# A Competitive Attachment Model for Resolving Syntactic Ambiguities in Natural Language Parsing

by

Suzanne Ava Stevenson

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1994

Advisory Committee:

Professor James Reggia, Chair/Advisor
Associate Professor Amy Weinberg, Co-Chair/Advisor
Associate Professor James Hendler
Assistant Professor Bonnie Dorr
Professor Norbert Hornstein

# Abstract

Linguistic ambiguity is the greatest obstacle to achieving practical computational systems for natural language understanding. By contrast, people experience surprisingly little difficulty in interpreting ambiguous linguistic input. This dissertation explores distributed computational techniques for mimicking the human ability to resolve syntactic ambiguities efficiently and effectively. The competitive attachment theory of parsing formulates the processing of an ambiguity as a competition for activation within a hybrid connectionist network. Determining the grammaticality of an input relies on a new approach to distributed communication that integrates numeric and symbolic constraints on passing features through the parsing network. The method establishes syntactic relations both incrementally and efficiently, and underlies the ability of the model to establish long-distance syntactic relations using only local communication within a network. The competitive distribution of numeric evidence focuses the activation of the network onto a particular structural interpretation of the input, resolving ambiguities. In contrast to previous approaches to ambiguity resolution, the model makes no use of explicit preference heuristics or revision strategies. Crucially, the structural decisions of the model conform with human preferences, without those preferences having been incorporated explicitly into the parser. Furthermore, the competitive dynamics of the parsing network account for additional on-line processing data that other models of syntactic preferences have left unaddressed.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The pervasiveness of ambiguity in language poses a major obstacle to achieving human-like performance in natural language understanding (NLU) systems. By contrast, people have surprisingly little difficulty in processing and resolving linguistic ambiguities. In particular, the human parser appears to immediately integrate each successive word of a sentence into a coherent syntactic structure, and people are generally not even aware that there *are* syntactic ambiguities in the sentences that they hear. These observations lead to the two motivating assumptions of this research. First, a deeper understanding of the computational processes that underlie human linguistic ability is a prerequisite for achieving comparable abilities in an NLU system. Second, increasing our understanding relies on the investigation of computational models whose behavior accounts for psycholinguistic observations. This dissertation presents a computational theory of syntactic processing in which linguistically- and computationally-justified processing mechanisms yield behavior that matches human performance in resolving syntactic ambiguities.

## 1.1 Motivation

Consider the situation of a natural language parser building the syntactic structure of a sentence beginning *Sara knows women*. If the sentence ends at this point, then the parser must attach *women* to the parse tree as the object of the verb *know*, as in Figure 1.1(a).[1] On the other hand, if the sentence continues with the word *succeed*, then the object of *know* is the subordinate clause *women succeed*. In this case, *women* must be attached as the subject of the subordinate clause, as in Figure 1.1(b). When first processing the word *women*, the parser cannot know whether it should attach *women* as the direct object of the verb or as the subject of a subordinate clause. Thus, the parser is faced with a syntactic attachment ambiguity.

Choice points such as these arise quite frequently in the processing of normal linguistic input, posing a potential problem for an NLU system, which is expected to arrive quickly

[1] For simplicity, this figure uses a traditional phrase structure representation; however, the parser developed here uses $\overline{\mathrm{X}}$ phrases.

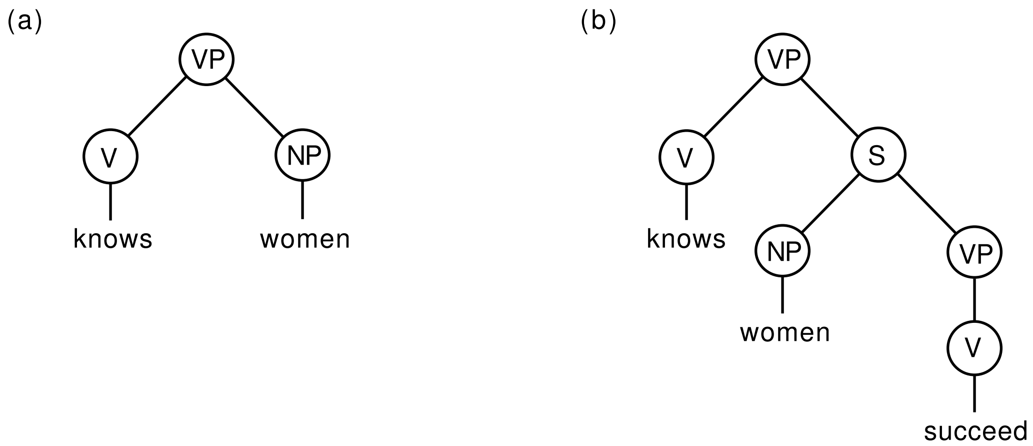


Figure 1.1: In the sentence beginning *Sara knows women*, the parser faces a syntactic ambiguity at the word *women*: the NP may attach as the object of the verb, as in (a), or as the subject of a sentential object of the verb, as in (b).

at the best structural interpretation of the input. The situation is further complicated by the fact that people generally have consistent strong preferences for a single reading of a syntactically ambiguous input. A computational parser must not only process attachment ambiguities as quickly as people do (essentially, as each word is heard or read), but should also resolve any ambiguities in a manner that conforms with human expectations.

Syntactic ambiguity thus presents a challenge in the design of natural language parsers; efficient methods for keeping track of multiple structural alternatives and for choosing between them have been an elusive goal. Building structure for all of the attachment alternatives for every word in the sentence would use a prohibitive amount of computing resources. Similarly, incorporating a large number of situation-specific heuristics to choose a preferred structure is not only inelegant, but leads to a system that is difficult to maintain and to extend. In order both to be efficient and to match the structural interpretation of the input that people expect, a parser must have a principled and parsimonious method for carefully selecting which attachment possibilities to maintain and which to discard.

In order to accomplish this, a number of design decisions must be made in developing a computational system for parsing natural language:

1. When the parser is presented with a syntactic ambiguity, will it initially build multiple structures corresponding to the attachment alternatives, or will it build only a single structure for the preferred interpretation?

2. How will the parser determine what the preferred structural interpretation of the ambiguous input is?

3. What will the parser do if the continuation of the input is incompatible with its currently preferred structural hypothesis? Will it be able to revise its initial hypothesis,

and if so, how will it proceed to do so?

Each of these interrelated design decisions raises unresolved issues in natural language processing. Computational solutions that yield efficient, human-like ambiguity resolution behavior have not yet been achieved. However, what has been so difficult to attain in NLU systems appears effortless for the human parser. The computational mechanisms that underlie human syntactic processing enable people to efficiently and consistently parse natural language. The human parser may not only be the best model for the output behavior that an NLU system is trying to achieve, but may in fact be the best model of *how* to achieve that behavior as well. This observation motivates the computational modeling of the mechanisms used by people in resolving syntactic ambiguity.

While previous NLU systems have incorporated heuristics corresponding to descriptions of human behavior, they have failed to capture the general principles underlying the computational process of ambiguity resolution. An NLU system that instead incorporates deeper principles of the human parsing process has the potential to better match human expectations in its behavior. Furthermore, a better understanding of the computational underpinnings of human behavior will form the basis for an approach that is more likely to be extensible to a wider range of linguistic phenomena. Thus, the goal of the research here is to develop a model in which human-like behavior is an emergent property of its fundamental computational assumptions. The model must be evaluated by comparing its behavior to that of the human parser within the three areas of the ambiguity resolution process described above.

## 1.2 Overview of the Competitive Attachment Model

This dissertation develops novel computational techniques for producing human-like behavior in a natural language parser, and tests their performance within a number of computer simulations on linguistic input. The techniques form the basis of a computational theory of parsing that models the processing of an ambiguity as a competition for activation among a set of structural alternatives within a hybrid connectionist network. The model provides a parsimonious account of syntactic ambiguity resolution in which the parsing decisions that conform to human expectations arise from a small set of independently motivated computational assumptions.

The first fundamental assumption is that parsing is a process of distributed decision-making within a hybrid symbolic/numeric connectionist architecture. The hybrid approach supports the direct encoding of constraint-based linguistic competence using simple symbolic features,[2] and captures the weighting of performance effects using spreading activation. Like other connectionist parsers, the model has no global controller; control of a parse is distributed among the independent processing nodes of the parsing network. However, in

[2]By a direct encoding, I mean that explicit constraint-based knowledge is used to determine the grammaticality of parse tree attachments, rather than that knowledge being "compiled out" to yield phrase structure rules that guide the parse. See Dorr (1993), Fong (1991), Kashket (1991), and Merlo (1992), for other work exploring the use of constraint-based linguistic knowledge within natural language processing systems.

contrast to other connectionist approaches, the network is not structured *a priori* according to context-free rule templates. The parser in fact makes no use of traditional phrase structure rules. Thus, syntactic phrasal nodes must actively determine their structure by trying to attach themselves together to form a valid parse tree. Valid syntactic relations among the phrasal nodes are established incrementally and efficiently through a novel form of feature-passing. The communication method relies on an integration of symbolic and numeric constraints on passing features through the parsing network. The feature-passing algorithm enables the model to establish even long-distance syntactic relations using only local communication within the network.

The communication of symbolic features through the network determines all of the valid attachment structures; numeric competition in the network is necessary to focus activation onto a winning subset of the attachments that form a legitimate parse state. The second basic assumption of the model is that competition in the parsing network is effected solely through the use of competition-based spreading activation (CBSA) (Reggia, 1987); the use of inhibitory links is not allowed. The sole use of CBSA in a network configuration of this complexity has not previously been attempted, and requires that there be additional limitations placed on the structure of the parsing network. The use of CBSA and its associated constraints yield a principled determination of which alternatives to consider when there is an attachment ambiguity. The competitive spread of numeric evidence through the restricted network structure then focuses the numeric activation onto a particular interpretation of the linguistic input, resolving ambiguities.

The competition mechanism applies uniformly at all nodes to determine the syntactic attachment choices, eliminating the need for construction-specific preference heuristics in the parser. Explicit revision strategies are also unnecessary—the competition mechanism constrains both the initial structural choices that the parser makes, as well as its potential for revising erroneous decisions. The parsing decisions that emerge from this competitive attachment process conform with human judgments of preference and acceptability. The competitive dynamics of the model also mimic finer-grained on-line processing effects in human ambiguity resolution.

The final underlying assumption of the model contributes to the computational feasibility of the parsing approach; it specifies that the network is dynamically constructed by allocating generic phrasal nodes in response to the input. The phrasal nodes are instantiated with simple symbolic features based on the features of the input words. Using dynamically instantiated phrasal nodes avoids several computational problems found in other connectionist parsers: the prior allocation of a large, fixed number of nodes; the duplication of nodes within multiple copies of rule templates; and the restriction to a fixed maximum sentence length. Since the phrasal nodes can be activated only by the input, the parallelism of the parser is constrained to nodes with overt evidence. This gives the model better scale-up potential by reducing the number of syntactic nodes and thereby decreasing the number of attachments that must be considered at any particular point in the parse. The constrained parallelism has further advantages for the model, since it leads to a better match with the degree of parallelism observed in the human processing of ambiguities.

The three fundamental assumptions of the model interact to define its competitive attachment process. A computational parser was implemented that embodies these properties and serves as a testbed for the proposed model of parsing. A large number of simulations, discussed in Chapter 5, establish the effectiveness of the competitive attachment parsing mechanism by demonstrating its consistent and correct attachment behavior across a range of structural configurations. Further tests of the parser in Chapter 7 focus on examples from the psycholinguistic literature. These simulations attest to the ability of the model to mimic human behavior along the three dimensions of ambiguity resolution discussed in Section 1.1: the degree of parallelism it displays, the syntactic preferences it exhibits, and the manner in which it revises structural decisions. Since the observed behavior of the model emerges from its underlying properties, the competitive attachment approach is proposed as a principled model of the ambiguity resolution process in the human parser.

