines to identify and construct possible attachments. The grammar consultant routines report to the hypothesizers whether any attachment is syntactically possible, whereas the grammar application routines are called by the attachment processor actually to form an attachment. (As a side effect, the grammar application routines will also call on the semantic interpreter to perform corresponding semantic operations.)

As mentioned in Section 4.3, the grammar used by the system is derived from GB. Thus, the parser builds tree structures according to $\overline{X}$ theory (parameterized for English), which predicts that all phrases will have the structure shown in Figure 4 (p. 334, including some optional levels), with parameterization simply fixing the order of the branches at each level. Both $\overline{X}$ and $\overline{\overline{X}}$ (i.e., XP), are considered projections of the head X. The system considers nouns, adjectives, adverbs, prepositions, determiners, and INFL to all be heads. It also assumes the maximal bar level for each head to be two, that is, there are only two types of nonterminal nodes, $\overline{X}$ and XP.

A word may be restricted to occurring only in certain contexts, requiring (or prohibiting) a specifier or some arguments, and may require these constituents to have certain features if they are present. These restrictions are contained in the word's entry in the lexicon.

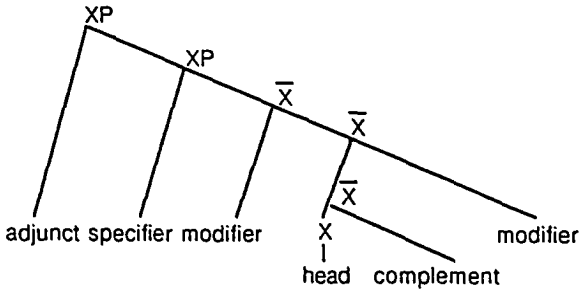

Figure 4. $\overline{X}$ structure.

Finding a good theoretical account of what can modify what, from what direction, is still an open question in linguistics, and thus the parser also relies on a small set of context-free phrase structure rules to describe this information. When a possible modifier (such as an adjective, PP, relative clause, or adverb) or a possible first word of a modifier needs to be attached, a hypothesizer can call a grammar consultant routine that will attempt to find a sequence of rules that links the incoming item to current structure or its expectations.

The system also checks for elements introduced by morphology (e.g., plural noun phrases license null determiners) and expectations created by Case theory (e.g., when it finds a determiner phrase, it sets up an expectation for a Case assigner). The hypothesizers may call the routine that checks for morphological elements. This routine also provides them with a time-cost factor indicating the relative complexity of the object licensed (e.g., a null determiner or INFL is considered less complex than a null complementizer/INFL combination).

5.2.1 The Lexicon. Following Brunson (1988), the lexicon is divided into two parts, the word lexicon and the category lexicon. The category lexicon associates each syntactic category with its default selectional information as well as its semantic type. The word lexicon associates each word with its "meanings," the corresponding syntactic categories, and any exceptional selectional information. Both parts of the lexicon are implemented in MRS, a Prolog-like knowledge representation language (Russell, 1985).

The lexicon is very important to the semantic interpretation process, because semantic representations are constructed using word meanings and semantic types (Section 5.5). In addition, since syntactic structures are constructed by projecting phrasal heads and attaching the projections together on the basis of selectional information stored in the lexicon, the lexicon also plays a central role in syntactic analysis. Thus, the lexicon includes information that subsumes much of Case theory and θ-theory.

```
determiner       (DP det NP)
complementizer   (nil comp IP)
preposition      (nil prep DP)
INFL             (DP infl VP)
```

Figure 5. Default phrase grids.

Case Theory. In GB, Case theory provides constraints on the distribution of lexical determiner phrases (non-null DPs) by requiring that in the surface syntactic structure all lexical DPs must have "Case," an abstract property of constituents. Lexical DPs get Case by being adjacent to items such as verbs, prepositions, or INFL.[8] Thus, when the parser's Case theory routine recognizes a DP, it immediately looks for a corresponding Case assigner, and if it fails to find one, it creates an expectation for one (Brunson, 1988). Information about which elements assign Case and what they can assign it to, is included in the word and category lexicons.

The categorial lexical entries for determiner, complementizer, preposition, and INFL include a default phrase grid of the form:

(*specifier category complement*)

as shown in Figure 5. The elements of these grids are taken to be obligatorily adjacent in the sentence, and must be in the order given. A particular element may be explicitly marked as optional, as described below.

The word lexicon includes the syntactic category of the word, syntactic features such as number and Case, modifications to the default category entries—for example, the fact that "the" does not select a specifier—as well as semantic information. The entry for the noun "bank" is shown in Figure 6 (p. 336). Nouns, verbs, and adjectives all have selectional properties (including a phrase grid similar to general category entries, labeled with the token "cases:" as shown in Figure 7, p. 336) stored with the individual words' entries. To account for the fact that some verbs have arguments that are optional or that can be one of a set of syntactic types, there are also mechanisms for indicating optionality and "enumerated" types: Optional arguments are listed in parentheses and enumerated types are listed in a parenthesized list beginning with the token "OR." For example, optionality is illustrated in the lexical entry for the verb "read," which selects two optional DP complements, in Figure 7. (We will explain the meaning of "th-grids" and "th-roles" in the next section; "rating" is intended to be a measure of the relative frequency of usage among the different meanings, but is not used in the current implementation. "HANIM" corresponds to the