## 1.3 Organization of the Thesis

This chapter has briefly discussed the problem of syntactic ambiguity for NLU systems, and the motivations for the approach to parsing taken here. Chapter 2 describes previous approaches to resolving syntactic ambiguities in natural language parsing; both computational and psycholinguistic models will be reviewed. Since the network architecture of the competitive attachment model is its key feature, other massively parallel approaches to NLU will be discussed as well. Chapter 3 gives the detailed computational and linguistic justifications for each of the three fundamental design assumptions discussed above. Chapter 4 then follows with a high-level overview of the competitive attachment model. The chapter presents an example parse and describes the critical attachment behaviors that result from the underlying properties of the model.

The next three chapters describe the computational parser that was built based on the proposed model, and present the results of its evaluation. Chapter 5 describes the numeric processing components of the parser, including the competitive activation functions responsible for the attachment decisions of the parser. The results of a large number of simulations are presented, demonstrating the effectiveness of the competitive attachment approach in parsing. Chapter 6 next presents the symbolic processing components of the parser, describing the symbolic features and message-passing facilities derived from the linguistic theory. The chapter demonstrates how the message-passing functions of the parser incorporate the grammatical restrictions on establishing syntactic relations among the nodes of the parsing network. In Chapter 7, the results of the parser on a number of psycholinguistically relevant examples are presented. The chapter describes in detail the correspondence between the behavior of the model in processing syntactic ambiguities, and human behavior revealed in experimental work.

Chapter 8 concludes the dissertation with a summary of its contributions and a discussion of some future directions for the research.

# Chapter 2

# Related Work

This chapter provides an overview of research related to the competitive attachment approach. Section 2.1 describes previous work in computational linguistics and psycholinguistics whose aim is to model human behavior in processing syntactic ambiguities. Because distributed network processing is a crucial property of the competitive attachment model, Section 2.2 turns to a discussion of related work in massively parallel parsing.

## 2.1 Parsing Syntactic Ambiguities

In order to parse syntactic ambiguities in a way that is compatible with human behavior, a natural language parsing model must be constrained such that the decisions it makes matches those of the human parser. Research in computational linguistics and psycholinguistics has taken a number of broad approaches in pursuing this goal. The most common has been to augment a traditional serial parser with heuristics to guide its syntactic decisions when faced with an ambiguity. Another approach has been to determine a set of well-motivated computational restrictions on a serial mechanism that will give rise to the observed human choices. A variation on this type of approach is to derive these computational constraints directly from properties of a linguistic theory. Recently, approaches using non-traditional parallel mechanisms have arisen, which emphasize the centrality of soft constraints in contrast to discrete rules or strategies. Finally, some research has focused on semantic processing accounts of the human resolution of syntactic ambiguity. This section will give a brief overview of key research in each of these areas.

### 2.1.1 Serial Models with Heuristics

Beginning over twenty years ago, researchers have tried to determined what processing strategies, in conjunction with purely grammatical knowledge, could account for human preferences in parsing ambiguous or temporarily ambiguous linguistic input. Early work took the form of positing explicit heuristics that would apply to specific syntactic constructions to guide the parser to the preferred structural analysis (Fodor, Bever, & Garrett, 1974; Kimball, 1973). Kimball's influential work inspired the best known and longest-lived model of this type, which has been developed and refined by Frazier and her colleagues (Frazier, 1978; Frazier, 1987; Frazier, Clifton, & Randall, 1983; Frazier & Fodor, 1978; Frazier & Rayner, 1982). In Frazier's serial model, parsing decisions are guided by a small number of generally applicable structural heuristics—most notably, Minimal Attachment and Late Closure. These heuristics might resolve a temporary ambiguity within a sentence in a way that is incompatible with the continuation of the sentence; in those situations, the structural analysis of the input must be corrected by explicit revision strategies. The primary shortcomings of the theory arise from the lack of generality of its preference and recovery mechanisms: multiple, unrelated preference heuristics are required, and the proposed revision strategies crucially rely on construction-specific properties.

McRoy & Hirst (1990) propose a "race-based" parser based on Frazier's model that improves the account of human behavior by providing a unified computational framework for capturing a range of structural preferences.[1] The research addresses the fact that the set of previously proposed structural heuristics were unrelated and had weak computational motivation. McRoy & Hirst demonstrate that a number of preference strategies (including Frazier's and others) can all be interpreted in terms of their effect on the time it takes for the parser to create various structures in response to an ambiguity. A single parsing strategy results, which is to prefer the structure that is built most quickly. While McRoy & Hirst's model provides a more parsimonious account of initial human preferences, the parser, like Frazier's, continues to rely on construction-specific revision strategies.

In a related approach, Gorrell (1987) proposes a "ranked parallel" model in which the rankings of syntactic alternatives are similarly based on the outcome of a parsing race—that is, the analyses are pursued in parallel, but ranked according to how quickly they are initially constructed. The ranking of parallel alternatives is claimed to underlie the observed serial behavior of the human parser, since higher ranked structures are assumed to be more salient or more readily accessible to higher-level processing. The model accounts both for some important preference data, as well as for evidence of the maintenance of multiple structures in the human parser. Although limited in computational detail and in scope (addressing only "Minimal Attachment" structures), the crucial idea of weighted parallel alternatives in syntactic processing survives in various current models (for example, compare Gibson, 1991; MacDonald, Pearlmutter, and Seidenberg, 1993).

Both Shieber (1983) and Abney (1989) propose serial models in which parsing heuristics are formulated as built-in conflict resolution strategies that guide the parser when it has more than one action it can perform in response to a new input token. Shieber proposes a rule-based shift-reduce parser in which shifting is preferred over reducing, and long rule reductions are preferred over short ones. These simple conflict resolution strategies provide an elegant account of certain cases of Minimal Attachment and Late Closure preferences. However, the model is limited in its account of on-line processing behavior; for example, the research does not address the issue of how easy or difficult it is to revise the initially preferred

[1]Although Frazier had described her model as involving a structure-building race, this was not the central focus of the work, and McRoy & Hirst were the first to describe the computational processing of such a race-based parser in detail.

structures. Abney adopts similar conflict resolution strategies within a more linguistically plausible licensing parser. The model also incorporates an explicit backtracking procedure to reanalyze erroneous initial structures; a "right-edge continuation" heuristic determines the choice point to backtrack to. Abney's model accounts for the distinction between revisable and non-revisable errors, but does not account for the range of difficulty observed in making allowable revisions.

Ford, Bresnan, & Kaplan (1982) also propose a serial backtracking parser, which is best known for the integration of lexical preferences into its parsing decisions. While previous models relied on purely structural properties to guide syntactic analysis, Ford, Bresnan, & Kaplan recognized the key role of specific lexical information in guiding a parse. The approach has a number of empirical shortcomings, arising from the formulation of the parser as a rule-based, serial, backtracking mechanism. However, the model had great influence in demonstrating the importance of incorporating lexical information into the ambiguity resolution process.

Fodor and Inoue (Fodor & Inoue, 1994; Inoue & Fodor, in press) have more recently proposed new models within the general approach of serial parsing guided by heuristics, with the goal of providing a universal parsing mechanism that matches human behavior in processing both English and Japanese. In the "information-paced" parser (Inoue & Fodor, in press), a serial mechanism determines the best structure that is compatible with the available evidence derived from the input, and a record of choice points is maintained to ease any necessary revisions. Given the power of the parser to arbitrarily go back and revise earlier decisions, Inoue & Fodor develop a constraint on restructuring that restricts the revision of thematic role interpretations. This thematic restructuring constraint captures limitations on reanalysis in English and Japanese examples. In Fodor & Inoue (1994), a "diagnosis" model is proposed in which the process of revision is made more central to their account of cross-linguistic data. The behavior of the model is determined primarily by its ability to diagnose structuring errors and recover from them, utilizing principles such as "Attach Anyway" and "Steal." Although the emphasis is shifted from preference behavior to reanalysis, these models still crucially rely on the enumeration of specific processing heuristics.

### 2.1.2 Serial Models with Computational Restrictions

The models described in the previous section all attempt to match human structural preferences by incorporating processing strategies that mimic human behavior. Another important line of research has instead focused on independently-justified computational restrictions that could lead to the observed behavior, without having to build in that behavior directly. One of the founding pieces of work along these lines is the deterministic parser of Marcus (1980), in which a small set of well-motivated computational assumptions underlie the parser's human-like behavior in processing a number of types of ambiguities. Milne (1982, 1986) extends Marcus's parser, in an attempt to give a more extensive account of human behavior in resolving lexical ambiguities and of human inability to parse well-known grammatical constructions. However, the model relies on very specific processing rules and assumptions around the amount of allowable lookahead. Kwasny & Faisal (1992) develop a hybrid connectionist model within the deterministic parsing framework that addresses some of the problems with Milne's approach. In their model, a Marcus parser is augmented with a network component that decides which of multiple actions to pursue at each point in the parse. The generalization ability of the network avoids the need for a large number of very specific rules. However, the model is limited in its grammatical scope, as well as in its ability to match human performance in processing syntactic ambiguities.

More recent work inspired by the "determinism hypothesis" has combined constraints imposed by a serial, deterministic mechanism with assumptions about the necessity of fast, incremental interpretation. Weinberg's (1991) Minimal Commitment model draws on a method of minimally specified representation ("D-theory" of Marcus, Hindle, & Fleck (1983)) to provide a principled form of limited parallelism in maintaining multiple structural analyses. In conjunction with an explicit strategy that favors immediate thematic interpretation, her model accounts for a wide range of human preference phenomena in both English and Japanese. Gorrell (in press) uses similar D-theoretic motivations, relying on restrictions on the reanalysis of precedence relations to extend the account of human behavior. In both of these models, the reanalysis process is de-emphasized, and an account of the observed range of recoverability from parsing errors is lacking.

Crocker (1992) proposes another model in which computational restrictions arising from the interpretation process constrain the parser's operations in response to an ambiguity. His principle-based syntactic parser is guided by the global requirement to maximize the interpretation of an input. Crocker claims that this fundamental assumption accounts for a number of structural preferences in English, German, and Dutch, obviating the need for explicit preference heuristics. Crocker's model matches some critical initial preference data, and underlines the importance of cross-linguistic verification of more general attachment principles. The requirement to maximize interpretation, however, is realized through the incorporation of generalized preference heuristics ("Argument Attachment" and "Deep-Structure Attachment") that are not necessary computational properties of the parsing mechanism.

### 2.1.3 Grammatically-Based Computational Restrictions

Pritchett (1992) has developed a serial parsing model in which a constraint-based approach to grammar plays a central role in defining the computational restrictions on syntactic processing operations.[2] In this framework, the grammatical constraints from the linguistic theory are directly interpreted as processing constraints on the parser. The approach promises a more principled determination of processing constraints, by deriving them from independently motivated linguistic factors. However, in practice the grammatical knowledge is not sufficiently constraining to explain human processing limitations. In order to complete his account, Pritchett must postulate an additional mechanism, the "On-Line Locality Constraint," which restricts the types of revisions that his serial parser can perform.

[2]An implementation of Pritchett's theory is described in Paolucci (1993).

The parallel model developed by Gibson (Gibson, 1991; Gibson et al., 1993) begins with a similar basis in linguistic theory.[3] The parser pursues all possible analyses of an input in parallel, and calculates the costs of maintaining each structural alternative. A principled method for determining the relative cost of each alternative derives from how well the structure satisfies its grammatical constraints. The costs are then used as the basis for pruning the space of possible structural analyses. Gibson develops a "beam search" algorithm in which structures that are close enough in cost are maintained in parallel, while those that have significantly greater costs are discarded.[4] The resulting parsing model attains an impressive coverage of human preference and recovery data. The primary shortcoming of the work is the lack of a theory to constrain the processing costs that can be postulated. To the extent that the costs can be derived from the linguistic theory, the assumptions underlying the model's behavior receive independent motivation. However, as in Pritchett's model, the linguistic theory is unable to yield sufficient processing constraints to provide an adequate match with human behavior. In order to account for a wider range of empirical data, Gibson must postulate additional cost mechanisms, such as "Recency Preference" and "Predicate Proximity," whose independent justification is less clear.

### 2.1.4 Parallel Models with Soft Constraints

Gibson's model is an example of the move toward a more continuous ranking of alternative structural analyses within approaches to mimicking human parsing. Many recent psycholinguistic studies present evidence that human behavior is guided by the application of a number of soft constraints (for example, MacDonald, 1994; Spivey-Knowlton, Trueswell, & Tanenhaus, 1993; Taraban & McClelland, 1990). The results emphasize the importance and timing of information that derives from individual lexical entries. To account for this data, MacDonald, Pearlmutter, & Seidenberg (1993) outline an approach to a constraint-based lexicalist parser. The model depends crucially on numeric competition among activated partial structures to resolve ambiguities. The activation levels that determine parsing preferences derive entirely from differential frequencies of lexical associations. MacDonald and her colleagues do not propose an underlying grammatical or computational explanation of the relevant frequency distributions, assuming instead that they merely reflect the statistical patterns of the language.[5]

Although there are no implemented parsing systems within this paradigm, the proposal of

[3] See Gibson (1987) and Clark (1988) for earlier proposals within this framework.

[4] It is interesting to note that the beam search restriction that Gibson imposes on his parser can be interpreted as a high level view of the type of competitive processing that naturally falls out of the competitive attachment architecture developed here: In a competition for activation, alternatives that are close in activation will compete over a lengthy period of time, while a great difference in activation will allow one alternative to quickly dominate another.

[5] The research here complements that of MacDonald and her colleagues in searching for underlying differences in computational complexity that could account for the observed frequencies of possible structural configurations.

Schubert (1984, 1986) anticipates the importance of developing computational mechanisms to support the integration of multiple preference sources. Schubert sketches a framework in which numeric combination of syntactic and semantic influences plays a key role in accounting for human structural preferences. Factors such as recency of attachment and strength of expectations interact to determine the best interpretation of an input. Schubert assumes a "full-paths" parser, which pursues all analyses in parallel and relies solely on the numeric weights to determine the most preferred structure. The proposal leaves unaddressed the issue of revising erroneous initial preferences.

Other computational approaches that rely on the satisfaction of soft constraints will be discussed in the section below on massively parallel parsing.

### 2.1.5 Semantically-Based Models

A number of approaches to modeling human behavior in processing syntactic ambiguities have proposed that structural preferences arise from semantic rather than syntactic sources of constraining information. Within the sentence processing literature, Altmann, Crain, and Steedman have proposed that discourse context and presuppositional constraints are instrumental in disambiguating syntactic ambiguities, and that syntactic structure itself does not play a role (for example, Altmann, 1988; Altmann & Steedman, 1988; Crain & Steedman, 1985). However, because of their fundamental assumptions, they are unable to account for the purely syntactic influences on initial parsing preferences that have been identified (for example, see the discussion in Gibson, 1991). Furthermore, the context-based proposals fail to explain how semantic and pragmatic information alone can guide the reanalysis of erroneous syntactic structure.

In Artificial Intelligence, it is not uncommon for researchers to propose parsing models based on "semantic grammars," in which syntactic knowledge is assumed to play a peripheral role. Cardie & Lehnert (1991) claim that their parser mimics a number of so-called syntactic preferences with a semantic account. However, some effects in their model in fact arise from syntactic processing (for example, reactivation of antecedents), while others arise from semantic processing (for example, the filled-gap effect). Furthermore, there are effects such as structural constraints on extraction that their syntactically-impoverished model cannot explain. Although they demonstrate that certain so-called syntactic processing effects have an alternative semantic explanation, they are unable to give a unified account of syntactic processing phenomena purely in terms of semantics. In related work, Jurafsky (1991) proposes a unified semantic grammar approach to parsing. His model is also unable to explain some key syntactic distinctions in a well-motivated way; for example, as in the Cardie & Lehnert model, constraints on extraction pose difficulties for his semantic gap account. Thus it appears that while semantics and constraints on interpretation contribute to the resolution of ambiguity, some purely syntactic properties must be taken into account as well.

### 2.1.6 Summary

In conclusion, parsing models have made tremendous progress in their ability to mimic human structural preferences. The move from discrete heuristics to continuous measures of acceptability shows great promise. However, most approaches still rely to some extent on stipulating constraints rather than deriving the processing restrictions from more fundamental properties of linguistics or computation. Furthermore, the emphasis has been on accounting for initial preferences and processing breakdown. Revision processes have only recently received critical attention, and models of the recovery process thus far have been strategy-based, leaving a need for more integrated accounts of revision.

## 2.2 Massively Parallel Parsing

There has been a great deal of interest in exploring the fruitfulness of non-traditional parallel architectures for natural language processing. This section will describe work on connectionist parsing models using both local and distributed representations, and work aimed at addressing some of the problems that arise in connectionist approaches. The section will also discuss a number of parallel architectures that are not purely connectionist, but that exploit massively parallel or distributed processing technology to address open problems in parsing natural language.

### 2.2.1 Connectionist Parsing Models

Connectionist parsers have commonly used localist representations of rule-based syntactic knowledge,[6] in which the network is structured *a priori* to represent context-free rule templates (for example, Cottrell, 1989; Fanty, 1985, Selman & Hirst, 1985). This type of model is limited to representing sentences of a maximum fixed length from a context-free language. In spite of the shortcomings for natural language parsing, the work of Cottrell (1989) was important not only for its modeling of word-sense disambiguation (a research problem that will not be addressed here), but for its account of "Minimal Attachment" behavior in human parsing. In Cottrell's system, simpler ("minimal") structures accrue activation more quickly than complex structures, elegantly explaining the observed Minimal Attachment preferences. Howells (1988) proposes a more dynamic localist model of parsing, but it too is limited to knowledge of tree structure which is representable as context-free rules. Furthermore, the parser cannot be interpreted as an on-line model of human parsing since it depends on simultaneous activation of the input tokens.

The localist context-free parsing techniques appear to be overly simplistic for the problem of understanding human language. However, the use of distributed representations, while promising more flexibility, has acheived limited success. Hanson & Kegl (1987) use distributed representations to exploit the learning ability of connectionist networks, but their

[6]In a localist representation, processing nodes and symbolic concepts stand in a one-to-one relationship.

model is limited to part-of-speech tagging and prediction of the next syntactic category in a linear pattern. Chalmers (1992) proposes a model which can generalize a simplified active/passive transformation over distributed representations of sentential input, but the syntactic capability of the model is again quite restricted. McClelland & Kawamoto (1986) also exploit the generalization properties of distributed representations, but their model is limited to matching phrases to the roles they play in a sentence. Sopena (1992) develops a model in which roles are determined within embedded syntactic structures, but the syntactic capabilities of the network are again quite impoverished.

To explore the possibilities of connectionism for capturing more realistic human linguistic knowledge, Rager & Berg (1992) use existing connectionist techniques to encode a subset of Government-Binding theory. However, the restrictions imposed by the connectionist encoding scheme necessitate a large and unwieldy representation of the syntactic knowledge. Other researchers have focused on extending the connectionist techniques themselves to enable more direct and elegant representational mechanisms for parsing. Henderson (to appear) presents a model of parsing founded on connectionist techniques that allow the simultaneous binding of multiple variables. This framework not only allows him to represent a linguistically plausible grammatical formalism, but also leads to some psycholinguistically relevant processing behavior regarding long-distance dependencies and center-embedding.

The major focus in extending connectionist methods for natural language parsing has been to develop techniques to enable more dynamic behavior. In order to support their model of semantic disambiguation, Waltz & Pollack (1985) had to use a traditional chart parser as a front end to their network creation process. This inspired Pollack (1985) to explore true connectionist techniques in the form of multiplicative connections and context-adjusting processing nodes for achieving the network dynamism necessary for parsing. However, the contribution is in the exploration of low-level connectionist techniques, and not in the proposal of a natural language parsing model. Charniak & Santos (1987) develop a technique of shifting information sequentially across the nodes of a parsing network to avoid the problems of a static network structure, but it is unlikely that the technique would scale up to realistic sized parsing problems. More recently, Reilly (1992) combines the techniques of recursive auto-associative memory (RAAM) and simple recurrent networks (SRNs) to exploit the possibilities of incrementally building embedded structure, as is required by the on-line parsing problem. However, as Reilly notes, the RAAM technique does not provide adequate generalization ability, limiting the practicality of the approach.

### 2.2.2 Other Massively Parallel Approaches

Other distributed processing approaches have in general fared better than pure connectionist methods at enabling the development of more comprehensive natural language parsing systems. Small (1981) proposes a model in which input words instantiate independent processing nodes according to the individual lexical entries of the words. The model is able to successfully resolve a number of lexical and semantic ambiguities using only distributed processing among the "word experts." Each word expert has to be completely hand-coded—the individual processors are not constrained to perform the same algorithms as in most massively parallel frameworks. Small's model is therefore on the other end of the continuum from the simple, uniform processing nodes of a connectionist approach. Abney & Cole (1985) develop an actor-based implementation of Government-Binding theory, which also lacks uniformity of processing. Some nodes represent individual parsing entities, while others encode entire sub-modules of the linguistic theory. Because of the centralization of much of the knowledge within these complex processors, the model has difficulties with resolving conflicts among the knowledge sources.

The Active Production Network (APN) approach of Jones (1987) moves away from these less constrained distributed models towards a more connectionist-like approach. The APN framework captures syntactic knowledge in rule-based network templates, and uses spreading activation to encode feature co-occurrence among the processing nodes. The use of rules in the network is essential to providing a local environment for the binding of features. The parser developed by Lin (1993) relies instead on a message-passing implementation of Government-Binding theory, providing increased flexibility for capturing grammatical relations in the network. However, Lin's approach does not use simple processing nodes that perform the same algorithms across the network; the phrasal nodes of his parser execute different message-passing computations depending on the category of the phrase. In either Jones' or Lin's models, ambiguity resolution procedures are not an integral part of the parsing mechanism, and would have to be added onto the existing parser. In neither case does observed human behavior in the processing of ambiguity follow directly from the proposed computational mechanisms.

Kempen and Vosse (Kempen & Vosse, 1989; Vosse & Kempen, 1991) propose a computational model of human parsing that is similar in spirit to the approach developed here. Their model exploits the use of hybrid symbolic/numeric techniques within a network of simple, uniform processors. The resulting Unification Space parser is a massively parallel, rule-based approach that models parsing as a simulated annealing process. In contrast to most other massively parallel parsers, the model matches a wide range of human structural preferences, in Dutch as well as in English. As in the proposal of Schubert (1984), the preference behavior of the Unification Space model is determined primarily by specific numerical strength and decay values. Attempts at reanalysis of erroneous structures appear to be limited only through the mechanism of decay.