[8] The Case assigners in GB are the nonlexical category INFL and lexical categories that have the feature [−N].

```
(bank noun  ((number singular))
            (meaning: BANK-BUILDING
             rating:  20
             isa:     BUILDING)
            (meaning: STEEP-SLOPE
             rating:  10
             isa:     NATURAL-LOCATION)
            (meaning: FINANCIAL-INSTITUTION
             rating:  20
             isa:     ORGANIZATION)
            (meaning: RIVER-EDGE
             rating:  20
             isa:     NATURAL-LOCATION))
```

Figure 6. Sample lexical entry for a noun.

```
(read verb ((form infinitive en) (tense <past>) (case) (case))
           (meaning: READ
            isa: PRESENTATION
            rating: 20
            cases: (nil read (DP)(DP))
            th-grids: ((Agent Theme) (Agent Beneficiary Theme))
            th-roles:
                      ((role: Agent
                        sr-reqs: HANIM)
                       (role: Beneficiary
                        sr-reqs: HANIM)
                       (role: Theme
                        sr-reqs: WRITTEN-THING))))
```

Figure 7. Sample lexical entry for a verb.

knowledge-base (KB) category for people, or entities construed as people, that is, "higher animate beings.")

Case theory constraints are satisfied because the grammar routines enforce the obligatory adjacency and ordering of elements satisfying the phrase grids and because a Case theory routine (called directly by each processing stage) checks for DPs without Case. (Case assigners will have the necessary "Case" features in their lexical entry.)

θ-Theory. θ-theory provides semantic information about the roles that various arguments bear to the word they complement (see Chomsky 1981; Sells, 1985; and references therein). In particular, it is intended that there be

a one-to-one mapping between arguments and θ-roles (Chomsky, 1981), where arguments of a head may be internal or external to the corresponding phrase; for example, the verb "buy" requires two θ-roles: {AGENT} {THEME}.[9] Associated with each verb in the lexicon there is an information template (called the *argument structure* or *θ-grid*), listing the thematic roles of the verb, perhaps along with some information about the syntactic structures that may bear those roles. Similarly, NPs inside PPs receive their θ-role via the preposition.

The θ-role information has been encoded into the word lexicon of the sentence-processing system. For a preposition, the θ-role of its NP argument will be identified with the set of possible "meanings" of the preposition itself. For verbs, we include a list of θ-grids, labeled with the token "th-grids:", which list the verb's θ-roles. A verb's entry also includes a list of selectional restrictions, describing allowable arguments for each θ-role, labeled with the token "th-roles:" (Figure 7). The θ-role information we encode is based on Fillmore's (1968) notion of case, subsequently used in GB. (For similar approaches, see Hirst & Charniak, 1982; Wilensky & Arens, 1980.)

5.3 Arguments

In addition to their obligatory arguments, some words (most notably verbs) allow optional arguments or modifiers. Roughly, optional arguments change the meaning of the verb in some way, whereas modifiers are more circumstantial. For example, many verbs allow a BENEFICIARY, commonly considered an optional argument. "Read" is one such verb, as shown by (3), in which "Evan" is the AGENT, "the book" is the THEME, and "Mara" is the BENEFICIARY.

(3) Evan read the book to Mara.

Similarly, most verbs permit a TEMPORAL, such as "today" in Sentence (4), usually considered a modifier.

(4) Murray sent more electronic mail today.

However, deciding whether something is an optional argument or a modifier is notoriously problematic (Somers, 1987; Vater, 1978).

A related difficulty is deciding which of a verb's arguments and θ-roles should be listed in its lexical entry. Obligatory arguments, and arguments to

[9] θ-roles, although often identified with a single thematic relation as in the above example, are actually nonempty *sets* of thematic relations (Jackendoff, 1972) as demonstrated by the following sentence:

(i) Tom gave the book to Diane.

where the subject NP "Tom" is both the AGENT and the SOURCE.

(a) Path	*e.g.*, into the sea
(b) Environment	*e.g.*, on land
(c) Means of conveyance	*e.g.*, by car
(d) Modes of action in locomotion	*e.g.*, momentary
(e) Velocity	*e.g.*, quickly
(f) Manner of locomotion	*e.g.*, sneakily

Figure 8. Optional arguments for verbs of movement.

which the verb assigns Case are included because they are highly expected and small in number. But there are too many possible optional arguments and modifiers with too many possible orderings to make explicitly including them a feasible enterprise. Instead, our system exploits the apparent fact that verbs can be grouped into classes according to the optional arguments they permit, and so includes a mechanism for introducing optional arguments to verb phrases when encountered. For example, verbs of movement all expect the optional arguments shown in Figure 8, except for arguments implicit in the verb itself (e.g., "swim" lexicalizes the Environment "in water"; Fink, 1978). Different arguments and modifiers often have different levels of preference (i.e., expectedness), and the system accounts for this also.

In the model, the time-cost distinction between arguments and modifiers is presumed to result from the former being identified in the lexicon, a relatively quick operation, whereas the latter are identified by the KB, a relatively slow operation. However, in the implementation, our choice of what to put in the lexicon has been made on pragmatic, rather than theoretic, grounds. Because the system cannot distinguish arguments and modifiers on the basis of where knowledge of them is actually stored, and because it also needs to represent variations in preference strengths, information about thematic arguments and modifiers has been consolidated in an argument consultant routine that encodes the time-cost distinctions explicitly. A hypothesizer may ask the argument consultant routine whether a verb allows a certain argument and the routine will respond with a time-cost factor given by a static rating scale, ranging between 0 and 1. This value can then be used by the hypothesizer to compute the time-cost function for the corresponding attachment. Although the values of the time-cost factors must be determined experimentally, the underlying ordering is determined by theoretical criteria, such as whether a particular item is an argument or a modifier of a particular verb-type and the relative strength of the preference a verb-type has for an item.