### 2.2.3 Summary

At this point in the development of connectionist techniques, the ability to support high-level modeling of human parsing has not been demonstrated.[7] The issues of dynamically encoding structure and representing the non-local relationships necessary in syntax remain open problems. On the other hand, massively parallel models that incorporate symbolic capabilities show promise in their ability to model human linguistic performance. This success

[7]However, recent work on Optimality Theory (Prince & Smolensky, 1993) has potential to relate high-level symbolic representations of linguistic knowledge to well-established connectionist processing techniques.

encourages the exploration of hybrid connectionist techniques that can support syntactic ambiguity resolution, while at the same time reaping the benefits of distributed parsing within a network of simple, uniform processing nodes.

# Chapter 3

# Architectural Assumptions

This chapter describes the fundamental architectural assumptions that underlie the principled model of human parsing developed here. The competitive attachment model is able to predict a number of critical attachment behaviors of the human sentence processor. The model is highly restricted in that its computational architecture is constrained by independent computational and linguistic factors. The behavior of the model is not explicitly built into its architecture, but rather emerges from the interaction of these independently motivated assumptions. Although it may be possible to achieve human-like performance with a less constrained parser—for example, by building in human behaviors as explicit heuristics—a model that does so yields little insight into the computational properties underlying the human ability to process attachment ambiguities. By developing a model whose attachment behaviors fall out from its restricted computational design, we can gain a deeper understanding of the problem of natural language parsing and how to capture this complex behavior in a computational system.

The design of the competitive attachment model is based on three primary architectural assumptions that determine the essential characteristics of the parser. The first assumption is that the model is a hybrid connectionist network in which processors that represent syntactic phrases locally communicate simple symbolic features and numeric activation to determine their parse tree structure. This framework entails that there is no global controller in the parser; rather, all parsing decisions are made in a distributed fashion by the syntactic phrasal processors. The chief restrictions on the model follow from two additional assumptions: There are no inhibitory connections between nodes in the network, and the network structure is determined dynamically in response to the input. This chapter describes these three fundamental computational assumptions that constrain the architecture of the parser. Each of these design decisions is discussed with regard to its computational and linguistic motivations.

## 3.1 A Hybrid Connectionist Parsing Network

Recent research in Artificial Intelligence (AI) has focused on determining the relative merits of two competing paradigms of human information processing. Traditionally, intelligent behavior has been modeled within a serial, symbolic processing paradigm. In this approach, a powerful global processor manipulates symbols that represent information in the problem domain. The symbolic information typically takes the form of rules that encode the general structure of a solution to a given class of problems. These rules are applied to an input to build a structure that corresponds to a solution to a particular problem.

Newer approaches in AI have described intelligent processes as the global behavior that emerges within a massively parallel network of computationally simple processing units (for example, Anderson, 1983; Fahlman, 1981; Feldman & Ballard, 1982; McClelland et al., 1986; Reggia & Sutton, 1988; Rumelhart et al., 1986; Smolensky, 1988). Each processor can only perform simple computations on numeric values, and communicate the results in parallel to all of its neighboring nodes in the network. The solution to a problem consists of a pattern of numeric activation distributed across the processors. There is no process that centrally controls or interprets this distributed information, hence the global behavior of the system arises solely from local numeric computations in the network. This connectionist paradigm has strengths and weaknesses that are complementary to the traditional models of intelligence;[1] this fact has led to investigations of combining the two approaches in so-called *hybrid* models of intelligence (for example, Hendler, 1987; Kimura, Suzuoka, & Amano, 1992; Slack, 1991; Vosse & Kempen, 1991; Waltz & Pollack, 1985; Wermter & Lehnert, 1989).

A close examination of the problem of structural disambiguation in natural language parsing has motivated the design of a hybrid model in the research presented here. The process of resolving an ambiguity has two components: identifying the grammatical attachments for a syntactic phrase, and choosing the preferred attachment from among those. Thus, one aspect of structural disambiguation involves the *competence* of the parser, since linguistic knowledge determines which attachment alternatives are grammatical. The other aspect of the task brings in *performance* factors; computational restrictions prune the space of attachment possibilities, and determine which of the valid attachments to adopt.[2]

This bipartite division of the factors involved in structural disambiguation mirrors the opposing approaches to modeling intelligence in AI. Traditional symbolic processing models have proven successful at encoding and manipulating discrete competence knowledge. This paradigm allows a natural language parser to directly represent the symbolic linguistic knowledge needed to describe tree structures and the grammatical relationships within them. Connectionist models, on the other hand, have demonstrated their usefulness for integrating the multitude of factors affecting performance. Connectionist approaches to NLU can naturally simulate the extralinguistic conditions, such as priming effects, that play an important role in determining the preferred interpretation of a sentence (for example, Cottrell, 1989; Waltz & Pollack, 1985). The motivation for a hybrid approach to structural disambiguation arises from the necessity of capturing within a single model the abilities of each of these two

[1]For an in-depth discussion of the potential weaknesses of connectionist approaches within the domain of language processing, see Pinker & Prince (1988).

[2]By separating the structural disambiguation task into these two components, I am not claiming that they are independent subtasks. The division is simply a characterization of the types of information brought to bear on the problem.

information processing paradigms.

The question, of course, is how to combine these divergent approaches in a principled way. In fact, linguistic and computational factors independently motivate the basis for the competitive attachment model as a massively parallel network integrating the two techniques of symbolic constraint satisfaction and numeric spreading activation. First of all, a network architecture allows for a direct mapping of the necessary linguistic knowledge into the computational framework. A recent advance in linguistic theory has been to adopt a so-called "principles and parameters" approach to capturing human linguistic knowledge (Chomsky, 1981, 1986a). This type of approach is a reaction to serious drawbacks of rule-based systems, in which the construction-specific nature of rules can lead to extremely large grammars, with rules that do not generalize well to new constructions or to other languages. Government-Binding theory (GB), which embodies the principles and parameters approach, replaces traditional rules with a set of simple features and general (non-construction-specific) constraints. In GB, the validity of syntactic structures is achieved by locally satisfying the grammatical constraints among neighboring syntactic phrases. The distributed network model developed here is able to directly encode this formulation of linguistic knowledge as a set of simultaneous local constraints. Syntactic phrases are independent processors that actively try to satisfy the constraints on potential attachments through the strictly local communication of relevant grammatical features.

In addition to its ability to directly capture a well-motivated linguistic theory, the nature of the proposed model also allows it to avoid the computational problems associated with traditional rule-based systems. Even with a parallel architecture, memory limitations are quickly exceeded if reasonable coverage of a language is attempted through construction-specific rules. Furthermore, it is difficult to extend a set of rules to increase its coverage of a language, and a set of rules developed for one language can rarely serve as the basis for the grammar of another language. Representing grammatical knowledge as a set of simultaneous declarative constraints makes the parser's knowledge base more compact and more amenable to future extensions (Berwick, 1982).

Further computational considerations of the parsing process motivate the proposal of a hybrid architecture that integrates spreading activation into the symbolic network model. First, a local, discrete decision-making process among parallel alternatives can get "stuck" in an inconsistent solution without the benefit of a global overseer. By gradually amassing activation within a mutually supporting set of attachments, spreading activation methods can avoid the complex communication protocols required by purely symbolic approaches to local, distributed decision-making. Second, spreading activation is able to capture diverse performance factors within a single mechanism. For example, recency, frequency, and salience can all be translated directly into activation through the use of decay, weights, and priming, as shown in Figure 3.1. Level of activation then provides a meaningful way of comparing the relative influence of these various performance effects, as well as a means of encoding the result of their interaction.

In conclusion, this first computational assumption establishes the basic framework of the model as a massively parallel, distributed network that integrates aspects of symbolic and

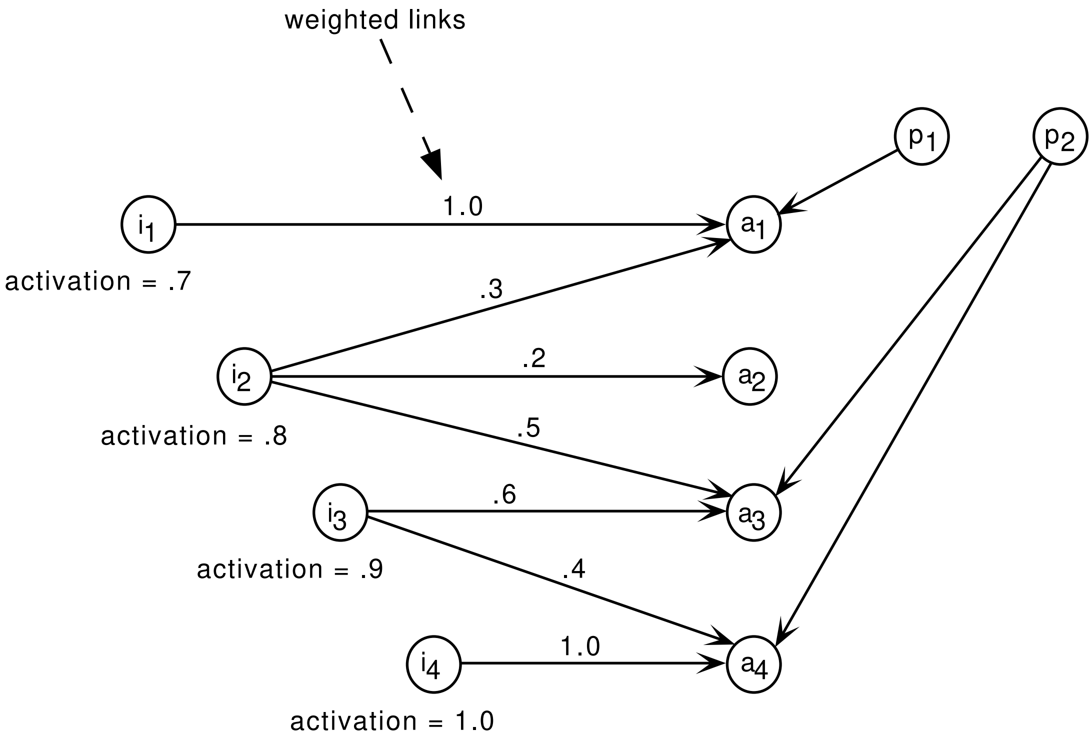


Figure 3.1: Sample network in which spreading activation can integrate diverse performance factors. The $i$ (input) nodes are activated serially; their activation decays over time, capturing recency effects. The weights from the $i$ nodes to the $a$ (answer) nodes can encode frequency information. The $p$ nodes represent salient information; they prime certain $a$ nodes by outputting activation to them. The input to the $a$ nodes is a function of the weighted activation from the $i$ and $p$ nodes, and thus the level of activation of each answer is determined by a combination of these recency, frequency, and salience effects.

numeric processing. The parser has the ability to manipulate symbolic linguistic information and to build structure, but processing is limited to local communication of simple features and numeric activation. Spreading activation captures the performance factors involved in syntactic processing, such as weighing evidence for alternative attachment possibilities. Since there is no global overseer, control of the parsing process is distributed among the processors that represent syntactic phrases. Each syntactic processor must immediately and actively try to group itself with previously structured phrases in the developing parse tree. This active, distributed parsing process is constrained in a principled way by the other fundamental properties of the model, described below.

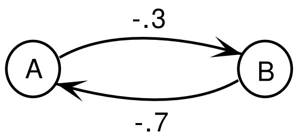


Figure 3.2: Competition through direct inhibition. Nodes A and B suppress each other's activation by sending negative output to each other. For example, the output from A to B is $-.3$ times the activation of A.

## 3.2 Competitive Behavior in the Model

Typically in a connectionist network, a processing unit represents some symbolic feature or hypothesis relevant to the problem domain, and the numeric activation of the unit encodes the strength of belief in that hypothesis. A "solution" in the network then consists of the highly active units. The independent processing units converge on a solution by locally communicating numeric activation among themselves. A unit receives activation from each of its neighbors in the network, combines this input with other information to determine its new activation level, then in turn outputs activation to its neighbors based on its current level of activity. This iterative process is stopped when the network reaches a pre-defined acceptable state—for example, when the activation level of each node is above some threshold $\theta$ or below some threshold $\phi$. The goal is that the set of nodes that are active in the acceptable state represent a well-defined solution to the given problem.

Since a consistent solution cannot allow units representing incompatible alternatives to be active simultaneously, competition among them plays an important role in the successful convergence of a network. In most connectionist models, processing units representing incompatible hypotheses compete by directly suppressing each other's activation levels. The relationship between two competing nodes is represented by an *inhibitory* connection between them—a connection that has a negative weight associated with it, as shown in Figure 3.2. Each unit sends a negative activation value to its incompatible neighbor by multiplying its output activation by the negative weight associated with the inhibitory connection. The negative input then acts as a direct influence to lower the activation of the receiving node. The intent is that as each node attempts to suppress the other, one will steadily decrease in activation, while the other's activation increases. Eventually the latter node will be the only active one of the pair. Thus, the use of inhibitory links can force a winner-take-all competitive behavior that ensures that only one of a set of incompatible nodes may be active when the network reaches an acceptable state.

An alternative approach to producing useful competitive behavior is through a technique called competition-based spreading activation (CBSA) (Peng & Reggia, 1989; Reggia, 1987; Reggia, Peng, & Bourret, 1991; Sutton, 1992). In this approach, competing processing units vie for a portion of the fixed amount of activation being output from a common

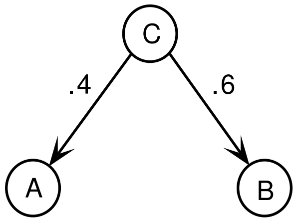


Figure 3.3: Competition through competition-based spreading activation. Nodes A and B have no direct connection to each other; instead, they indirectly compete for the output allocated to them proportionally from node C, as in equation 3.1.

source (another processing unit); see Figure 3.3. The source node uses a CBSA method to determine how to apportion its available output activation, sending more of its output to the competing node with the higher activation. For example, a simple CBSA output method is the following:

$$o_{ji} = \frac{a_j w_{ji}}{\sum_k a_k w_{ki}} \cdot a_i \tag{3.1}$$

where:

$o_{ji}$ is the output from node $n_i$ to node $n_j$;
$a_i$ is the activation of node $n_i$;
$w_{ji}$ is the (positive) weight on the connection from node $n_i$ to node $n_j$; and
$k$ ranges over all nodes connected to node $n_i$.

The distinguishing feature of this output function is the reference to the activation levels of $n_i$'s destination nodes.[3] This function says that the source node $n_i$ will send to the destination node $n_j$ a percentage of $n_i$'s activation based on the ratio of $n_j$'s activation to the sum of the activations of all of $n_i$'s destination nodes. Thus, the more active destination nodes receive a higher proportion of $n_i$'s output. Although within this approach two competing nodes do not directly suppress each other's activation, apportioning the output to them based on their current level of activation can also achieve a winner-take-all behavior (Reggia, 1987). In this case, it is the gradual decrease of positive input, rather than the increase of negative input, that eventually "turns off" all but one of a set of competing nodes.

In summary, there are two potential methods for achieving competitive dynamics within a connectionist network. The use of direct inhibition is an approach in which the actual structure of the network encodes the necessary competitive relationships. Competition is brought about by means of explicit inhibitory links between any two incompatible alternatives. By

[3]This is in contrast to a traditional output method, in which the output that a source node sends to each destination node is affected only by the activation level of the source and the weight on the outgoing connection.

contrast, with the use of CBSA the competitive relationships are implicitly determined by the output function of a node, with no structural effect on the network. Competition is instead realized through a method of allocating activation from a source to its neighbors. Determining which of these approaches is most appropriate to the model here requires first considering the role of competition within the parser—that is, what do the processing nodes represent, and what competitive relationships are necessary in order to achieve a consistent network solution?

As stated in the previous section, the competitive attachment parsing network is composed of processors representing syntactic phrases. Furthermore, additional processing units represent the potential attachments between those syntactic phrases.[4] Figure 3.4 shows part of an example network.[5] The syntactic nodes in the network are activated in response to the input words of a sentence, and remain active throughout the parse (although their level of activation does decay over time). An attachment node, on the other hand, is created between a pair of syntactic nodes that are potential sisters in the developing parse tree; it becomes active to the degree to which there is grammatical and extragrammatical evidence for its inclusion in the tree. When the network is in an acceptable state, the set of active attachment units will represent the attachments comprising the syntactic structure that the parser is building. Clearly not all potential attachments can be included in the parse tree; some attachments are incompatible with each other within a well-defined tree structure. Thus, since the network must focus activation onto a subset of the attachment nodes, it is those nodes that are relevant to an examination of the competitive relationships in the network.

For example, the subnetwork depicted in Figure 3.4 has more attachment nodes than are allowed to be activated in a valid parse tree. Here the verb *know*, represented by the V node, may have the NP node or the IP node as its sister in the parse tree; the possible attachments are represented by attachment nodes $a_1$ and $a_2$ respectively. (In the parser, the object of *know* is attached as a sister to the V node.) Only one of these attachment nodes may be active when the network settles, since *know* can only have a single object; $a_1$ and $a_2$ are therefore competing nodes in the network. This fact may be represented either by creating a direct inhibitory link between them, as in Figure 3.5, or by having the V node employ a CBSA output function to bring about their indirect competition, as in Figure 3.6.

CBSA was selected as the mechanism by which to choose between sister attachments because the approach directly captures the relevant characteristics of this type of competitive relationship. It is natural for a node to have information about the type of sisters it prefers,

[4]Explicit representation of an attachment as a processor is a necessity in this type of model. A connection in the network between two syntactic units cannot represent an attachment in the parse tree, since there would be no way of distinguishing which connections between phrases are the actual syntactic attachments and which are simply network communication links.

[5]In the parser, attachment nodes represent a *sister* relation between phrasal nodes in the parse tree. In this and all other figures, attachment nodes are depicted as small squares, which are white when inactive and black when fully activated. Phrasal nodes are shown as large circles with the category of the phrase displayed inside.

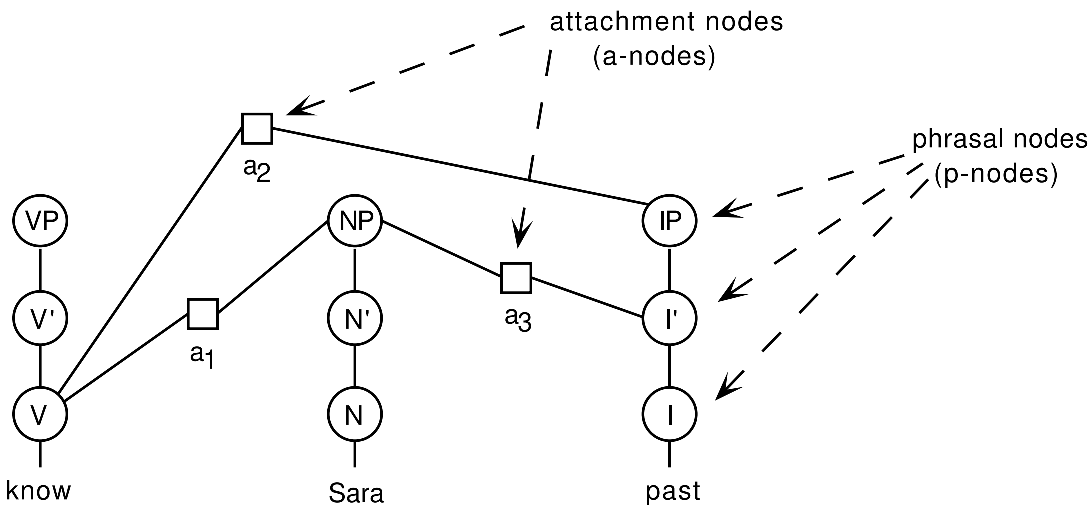


Figure 3.4: Part of the network of syntactic phrases and attachments that is generated given the input sentence: *I know Sara ran.*

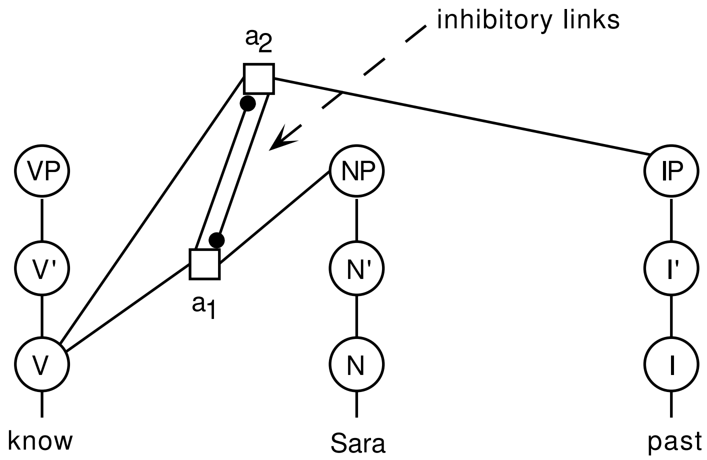


Figure 3.5: Competition between $a_1$ and $a_2$ manifested by an explicit bidirectional inhibitory link between them. The V node outputs the same activation to each of $a_1$ and $a_2$, which then try to suppress each other with negative output.

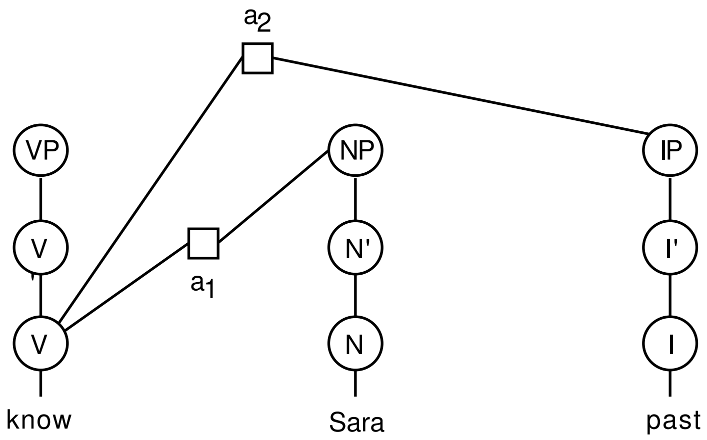


Figure 3.6: Competition between $a_1$ and $a_2$ is brought about by the V node's CBSA output function. The output from V to $a_1$ and $a_2$ is proportional to each one's activation; $a_1$ and $a_2$ have no direct connection to or affect on each other.

and for it to apportion its output activation among potential sisters accordingly. For example, the lexical entry for a verb specifies what kinds of complements it takes, and what the relative preference for those complement possibilities are. A V node in the network uses this information, which it inherits from its lexical entry, to directly affect the symbolic features and numeric activation passed to the V's attachment nodes. It is a straightforward extension for the V node to bring about the indirect competition between its attachment nodes by using a CBSA function to determine its output to them.

By contrast, it is much less justifiable for the competing attachment nodes to have a direct influence on each other, as would be required by an approach to competition based on using inhibitory links. The semantics of an attachment within the parse tree is determined solely by the properties of the two phrases that are attached, and not by the relation of the attachment to other potential attachments.[6] Using inhibitory links to create a direct relation between any two competing attachment nodes would therefore not conform to the direct mapping established thus far in the model between the parse tree semantics and the network structure.