5.4 Lexical Disambiguation

The system resolves lexical ambiguities in parallel with syntactic processing. Sentence processing is complicated by the fact that many of the words we

use have multiple meanings. Lexical disambiguation cannot occur prior to syntactic processing because words that are categorially ambiguous (e.g., "sink" can be a noun or a verb) require syntactic context to resolve the ambiguity. Our sentence processor incorporates Hirst's approach to lexical ambiguity resolution called *Polaroid Words* (Hirst, 1987, 1988a, 1988b; Hirst & Charniak, 1982).

As each word is read, a process responsible for the disambiguation of that word is created. The process starts with all possible senses (i.e., meanings) of the word, and uses information such as activation by semantic context (as by marker passing from related words in a semantic network, but not used in our current implementation) and selectional constraints among neighboring words to try to resolve any ambiguity. Thereafter, the process will be reactivated whenever a new word is encountered or any associated PWs change. Syntactic processing interacts with lexical disambiguation by causing a PW to eliminate all senses corresponding to syntactic categories incompatible with the current syntactic structure, thus triggering reactivation.[10]

5.5 Semantic Processing

Also in parallel with syntactic processing, the system computes semantic representations for the constituents that have been encountered. The semantic interpretation scheme used by the system is an adaptation of the Absity system (Hirst, 1983, 1987, 1988b) for use with GB (see Figure 9, p. 340).[11] For every syntactic attachment operation there is a corresponding semantic operation, and when an attachment is performed by a grammar application routine, the corresponding syntactic and semantic operations are executed in tandem; the grammar routine calls on Absity to handle the semantic operations. The representations produced by Absity are immediate descriptions of semantic objects (i.e., frames, or frame-related objects) in the knowledge base.

5.6 The Knowledge Base

The system's knowledge base (KB) encodes information about objects the system knows about, the way objects are related to each other in the real world, and instances of objects the system has encountered during sentence processing. The KB classifies objects in a hierarchy (e.g., a bank is a building and a building is a location constructed by humans). The static information in the KB is used by PWs, Absity, and the argument consultant (e.g., for checking selectional constraints or determining an object's classification).

---

[10] The PW model was also used in the earlier cited work to disambiguate θ-roles by means of "pseudopreposition" tokens inserted into the sentence. In order to remain compatible with GB, however, the present system disambiguates θ-grids within the process that is responsible for the verb.

[11] Symbols beginning with "$" are variables.

Syntactic Type	Semantic Type	Example Semantic Representation
Determiner	Frame determiner	`(the $x)`
Noun	Frame descriptor	`(apple $x)`
Adjective	Slot-filler pair	`(color $x red)`
Preposition	Slot name	`location`
Verb	Frame descriptor	`(put $x)`
INFL	F-infl	`<PAST>`
Noun Phrase	Frame descriptor	`(apple $x (color $x red))`
Determiner Phrase	Frame statement	`(the $x (apple $x (color $x red)))`
Prepositional Phrase	Slot-filler pair	`(location $x (the $y (table $y)))`
Verb Phrase	Frame descriptor	`(put $x (theme $x (the $y (apple $y)))` `(location $x (the $z (table $z)))`
INFL Phrase	Frame-statement/ frame-determiner	`(<PAST> (read $x` `(agent $x John) ...))`

Figure 9. Some example semantic types and representations.

H(a.1) = 1
H(a.2) = 2
H(a.3) = round(1.5 * E * T)
H(a.4) = 3 * T
H(a.5) = round(A * T)
H(a.6) = 11
H(a.7) = 1 + G

A = 0.25, for a very strong verb-frame preference
A = 0.30, for a strong verb-frame preference
A = 1.00, for a weak verb-frame preference

E = 0.50, if the item is primed
E = 1.25, if the item might possibly exist

G = 2, to build a null determiner or INFL
G = 5, to build a reduced relative clause

T = the number of nodes that were traversed

Figure 10. Time-cost functions.

Instances of objects are created dynamically; as sentences are processed, the objects and events they describe are added to the KB. The KB also provides consultant routines that are used by the hypothesizers for checking for the existence (or possibility) of described instances.

The underlying knowledge representation language for the KB is MRS, (a Prolog-like, knowledge representation language with meta-level reasoning capabilities; Russell, 1985). Additionally, the system uses a set of frame representation constructs implemented on top of MRS to approximate the syntax of the PSN (Procedural Semantics Network) language (Becker & Greiner, 1987; Levesque & Mylopoulos, 1979).

5.7 An Example Sentence

A simple example will show how the components of the system fit together and how the timing relationships come into play. (For simplicity, we will concentrate only on the syntactic processing.) We will assume the time-cost functions shown in Figure 10 where H(a,i) is the time-cost function returned by Hypothesizer i for Attachment a, E is the time-cost factor returned by the KB routines, A is the time-cost factor returned by the argument processor, G is a time-cost factor returned by a grammar consultant routine, and T is a number computed by the hypothesizer itself indicating the number of nodes traversed. These functions satisfy most of the constraints

```
(unset (bar 2)
  (det (bar 2) dan)
  (Expects:  (OR: ((det (bar 1) ((case))) (infl (bar 1) ((case)))))))
```

Figure 11. Expectation for a Case assigner.

discussed in Section 5.1.2; however, they do not account for structural priming of a reduced relative clause (Hypothesizer 6), and they do not assess any cost for using infrequently used word senses.

Suppose the system is processing Sentence (5):

(5) Dan read the letter to Chrysanne.

When processing begins, the buffer is empty. When "Dan" is processed, the system retrieves the word from the lexicon and creates a structure to represent it. Finding no ambiguity, the system immediately creates the corresponding syntactic representation, (det (bar 2) dan), and attempts to attach the structure into the current package. Since no attachments are possible, the structure is just left in the input buffer as the start of a new package.

At the next step of the algorithm, the system checks for elements predicted by Case theory. It finds that the item in the buffer needs Case and sets up an expectation for a Case assigner, as in Figure 11 (here we use a list representation for tree structures in which the token "unset" is a placeholder for any unknown value). Specifically, the system builds a maximally projected node of unknown category that has the DP node as its child and an expectation for either a $\overline{D}$ or an $\overline{I}$ with Case.

Next, "read" is processed. According to the lexicon, "read" is a verb expecting (but not requiring) two DP arguments; it has two possible θ-grids, as shown in Figure 7 (p. 336). To attach "read" to the current package, the attachment processor is called and it calls the applicable hypothesizers. Hypothesizers 1, 2, 6, and 7 are activated and they identify two possibilities: Hypothesizer 7 reports that given a bit of missing morphology, "read" could be attached as the main verb (to satisfy the Case-induced expectation of the noun phrase) and it also reports that it could be attached as the start of a reduced relative clause. The latter loses the race because it is a much more complex attachment (and has a time cost of 6 compared to the former's 3), so the attachment processor executes the specification for adding the missing morphological elements (in this case an INFL) and attaching "read" as the main verb.

First, a grammar routine creates a node representing an INFL and adds it to the buffer. This node (Figure 12) contains expectations established by the category lexicon; it expects a DP to the left and a VP to the right. To attach this new node, Hypothesizers 1, 3, and 5 are called, but only one possibility is detected. Hypothesizer 1 suggests using the INFL to satisfy the existing syntactic expectation.

```
(infl (bar 0) agr
  (Specifier: (det (bar 2)))
  (Expects:  (verb (bar 2))))
```

Figure 12. An INFL node.

```
(infl (bar 2)
  (det (bar 2) dan)
  (infl (bar 1)
    (infl (bar 0) agr)
    (verb (bar 2)
      (verb (bar 1)
        (verb (bar 1)
          (verb (bar 0) read)
          (det (bar 2)
            (det (bar 1)
              (det (bar 0) the)
              (noun (bar 2)
                (noun (bar 1)
                  (noun (bar 0) letter)))))))
```

Figure 13. Syntactic representation after "letter" is added.

Then, the routine proceeds with the attachment of the verb. Hypothesizers 1, 2, 6, and 7 are called again, but now only one possibility is reported: Use the verb to fulfill the syntactic expectation of the INFL. The parser executes the attachment, completing the incorporation of "read."

Returning to the top-level algorithm, no new elements are predicted by Case theory, so the system begins processing the next word, "the," which is found to be a determiner expecting an NP. Hypothesizer 1 is called and suggests using the determiner to fulfill the expectation of "read" for a DP. The parser does so.

Again, no new elements are predicted by Case theory, so the system begins processing "letter," a noun without any expectations. Hypothesizers 1 and 7 are called, but only one suggestion is reported: Use the noun to fulfill the expectation of "the." The result is an $\overline{\overline{\text{INFL}}}$ (IP) with no outstanding expectations, as in Figure 13.

The parser is now ready to process the word "to," and the race process becomes more interesting. When "to" is input, the first stage contains a representation for "Dan read the letter," an IP with no outstanding expectations. (Since "to" cannot be a determiner, the verb's remaining syntactic

expectation was deleted.) "To" is categorially ambiguous, being possibly a preposition or an INFL. It is also semantically ambiguous; according to the system's lexicon it denotes either DESTINATION, BENEFICIARY, or PURPOSE. There are three phrase structure rules that match the input namely $\overline{N} \rightarrow \overline{N}$ PP, $\overline{V} \rightarrow \overline{V}$ PP, and $\overline{V} \rightarrow \overline{V}$ IP. Recall from Figure 7 that "read" isa PRESENTATION; hence by transitivity in the hierarchy of the knowledge base it is a TRANSFER and an ACTION. Thus, the argument processor has two rules that apply: a TRANSFER frame is closely related to a slot that is a THEME, BENEFICIARY, or SOURCE (time-cost factor 0.25); and an ACTION frame is related to a slot that is a PURPOSE (time-cost factor 0.3). Thus, there are three remaining possible structural interpretations, roughly construed as follows:

(a) There is a letter to someone and Dan read it.
(b) Dan read to someone, and what he read was the letter.
(c) Dan read a letter to serve some purpose.