CBSA therefore seems the most natural and effective means for accomplishing this primary competitive behavior in the model. The goal of restrictiveness motivates taking a uniform approach to competition by prohibiting the use of inhibitory links as an alternative means of achieving competitive behavior. Thus CBSA was adopted as the sole competitive mechanism within the parser. It will become clear in the following sections that this is a highly restrictive assumption, with far-reaching consequences for the design and behavior

[6]The knowledge that the attachments are mutually exclusive is external to the attachment itself, and is in fact a property of the phrases being attached. The CBSA approach recognizes this by encoding the property of mutual exclusiveness within the phrasal nodes, from which it originates.

of the model. For that reason, it is especially important to note that, as with the first fundamental design decision, this one too is well-motivated by computational and linguistic considerations.

Several computational benefits directly follow from the sole use of CBSA to effect competition, and the lack of inhibitory links in the model. First, some space efficiency is gained, since the need for a large number of inhibitory links is eliminated. In a parsing network of $n$ phrases, there are $\mathcal{O}(n^2)$ possible attachments;[7] thus, in an approach based on pairwise inhibition between competing attachment nodes, there are $\mathcal{O}(n^4)$ potential inhibitory links.[8] Second, the use of CBSA is actually more flexible in its ability to easily allow for a multiple-winners-take-all relationship among a set of competing nodes (Reggia, 1987). This ability is crucial to the correct behavior of the stack data structure in the parser, which is discussed in Section 4.1.3.

A third important computational benefit follows from the restrictiveness of these assumptions. If direct inhibition is used, *any* two processing units in a network may be made to compete by establishing inhibitory links between them. Using inhibitory links, a parser could create all possible attachments, and direct inhibition would in principle be able to prune the set of attachments down to those that form a valid tree. The use of CBSA, on the other hand, relies on the competing units being connected to a common unit that brings about the competition between them. The network must be structured so that any incompatible attachment nodes that are created are able to compete through CBSA. The parser is forced to perform some of the pruning of incompatible attachment nodes by actually limiting which attachments are created.[9] The effect is to reduce the number of attachments that are established at each step of the parse. While the total number of possible attachments among $n$ phrases is $n(n+1)$, a maximum of $4n$ of these is allowed under the restrictions of the CBSA approach.

The restrictiveness of the network structure that falls out from the CBSA assumption is also motivated by linguistic considerations. An important aspect of Government-Binding theory is the notion of *locality*: constraints on grammatical relations among syntactic phrases apply within very restricted local domains—for example, between sisters in the parse tree (Chomsky, 1986b; Rizzi, 1990). By requiring a direct mapping from the parse tree semantics to the network structure, the model ensures that a local relation between two nodes in the

---

[7]The exact number of possible attachments among $n$ phrases in the network is: $\sum_{i=1}^{n} 2i = n(n+1)$.

[8]In one formulation of the network, which used CBSA where possible, and direct inhibition for all other pairs of incompatible attachment nodes, the precise number of inhibitory links required was $(n^4 - 4n^3 + 8n^2 - 5n)/6$. Thus the sentence *Women know Sara ran*, whose representation uses 8 syntactic phrases, required 420 inhibitory links. Note that this large number is in fact a reduction over a pure direct inhibition approach, since CBSA was employed wherever possible.

[9]Note that this means that the use of CBSA, like the use of inhibitory links, does in fact entail that the structure of the network be altered in response to the requirements of the competitive relations. However, with CBSA the necessary change in structure is a *decrease* in the size and complexity of the network, while the use of inhibitory links involves a dramatic *increase* in size and complexity. Precisely how the restrictions imposed by CBSA determine which attachments can be created will be made clearer in Section 4.1.3.

network is a local relation within the parse tree represented. That is, two nodes are directly connected in the network only if they are neighbors in the parse tree. Thus, locality in the network is equivalent to locality in the parse tree. By comparison, allowing inhibitory links between arbitrary attachment nodes could clearly violate the locality restrictions derived from the parse tree semantics. It would be possible for two nodes that are distant in the parse tree to directly communicate with and affect each other in the network. In that situation, adhering to local communication in the network would not guarantee that the locality conditions of GB were respected. By prohibiting inhibitory links, the model imposes limitations on the network structure that help to maintain the locality constraints of GB.

## 3.3 Dynamic Instantiation of Syntactic Phrases

In the design of a hybrid symbolic/connectionist processing model, one of the issues that must be decided is how structure is manipulated in order to solve a problem. Traditional symbolic models typically build new structure as the solution to a problem, while connectionist models solve problems by activating a subset of the pre-existing structures that have been built into the network. The traditional approach imposes fewer constraints on the possible solution structures, since it is not necessary for the solution to exactly match a pre-determined, fixed structure. Rather, symbolic rules provide a means for creating and combining structures in a general way, allowing a more flexible response to conditions in the input. This flexibility has a price, however, in that any needed structure must be computed on-line during the problem-solving process.

The connectionist approach avoids this complex, on-line computation by replacing the creation of solutions with the recognition of solutions. One effect of this technique is to more highly constrain the solution space by restricting the possible solutions to a set of pre-computed, fixed patterns. But while the restrictiveness of this approach avoids the potentially expensive on-line computation, it entails a different kind of inefficiency. Because all possible solutions must be anticipated in the structure of the network, there can be a great deal of redundancy in the network that can use an inordinate amount of space.

This is in fact a potential problem in many connectionist models of natural language parsing. Grammatical knowledge in these systems has typically been encoded in context-free "rule templates," such as those shown in Figure 3.7 (Cottrell, 1989; Fanty, 1985; Selman & Hirst, 1985). The problem is that the number of each type of rule template in the network must be the maximum necessary to parse some arbitrary sentence in the language. But this number might be much larger than that needed to parse the vast majority of sentences that the model would be exposed to.[10] Although not a real disadvantage in demonstration

[10] Of course, because of the unbounded recursive structure of natural language, there *is* no maximum number of templates for a particular rule type that would allow all the sentences of a language to be parsed. Thus, this type of connectionist approach is in fact limited to parsing fixed length sentences. To avoid this limitation in practice requires imposing a relatively high maximum on the number of templates. The point is that any maximum adopted—for example, one based on when human performance breaks down—would

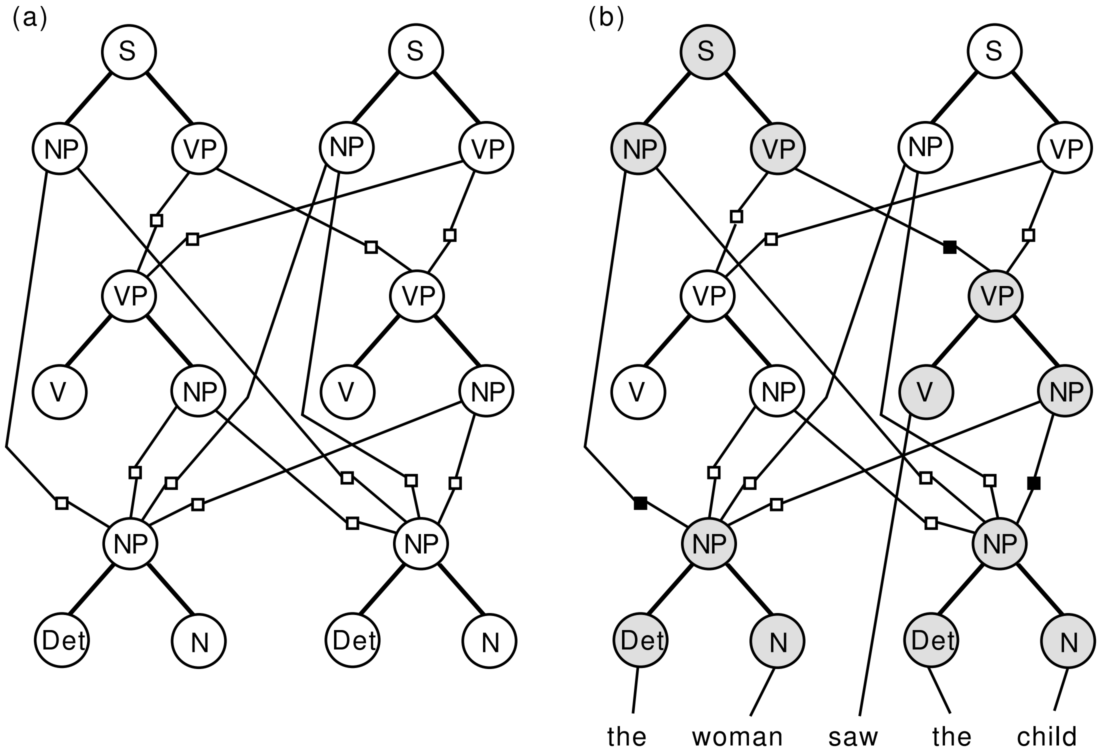


Figure 3.7: (a) A typical representation of tree structure knowledge in a connectionist parser. Multiple copies of a rule template exist for representing multiple instances of the application of the rule within a single sentence. The small squares represent "binding nodes" that link together the applicable rule templates to form the parse tree structure. (b) The shaded nodes are those that are active given the input *The woman saw the child.*

systems, the issue of how such an approach will scale up to more realistic-sized parsing applications cannot be ignored.

The intention here is to find a middle ground between these two contrasting problem-solving methodologies. Since the goal of this research is a restrictive model of on-line parsing, the standard traditional approach may be too unconstrained in the structures it allows, and too inefficient in the time needed to compute the necessary parse trees. However, the constraints of strict connectionist methods may be too restrictive to allow reasonable responsiveness to variability in the input, without encountering unnecessary space inefficiency. The competitive attachment model avoids these potential problems by striking a compromise between the ability to create an arbitrary network structure on the fly and the commitment to a totally fixed network structure.

The strategy adopted here is to allow dynamic creation of the parsing network, but to

still be much larger than that which is needed to parse most sentences.

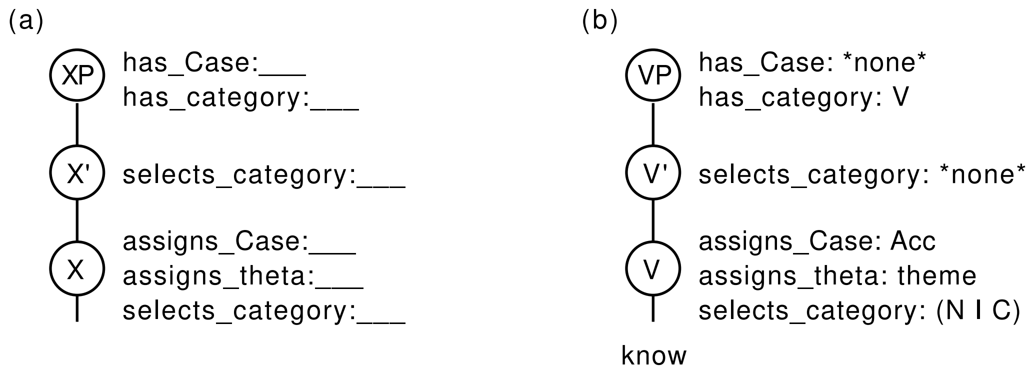


Figure 3.8: (a) Generic template for a syntactic phrase in the network. (b) Template with features instantiated based on the word *know*.

greatly restrict the structure building abilities of the parser. To create syntactic structure, the parser activates one of the syntactic phrases in a pool of generic phrasal templates, instantiates the symbolic features of the phrase according to the specific features of the current input word, and connects the instantiated phrase to the developing parse tree. There is only one kind of phrasal template, which can be used to represent any type of syntactic phrase; Figure 3.8 shows the template and a sample instantiation.[11] Thus, the size of the template pool only needs to be as large as the maximum number of phrases expected for any sentence, which is approximately equal to the number of words. This avoids the prohibitive space requirements of pure connectionist approaches.[12] On the other hand, on-line structure-building cost is kept low by limiting the parser's structural operations to the activation of existing templates and the setting of simple features within them.