We will assume there is no priming in the KB; that is, we assume that none of the following semantic structures, each of which would have primed one of these interpretations, is present:

(letter $x (beneficiary $x $y)), 'a letter to some y'
(read $x (beneficiary $x $y)), 'read to some y'
(read $x (purpose $x $y)), 'read to do some y'

Neither of the possibilities

(letter $x (destination $x $y)), 'a letter to some place y'
(read $x (destination $x $y)), 'read to some place y'

would be meaningful to the KB; DESTINATION denotes a final location, not to be confused with the BENEFICIARY of a letter).

Hypothesizers 1, 3, 4, and 5 are called and they report the following five possibilities:

1. Hypothesizer 3 reports that "to" can be the start of a PP modifier of the noun "letter" using the rule $\overline{N} \rightarrow \overline{N}$ PP, as in (a), with a time cost of 4 (which would have been 2 if there had been priming).
2. Hypothesizer 3 also reports that "to" can be the start of a PP modifier of the verb "read" using the rule $\overline{V} \rightarrow \overline{V}$ PP, as in (b), with a time cost of 11 (or 2 if there had been priming). The unprimed time estimate is longer than it is in the first case because "read" is farther away than "letter."
3. Hypothesizer 3 further reports that "to" can be the start of a verb-modifying clause like "to pass the time" using the rule $\overline{V} \rightarrow \overline{V}$ IP, as in (c), with a time cost of 11 (or 2 if this structure had been primed).
4. Hypothesizer 5 reports that "to" can be the start of a PP that fulfills the (somewhat) expected BENEFICIARY role of the verb "read," as in (b), with a time cost of 3.

```
(infl (bar 2)
  (det (bar 2) dan)
  (infl (bar 1)
    (infl (bar 0) agr)
    (verb (bar 2)
      (verb (bar 1)
        (verb (bar 1)
          (verb (bar 0) read)
          (det (bar 2)
            (det (bar 1)
              (det (bar 0) the)
              (noun (bar 2)
                (noun (bar 1)
                  (noun (bar 0) letter))))))
        (prep (bar 2)
          (prep (bar 1)
            (prep (bar 0) to))
          (Expects: (det (bar 2) at prep (bar 1)))))))))
```

Figure 14. Syntactic representation after "to" is added.

5. Hypothesizer 5 also reports that "to" can be the start of a clause that fulfills the (less expected) PURPOSE role of the verb "read," as in (c), with a time cost of 5.

The winner is the fourth suggestion, that is, to start a PP attached to the verb that fulfills its BENEFICIARY role, so this attachment is executed. The result is the structure shown in Figure 14, an IP containing a PP that expects a DP.

The parser proceeds, again finding no elements predicted by Case. It then simply attaches "Chrysanne" (an unambiguous DP) to fulfill the syntactic expectation of "to" as suggested by Hypothesizer 1, the only one called.

6. CONCLUSIONS

6.1 Some Needed Work

Although it improves upon earlier models, the model as proposed and implemented has several deficiencies of its own. For example, more work must be done on designing additional hypothesizers for proposing more complex structures and revisions, and a mechanism for checking semantic well-formedness and triggering any necessary reanalysis must be added. The system would also benefit from incorporating semantic expectations, triggered by referential failure (Altmann & Steedman, 1988). The semantic interpretation

process could also use some refinement (see Hirst, 1987, for a discussion of what Absity does not do).

Perhaps the largest lacuna in this work is the lack of empirical data for the relative timing of the hypothesizers. When we began the work, we expected that details of the relative rapidity of many of the subprocesses of comprehension would be readily available in the literature. In fact, they are not. We were therefore obliged to fall back on the intuitive principles that we have described: that lexical hypotheses can be tested more rapidly than syntactic hypotheses, for example, or that the construction of a parse tree takes a time roughly linear in the number of nodes created. Our model relies crucially on the assumption that such data, when available, will show both intra- and interindividual consistency (in terms of relative values); that is, a choice between the same alternatives in the same sentence and discourse context must not have different results at different times or in different people.

Until such data is available, our model may be thought to be vulnerable to the criticism that it is as circular as Frazier and Fodor's (1978). Just as their principle of MA relied on ad hoc aspects of their grammar, and their grammar seemed set up just to make MA work, so, too, our model and the timings used by the hypothesizers seem to have only each other as justification. The difference, however, is that our model is falsifiable—it makes clear empirical predictions about the time course of parsing—whereas Frazier and Fodor's suggestions about the precise structure of parse trees do not seem amenable to psycholinguistic observation. Moreover, our approach is able to account for when and why structural preferences (which may be seen as defaults) may be overridden.

6.2 Summary

The goal of this work has been to propose a psychologically plausible model of parsing that would be interesting to both linguists and computer scientists. We identified several important psychological claims about the human sentence-parsing mechanism, namely, that processing is influenced by limitations on working memory and by a number of structural preferences, such as RA, MA, RALR, and verb-frame preferences. Although there exist sentence-processing systems that give reasonable accounts of subsets of the psycholinguistic data, we found that none provide a good account of how the remaining data could be integrated consistently. In contrast, we proposed a processing model that gives a reasonable account of how one might bring all these necessary constraints and preferences together in one system. The starting point for the proposal was the Sausage Machine model (Fodor & Frazier, 1980; Frazier & Fodor, 1978), which provides a good account of memory constraints and sentence complexity, and incorporates most of the structural preferences we were seeking to include. From there, we attempted to overcome the more serious deficiencies of the Sausage Machine, namely its dependence on unprincipled aspects of its grammar and its omission of

verb-frame preferences. We thus proposed the following modifications, and described how they might be made:

- Employ a principled theory of grammar.
- Use estimated timing information to resolve conflicting preferences, that is, activate a set of routines (called "hypothesizers") to propose attachments, and execute the attachment that wins a "race."
- Add mechanisms to handle lexical disambiguation and semantic processing in parallel with syntactic processing.

The result is a sentence processor that integrates several accounts of human preferences for resolving ambiguity, without relying upon ad hoc features of its grammar.

REFERENCES

Abney, S.P. (1987). *The English noun phrase in its sentential aspect.* Unpublished doctoral dissertation, Massachusetts Institute of Technology, Cambridge.

Abney, S., & Cole, J. (1985). A government-binding parser. *Proceedings of the Fifteenth Annual Meeting of the North-Eastern Linguistic Society.*

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

Becker, S., & Greiner, R. (1987). *On the implementation of a frame language in MRS.* Unpublished manuscript, University of Toronto, Department of Computer Science.

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

Brunson, B.A. (1988). *A processing model for Walpiri syntax and implications for linguistic theory* (Tech. Rep. No. 208). Toronto, Canada: University of Toronto, Computer Systems Research Institute. (Based on Forum Paper, University of Toronto, Department of Linguistics, August 1986)

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

Church, K.W. (1980). *On memory limitations in natural language processing.* Master's thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge. (Published as Tech. Rep. No. MIT/LCS/TR-245)

Clifton, C., Frazier, L., & Connine, C. (1984). Lexical expectations in sentence comprehension. *Journal of Verbal Learning and Verbal Behavior, 23,* 696–708.

Connine, C., Ferreira, F., Jones, C., Clifton, C., Jr., & Frazier, L. (1984). Verb-frame preferences: Descriptive norms. *Journal of Psycholinguistic Research, 13,* 307–319.

Cottrell, G.W. (1988). *A connectionist approach to word sense disambiguation.* San Mateo, CA: Morgan Kaufmann.

Cowper, E.A. (1987). *An introduction to syntactic theory: The government-binding approach.* Unpublished manuscript, University of Toronto, Department of Linguistics.

Crain, S., & Steedman, M. (1985). On not being led up the garden path: The use of context by the psychological syntax processor. In D.R. Dowty, L. Karttunen, & A.M. Zwicky (Eds.), *Natural language parsing: Psychological, computational, and theoretical perspectives* (pp. 320–358). New York: Cambridge University Press.

Dorr, B. (1987, July). A principle-based approach to parsing for machine translation. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 78–88). Seattle, Washington.

Dowty, D., Wall, R.E., & Peters, S. (Eds.). (1981). *Introduction to Montague semantics.* Dordrecht, The Netherlands: D. Reidel.

Ferreira, F. (1986). The role of context in resolving syntactic ambiguity. *University of Massachusetts Occasional Papers in Linguistics, 9,* 40–62.

Ferreira, F., & Clifton, C., Jr. (1986). The independence of syntactic processing. *Journal of Memory and Language, 25,* 348–368.

Fillmore, C.J. (1968). The case for case. In E.W. Bach & R.T. Harms, (Eds.), *Universals in linguistic theory* (pp. 1–88). New York: Holt, Rinehart & Winston.

Fink, S. (1978). Case grammar and valence theory at a stalemate? Their relevance for semantic memory. In W. Abraham (Ed.), *Valence, semantic case and grammatical relations* (pp. 177–190). Amsterdam: John Benjamins.

Fodor, J.D. (1978). Parsing strategies and constraints on transformations. *Linguistic Inquiry, 9,* 427–473.

Fodor, J.D., & Frazier, L. (1980). Is the human sentence-parsing mechanism an ATN? *Cognition, 8,* 417–459.

Ford, M., Bresnan, J., & Kaplan, R. (1982). A competence-based theory of syntactic closure. In J. Bresnan (Ed.), *The mental representation of grammatical relations* (pp. 727–796). Cambridge, MA: MIT Press.

Frazier, L. (1978). *On comprehending sentences: Syntactic parsing strategies.* Bloomington, IN: Indiana University Linguistics Club.

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

Frazier, L., & Rayner, K. (1982). Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology, 14,* 178–210.

Frazier, L., Taft, L., Roeper, T., Clifton, C., Jr., & Ehrlich, K. (1984). Parallel structure: A source of facilitation in sentence comprehension. *Memory and Cognition, 12,* 421–430.

Hirst, G. (1983, June). A foundation for semantic interpretation. *The 21st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference* (pp. 64–73). Cambridge, MA.

Hirst, G. (1987). *Semantic interpretation and the resolution of ambiguity.* Cambridge: Cambridge University Press.

Hirst, G. (1988a). Resolving lexical ambiguity computationally with spreading activation and Polaroid Words. In S. Small, G. Cottrell, & M. Tanenhaus (Eds.), *Lexical ambiguity resolution* (pp, 73–107). San Mateo, CA: Morgan Kaufmann.

Hirst, G. (1988b). Semantic interpretation and ambiguity. *Artificial Intelligence, 34,* 131–177.

Hirst, G., & Charniak, E. (1982, August). Word sense and case slot disambiguation. *Proceedings, National Conference on Artificial Intelligence (AAAI–82)* (pp. 95–98). Pittsburgh.