The model also achieves a balance in the restrictiveness of the structures it can represent. Although the fixed templates highly constrain the form of syntactic structures, the instantiation process allows for greater flexibility than that found in approaches based entirely on a fixed network structure. In particular, the model allows important information in the input to naturally guide the parse, since the critical features of a phrase are established based on the current input. For example, the weight on the connection from a phrasal node to an attachment node is determined by the lexical features of the phrase. This allows the network to dynamically take into account the strength of expectation of a node for a sister with particular features.

[11]Other work in natural language processing based on Government-Binding theory has used similar generic phrase structure rules that are instantiated with specific features; see, for example, Dorr (1993) and Merlo (1992). The work here further explores the benefits of this type of representation within a distributed parsing approach.

[12]In the previously mentioned connectionist approaches, because of the static, *a priori* allocation of the networks, each *type of rule* must have the maximum number of copies available in the network, which means the network is far larger than needed for any given sentence.

The decision to dynamically create the parsing network raises the issue then of precisely when these generic phrasal templates are instantiated and connected to the developing parse tree. Most natural language parsers perform some degree of top-down hypothesizing of phrase structure. For example, when a parser for English encounters a determiner such as the word *the* in the input, in all likelihood it will immediately build the structure for a noun phrase, even though it has not yet seen the noun. While the presence of a determiner in English is proof of a noun to follow, there are other cases that may not be so clear. For example, although an adjective usually occurs within a noun phrase, it might not; consider *The children are happy campers* and *The children are happy.* Especially in models that consider multiple attachment possibilities in parallel, it is common for a parser to build syntactic phrases that are based on inconclusive evidence, and that may end up not being part of the final parse tree.

In a parser that consists of active syntactic processors, hypothesizing syntactic phrases in this way can greatly complicate the local communication and decision-making methods, since hypothesized phrasal nodes would have to behave differently from other phrasal nodes. More crucially, in a network that considers a large number of alternatives in parallel, activating phrases based on incomplete evidence could quickly overload the system. Thus the approach taken here of dynamically building the parsing network naturally leads to a restriction on top-down precomputation of structure. The competitive attachment parser can (and must) establish all potential attachments between existing phrases, but it can only activate the syntactic phrases themselves in response to overt, incontrovertible evidence in the input. The remainder of this section will discuss the motivations for these two design decisions—the dynamic instantiation of fixed phrasal templates, and the prohibition on top-down precomputation that follows from this assumption.

The computational reasons for building the network by instantiating uniform templates have already been presented above as the motivating factors for this approach. It was argued that this decision allows the model to achieve a balance in the space/time trade-off of structure building versus structure recognition. The approach also incorporates a lexically-driven aspect into the model that enables the parser to respond to conditions in the input in a straightforward yet flexible way.

The decision to limit the activation and instantiation of phrases to those with overt evidence in the input also has several computational justifications. It reduces the number of different types of nodes in the network and simplifies the specification of the processing algorithms they use, since hypothesized nodes would have different properties from "normal" phrases. This assumption also avoids the necessity of stipulating a cut-off point in how much structure is hypothesized. It is clear that unbounded hypothesizing of structure could lead to a network that is too large to be practical, so some bound must be determined. Activating phrases only in response to overt evidence yields the advantage of not having to establish some arbitrary bound. It also has efficiency benefits, because the number of active phrases is kept to a minimum. Furthermore, since hypothesized phrases are not activated, attachments to them cannot be represented in the network, and thus the number of attachment nodes that are established is also reduced.

Linguistic motivations for these two related design decisions were also of great importance. The use of a generic phrasal template in the parser is inspired by the lack of phrase structure rules in GB. $\overline{\text{X}}$ theory, a subsystem of GB, characterizes all phrases as having the same fixed structural shape, with differences in grammatical behavior entailed by features inherited from the lexical entry corresponding to the input. The lack of top-down precomputation in the model is also mirrored in the grammatical theory on which it is based, and is another case in which the linguistic principles map directly to the computational framework. A central notion in GB is that a syntactic phrase is *projected* from essential features of its *head*—for example, a noun is the head of a noun phrase, and its core features determine the existence of the NP, as well as its specific properties. The restriction to activating phrases only given bottom-up evidence in the input is the computational correlate of the condition of projecting a phrase from the features of its head.

## 3.4 Summary

This chapter has described the background and motivations for the three primary architectural assumptions of the competitive attachment model of parsing. Much of the justification for the design decisions stem from a desire to build a restricted computational parsing architecture—one whose computational power is limited by algorithmic simplicity and efficiency, as well as linguistic plausibility. Restrictions on the operation of the parser that are well-grounded in computational and linguistic considerations are much more likely to lead to behavior that matches human expectations in a principled manner. The remainder of the thesis will describe the properties of the model that result from these assumptions, and the behavior that emerges from them.

# Chapter 4

# Overview of the Model

This chapter provides a high-level description of the competitive attachment parsing model. Section 4.1 presents an overview of the model, explaining in detail how the design constraints from Chapter 3 are reflected in the features of the model and in the restrictions on its processing abilities. In Section 4.2, an example parse demonstrates the resulting operation of the parser. Section 4.3 concludes the chapter with an explication of the parser's critical attachment behaviors that underlie its ability to process ambiguities in a human-like manner. It will be shown that each of the crucial aspects of the model's behavior is a direct consequence of the well-motivated assumptions that form the basis for the parser's design. The intent of this chapter is to provide an understanding of the overall design of the parsing model and the key aspects of its attachment behavior. The description of the implementation of the parser in Chapters 5 and 6 will fill in more complete details of the model, and Chapter 7 will fully explicate the results of the system on example syntactic ambiguities.

## 4.1 Description of the Model

This section gives an overview of how the parser operates, describing the details of the parsing model that follow from the design constraints presented earlier. Section 4.1.1 focuses on the properties of the processing nodes and how they are structured in the network. Section 4.1.2 explains how symbolic knowledge and numeric activation are integrated within the approach. Section 4.1.3 describes the additional constraints on the structure of the network that follow from the decision to use competition-based spreading activation as the sole focusing mechanism in the model.

### 4.1.1 The Network Structure

A syntactic phrase is represented by three nodes, an X, X$'$, and XP, each of which is an independent processor; the label "X" is a generic name for a parsing node that can be instantiated as, for example, N (noun) or V (verb). Having one processor for each of the three phrasal nodes, rather than one processor for the complete phrase, encourages a modular design in which processors are specialized by phrase level. All syntactic processors perform the

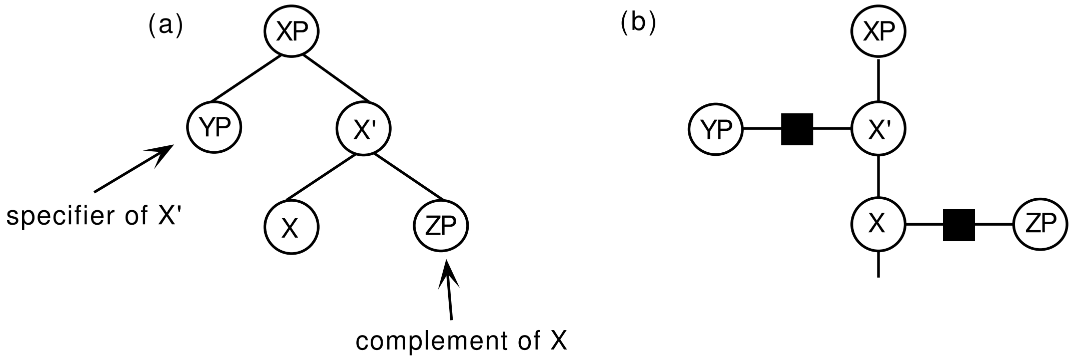


Figure 4.1: (a) Basic configuration of a phrase in $\overline{\mathrm{X}}$ theory. (b) Representation of the $\overline{\mathrm{X}}$ attachment relations within the competitive attachment model.

same algorithms, but the types of features that they store and communicate vary according to their level in the phrase. The resulting processors are simpler and less powerful than the processor for an entire phrase would have to be. For example, the processor for each node mediates exactly one attachment site—an attachment to its potential sister node.

The restriction of representing a phrase with a fixed number of nodes follows from the decision to dynamically build the network by instantiating fixed phrasal templates. The precise number of nodes is determined by considerations of $\overline{\mathrm{X}}$ theory, which specifies the properties of each level of structure within a syntactic phrase. A *complement* of a phrase, such as the object of a verb or preposition, is attached as a sister to the right of the lowest level node, the X. A *specifier* of a phrase, such as the determiner in an NP, is attached as a sister to the left of the middle level node, the X′.[1] The highest level of the phrase, the XP node, is what in turn attaches as a specifier or complement to some other phrase. These $\overline{\mathrm{X}}$ attachment relations are shown in Figure 4.1.[2] Note that in the parser, a parse tree connection between two separate $\overline{\mathrm{X}}$ phrases is encoded as a sisterhood relation.[3]

Even though they are independent processors, multiple nodes of a single syntactic phrase obviously have a close relationship to each other. The three nodes constitute a single phrasal entity and are attached to each other in any parse tree containing the phrase. In the parsing network, the three processors are thus connected to each other by direct communication links, which are distinguished from other network links. These links implicitly represent branches in the parse tree. The additional branches of the parse tree are represented explicitly by processing units called attachment nodes. There are bidirectional links between each attachment node and two phrasal nodes; these phrasal nodes are potential sisters in the

[1]The direction of attachment is language-dependent; the attachment directions given in the text are for English.

[2]In this and remaining figures, fully activated attachment nodes are shaded black.

[3]In the current implementation of the parser, only specifier and complement attachments are made. The extensions to handle adjuncts are discussed in Chapter 8.


Figure 4.2: (a) An attachment node representing a specifier relation connects to an XP level node and an X′ level node. (b) An attachment node representing a complement relation connects to an XP level node and an X node.

parse tree. One of the phrasal nodes is an XP node and the other is either an X or X′ node. Figure 4.2 shows the two types of attachment relations in the network.

There is one special kind of phrase in the model that consists of a single XP node, called an empty node. An empty node corresponds to a *trace* in Government-Binding theory. A trace is a place-holder for the logical position of a syntactic phrase that is displaced within a sentence. For example, consider the sentence:

Who$_i$ did Mary kiss $e_i$

The index $i$ on the trace $e$ indicates that the trace is a place-holder for the word *who*, which shares this index. The relationship between *who* and its trace allows the parser to determine that *who* is the logical object of *kiss*, even though it does not occur in the position normally held by that object. The trace $e_i$ *does* stand in this position, and transfers the relevant features to *who*. Since a trace obviously does not overtly appear in the input, the occurrence of an empty node in the network must be determined by other properties of a sentence. Thus empty nodes cannot be allocated the way normal phrases are. Instead, two empty nodes are activated along with the activation of each full syntactic phrase; these empty nodes act as the potential empty complement and empty specifier of the phrase. The initial configuration of a phrase with its empty nodes is shown in Figure 4.3; empty nodes are labeled with the letter "e."