Holmes, V.M. (1984). Parsing strategies and discourse context. *Journal of Psycholinguistic Research, 13,* 237–257.

Huang, X. (1985, August). Machine translation in the SDCG formalism. *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages* (pp. 135–144). Colgate University, New York.

Huang, X., & Guthrie, L. (1985). *Parsing in parallel* (Tech. Rep. No. MSSC–85–40). Las Cruces: New Mexico State University, Computing Research Laboratory.

Jackendoff, R. (1972). *Semantic interpretation in generative grammar.* Cambridge, MA: MIT Press.

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

Klatsky, R.L. (1980). *Human memory: Structures and processes* (2nd ed.). New York: W.H. Freeman.

Levesque, H., & Mylopoulos, J. (1979). A procedural semantics for semantic networks. In N.V. Findler (Ed.), *Associative networks: Representation and use of knowledge by computers* (pp. 93–120). New York: Academic Press.

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

Marslen-Wilson, W. (1975). Sentence perception as an interactive parallel process. *Science, 189,* 226–228.

Marslen-Wilson, W., & Tyler, L.K. (1980). The temporal structure of spoken language understanding. *Cognition, 8,* 1–71.

Marslen-Wilson, W.D., Tyler, L.K., & Seidenberg, M. (1978). Sentence processing and the clause boundary. In W.J.M. Levelt & G.B. Flores d'Arcais (Eds.), *Studies in the perception of language* (pp. 219–246). Chichester, England: Wiley & Sons.

McRoy, S.W. (1987). *The influence of time and memory constraints on the resolution of structural ambiguity.* Master's thesis, University of Toronto, Department of Computer Science. (Published as Tech. Rep. No. CSRI-209, CSRI, University of Toronto)

Mitchell, D.C., & Holmes, V.M. (1985). The role of specific information about the verb in parsing sentences with local structural ambiguity. *Journal of Memory and Language, 24,* 542–559.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In K.J.J. Hintikka, J.M.E. Moravcsik, & P.C. Suppes (Eds.), *Approaches to natural language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics* (pp. 221–242). Dordrecht, The Netherlands: D. Reidel. (Reprinted in R.H. Thomason (Ed.), *Formal philosophy: Selected papers of Richard Montague* (pp. 247–270), 1974. New Haven, CT: Yale University Press)

Pulman, S.G. (1986). Grammars, parsers, and memory limitations. *Language and Cognitive Processes, 1,* 197–225.

Rayner, K., Carlson, M., & Frazier, L. (1983). The interaction of syntax and semantics during sentence processing: Eye movements in the analysis of semantically biased sentences. *Journal of Verbal Learning and Verbal Behavior, 22,* 358–374.

Russell, S. (1985). The compleat guide to MRS (Tech. Rep. No. KSL-85-12). Stanford, CA: Stanford University, Knowledge Systems Laboratory.

Schubert, L. (1984, July). On parsing preferences. *Proceedings of the 10th International Conference on Computational Linguistics (COLING-84)* (pp. 247–250). Stanford, CA.

Sells, P. (1985). *Lectures on contemporary syntactic theories: An introduction to government-binding theory, generalized phrase structure grammar, and lexical-functional grammar.* Stanford, CA: Stanford University, Center for the Study of Language and Information.

Shieber, S.M. (1983, June). Sentence disambiguation by a shift-reduce parsing technique. *The 21st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference* (pp. 113–118). Cambridge, MA.

Somers, H.L. (1987). *Valency and case in computational linguistics.* Edinburgh, Scotland: Edinburgh University Press.

Vater, H. (1978). On the possibility of distinguishing between complements and adjuncts. In W. Abraham (Ed.), *Valence, semantic case and grammatical relations* (pp. 21–45). Amsterdam: John Benjamins.

Wanner, E. (1980). The ATN and the sausage machine: Which one is baloney? *Cognition, 8,* 209–225.

Wehrli, E. (1984). *A government-binding parser for French* (Working paper No. 48). Geneva, Switzerland: University of Geneva, Institut pour les études sémantiques et cognitives.

Weinberg, A. (1987). Modularity in the syntactic parser. In J. Garfield (Ed.), *Modularity in knowledge representation and natural-language understanding* (pp. 259–276). Cambridge, MA: MIT Press.

Wilensky, R., & Arens, Y. (1980). *PHRAN: A knowledge-based approach to natural language analysis* (Memo UCB/ERL M80/34). Berkeley: University of California, Electronics Research Laboratory.

Wilks, Y., Huang, X., & Fass, D. (1985, August). Syntax, preference, and right attachment. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 779–784). Los Angeles.

APPENDIX

General World Knowledge

```
(assert '(class-put class thing-class nil (class) nil))
(assert '(class-put class thing-class thing nil nil nil))
(assert '(class-put class thing-class LOCATION nil (thing) nil))
(assert '(class-put class thing-class NATURAL-LOCATION nil
    (LOCATION) nil))
(assert '(class-put class thing-class ARTIFICIAL-LOCATION nil
    (LOCATION) nil))
(assert '(class-put class thing-class BUILDING nil (ARTIFICIAL-
    (LOCATION) nil))
(assert '(class-put class thing-class PHYSOBJ nil (thing) nil))
(assert '(class-put class thing-class NATURAL-PHYSOBJ nil (PHYSOBJ)
    nil))
(assert '(class-put class thing-class ANIM nil (NATURAL-PHYSOBJ) nil))
(assert '(class-put class thing-class HANIM nil (ANIM) nil))
(assert '(class-put class thing-class WRITTEN-THING nil (PHYSOBJ) nil))
(assert '(class-put class thing-class ACTION nil (thing) nil))
(assert '(class-put class thing-class STATE nil (thing) nil))
(assert '(class-put class thing-class TENSE nil (thing) nil))
(assert '(class-put class thing-class TRANSFER nil (ACTION) nil))
(assert '(class-put class thing-class MOVEMENT nil (ACTION) nil))
(assert '(class-put class thing-class PHYSICAL-ACTION nil (ACTION) nil))
(assert '(class-put class thing-class PRESENTATION nil (ACTION) nil))
```

Category Lexicon

```
;;; The category lexicon contains default selectional information for
;;; lexical categories, i.e., ' '(specifier category complement)' ', and
;;; the semantic type for each projected bar level, i.e. 0, 1, and 2.

(def-category infl      (DP infl VP)
                        (f-infl frame-statement_frame-determiner
                         frame-statement_frame-determiner))
(def-category det       (DP det NP)
                        (frame-determiner frame-statement
                         frame-statement))
(def-category com)      (nil comp IP)
                        (frame-determiner frame statement
                         frame-statement))
(def-category noun      nil
                        (frame frame-descriptor frame-descriptor))
(def-category prep      (nil prep DP)
                        (slot sf-pair sf-pair))
```

```
(def-category verb      nil
                        (frame frame-descriptor frame-descriptor))
```

Word Lexicon

```
;;; The word lexicon contains exceptional selectional information,
;;; features, and semantic information for individual words.

;; Infls
(def-meanings
    agr infl ((case))
            (meaning: <pres>
             rating: 20
             isa: TENSE)
            (meaning: <past>
             rating: 20
             isa: TENSE))
(def-meanings
    to infl () (meaning: PURPOSE
             isa: TO-CLAUSE
             cases: (nil to VP[inf])
             rating: 10) )
;; Determiners
(def-meanings
    Chrysanne DP ((number sg))
                    (meaning:    CHRYSANNE
                     rating:     20
                     isa:        HANIM))
(def-meanings
    Dan DP ((number sg))
             meaning:    DAN
             rating:     20
             isa:        HANIM))
(def-meanings
    the det ((number sg pl))
            (meaning: THE
             cases: (nil the NP)
             rating: 20))
;; nouns
(def-meanings
    letter noun ((number sg))
                (meaning:    LETTER
                 rating:     20
                 isa:        WRITTEN-THING))
```

```
;; verbs
(def-meanings
    read verb ((form inf en) (tense <past>)
                (case) (case))
               (meaning: READ
                isa: PRESENTATION
                rating: 20
                cases: (nil read (DP) (DP))
                th-grids: ((Agent Theme)
                  (Agent Benef Theme))
                th-roles:
                          ((role: Agent
                            sr-reqs: HANIM)
                           (role: Benef
                            sr-reqs: HANIM)
                           (role: Theme
                            sr-reqs: WRITTEN-THING))))
;; prepositions
(def-meanings
    to prep () (meaning: Goal
                rating: 20
                sr-reqs: ANIM)
               (meanings: Destination
                rating: 20
                sr-reqs: LOCATION))
```

Phrase Structures

```
(defvar *ps-postmod-rules*
        '((Nbar→Nbar PP)
          (Nbar→Nbar CP)
          (Vbar→Vbar PP)
          (Vbar→Vbar IP)
          (Ibar→Ibar adv)
          ))
(defvar *ps-premod-rules*
       '((Nbar→nmod Nbar)
         (Ibar→adv Ibar)
         (IP→PP IP)))
```

Argument Processor

```
;; The function needs-slot decides how much a particular slot is
;; needed. The decision depends on taxonomic properties of the verb.
```

```
;;
;; The first arg is a verb frame description. The second arg is the
;; name of the slot.
(defun needs-slot (verb-rep the-slot)
  (let ((smalltime .25)
        (medtime .3)
        (bigtime 1)
        (the-frame (car verb-rep)))
    (cond (;;slot is not needed if already preasent
           (assoc the-slot (cddr verb-rep))
           bigtime)
          ;;(;;slot is not needed if entailed by meaning of verb
          ;;(entails-slot the-frame the-slot) bigtime)
          (;;Otherwise we have a long list of heuristics, based on world
           ;;knowledge about the verb. This is sort of an extention to the
           ;;KB.
           (and (or (isa-p the-frame 'PHYSICAL-ACTION)
                    (isa-p the-frame 'VISUAL-PERCEPTION))
                (eq the-slot 'INSTRUMENT)) smalltime)
          ((and (isa-p the-frame 'PRESENTATION)
                (eq the-slot 'GOAL)) smalltime)
          ((and (isa-p the-frame 'ACTION)
                (eq the-slot 'PURPOSE)) medtime)
          ((and (isa-p the-frame 'MOVEMENT)
                (eq the-slot 'LOCATION)) smalltime)
          ((and (isa-p the-frame 'TRANSFER)
                (or (eq the-slot 'THEME)
                    (eq the-slot 'BENEF)
                    (eq the-slot 'SOURCE))) smalltime)
          ((and (isa-p the-frame 'STATE)
                (eq the-slot 'DURATION)) smalltime)  )))
```