### 4.1.2 Activation of Processing Nodes

The parser maintains a finite pool of phrasal nodes and attachment nodes from which the parsing network is created. Allocating processing nodes from a fixed pool requires the development of memory management techniques. A processing node that is activated and connected to the parsing network is said to be allocated; one that is inactive and available in the memory pool is said to be deallocated. Because a phrase is represented by three independent processors, these three nodes must maintain the same numeric activation level so that they can be allocated and deallocated as a complete phrase. Phrases are initially activated by the input, and this activation decays slowly over time. When the activation level of a phrase drops below a given threshold, the template becomes available for reuse. However, a phrase that participates in a recent part of the parse tree should not be re-allocated for use as another phrase. Thus, when a phrasal node activates a new attachment, its own activation (and that of its entire phrase) is boosted accordingly.

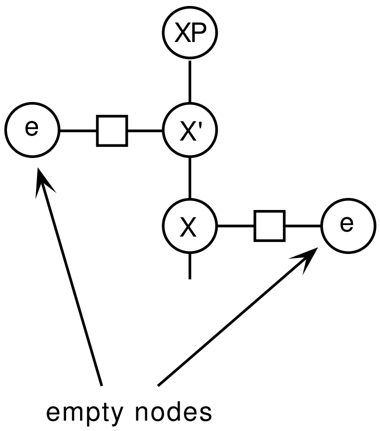


Figure 4.3: Two empty nodes are activated with each phrase, and attached in the specifier and complement positions.

An attachment node determines its level of activation by processing both the grammatical features and the numeric activation that it receives from each of its two phrasal nodes. This process is an iterative one in which information is communicated among the nodes in the network, and the activation level of each attachment node gradually becomes fixed. The resulting activation level indicates the strength of preference for an attachment between the two associated phrasal nodes to be included in the parse tree. The network is said to be in an *acceptable state* when the activation level of each attachment node is either above a certain threshold $\theta$ (the attachment node is "on"), or below some threshold $\phi$ (the attachment node is "off").[4]

There are several factors that affect the activation level of an attachment node. Although purely symbolic in nature, the grammatical features passed to an attachment node play an important role in its activation function. These features specify relevant linguistic information such as "the category of the XP is noun" and "the X node expects an XP sister of category verb." There are universal constraints embodied by each attachment node that determine the effect of the combination of these symbolic features on the activation level of the attachment node. Every attachment node applies the same constraint algorithm to its feature values; this algorithm is a computational encoding of certain central grammatical constraints in Government-Binding theory. For example, some of the relevant constraints are the following:

- **Selection**: The category of an XP must match the category expected by an X or X$'$.
- **Case Filter**: An NP in certain syntactic configurations must be assigned *Case*, which

[4]This property makes the management of the pool of attachment nodes quite simple: Attachment nodes that are on self-reinforce over time, since they explicitly represent the structure of the parse tree, while those that are off directly return to the pool of available attachment nodes.

is an abstract grammatical feature such as Nominative or Accusative.[5]

- **Theta Criterion**: An XP in certain syntactic configurations must be assigned a *theta role*, which is an abstract grammatical feature indicating the role (for example, agent or theme) of the phrase with respect to some predicate.[6]

It should be emphasized that these constraints are applied at an attachment node. That is, the constraints do not apply directly to an XP node, but rather to an XP node *in a certain attachment relation* to another node.

The effect of the constraint algorithm is to increase the potential activation level of an attachment node in proportion to the degree to which the given constraints are satisfied. Furthermore, a general computational restriction states that two different values specified for the same feature must be compatible; if this restriction is violated, the attachment node is invalid and set its activation level to 0. For example, an attachment node might receive a feature from its X node stating that it expects an XP sister of category verb, and a feature from its XP node stating that its category is noun. Since these two features specify two different values for the category of the XP node, the attachment node would immediately become inactive.

In addition to the effect of symbolic constraint satisfaction, the activation level of an attachment node depends directly on the numeric input from its associated phrasal nodes. In order for an attachment node to become fully active, it must receive the entire output from each of the two phrasal nodes. In fact, for the network to represent a consistent parse tree, the two phrasal nodes connected to each attachment node must "agree" either to turn the attachment node on, by both sending it all of their output, or to turn it off, by both sending it none of their output. Each of the phrasal nodes determines its output to an attachment node using a competition-based spreading activation function. A phrasal node may be connected to a number of attachment nodes, and it divides its output activation among them, proportional to their current activation level.[7] When the iterative processing of information in the network brings about an acceptable state, each phrasal node will have chosen a single attachment node to activate.[8]

---

[5]Case in Government-Binding theory is a technical term that refers to an abstract feature that is sometimes, but not always, visible in the phonetic form of a word. For example, the word *he* has Nominative Case, while *him* has Accusative Case. However, Case in English is not generally reflected in the form of a noun.

[6]Those readers familiar with GB will recognize that this is only "half" of the Theta Criterion. The other part of the Theta Criterion states that a predicate assigns each of its theta roles to exactly one phrase. This is accomplished indirectly in the parser through constraints on the number of attachments to a node.

[7]The phrasal node's output function also takes into account the weights on the links to the attachment nodes, as in the CBSA function shown in equation 3.1 on page 21. The role of weights in the parser will be discussed in more detail in Chapter 7; for now all weights will be assumed to be 1.0.

[8]Empty nodes are an exception. Each empty node connects to a single attachment node that may or may not become activated. If the attachment node turns off, the empty node is deactivated as well, since it has lost its connection to the rest of the parse tree.

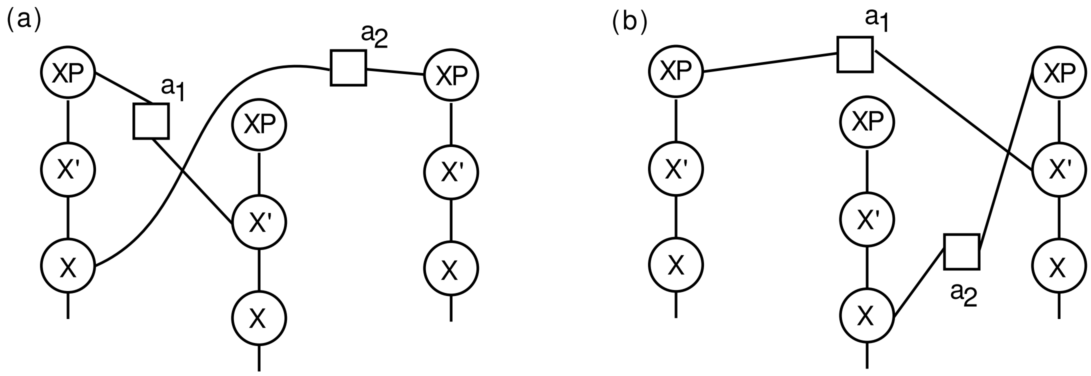

Figure 4.4: Example pairs of incompatible attachments that CBSA alone cannot prevent from being active simultaneously.

### 4.1.3 The Stack

The proportional allocation of output discussed above brings about competitive behavior between attachment nodes, focusing the activation in the network and helping it to converge on a consistent parse tree solution. It turns out, however, that if attachment nodes are created between *all* possible pairs of appropriate phrasal nodes (that is, all combinations of XP and X nodes, and XP and X′ nodes), then the use of CBSA is not sufficient to prevent the simultaneous activation of some incompatible attachment nodes. Figure 4.4 shows the types of structures in which CBSA is an insufficient competitive mechanism; both structural configurations involve violations of the proper nesting structure of a parse tree. The problems arise from the fact that, for CBSA to bring about competition between two nodes, they must be connected to a common node on which they both depend for input activation. In each of the subnetworks of Figure 4.4, attachment nodes $a_1$ and $a_2$ have no such phrasal node that outputs activation to both of them, so there is no mechanism by which to bring about their competition.

For this reason, additional steps must be taken in structuring the network to ensure that the parse tree represented is a valid one. One possibility would be to retract the model's prohibition on the use of direct inhibition. Then inhibitory links could be created between any two attachment nodes whose simultaneous activation would lead to an invalid parse tree. The computational and conceptual problems of this direct inhibition approach were discussed in Section 3.2. These problems, which were the motivation for prohibiting inhibitory links in the model, arise from the unrestrictiveness of the approach. So the preferred technique here is to find a way to restrict the model more, not less, in order to solve the problems exemplified in Figure 4.4.[9]

[9]In an earlier version of the model, an attempt was made to investigate the feasibility of using inhibitory links to ensure the validity of the trees created by the parser. The resulting network was large and quite structurally complex, due to the unrestricted number of attachment nodes and the complicated inhibitory

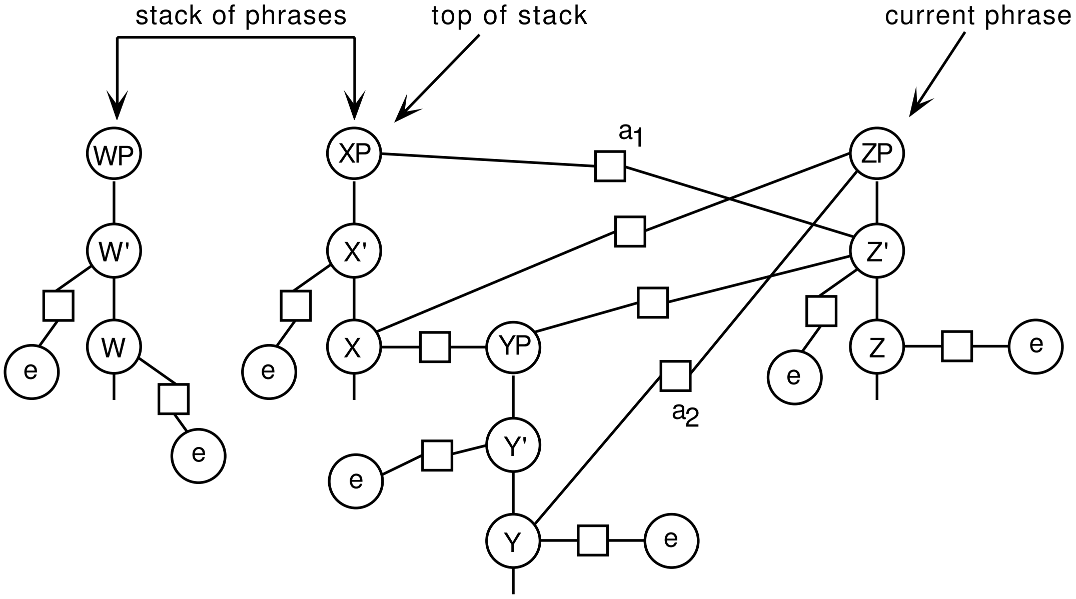


Figure 4.5: The stack of partial parse trees, and the current input phrase. The current phrase may establish attachments only to the right edge of the tree on top of the stack.

The solution adopted in the competitive attachment model is to impose structure on the network that ensures that the use of CBSA is sufficient to rule out any incompatible attachment nodes. An obvious structuring mechanism, used in more traditional parsing frameworks, is to maintain a stack of partial parse trees during the parse. Introducing a stack mechanism into the competitive attachment model greatly restricts the creation of potential attachments. The parser allows new potential attachments to be established only between the current input phrase and the right edge of the partial parse tree on the top of the stack, as shown in Figure 4.5. (Details of the parser's stack mechanism will be presented shortly.) Even given this restriction, it is clear that there are incompatible attachment nodes that do not compete through CBSA. For example, attachment nodes $a_1$ and $a_2$ in Figure 4.5 correspond to the incompatible attachment nodes $a_1$ and $a_2$ in Figure 4.4(b), and their simultaneous activation is still not directly ruled out by CBSA. Surprisingly, however, the restrictions on connections in the network now guarantee that incompatible attachments such as these cannot be simultaneously active. Each non-empty phrasal node must activate a single attachment, and a pair of phrasal nodes must agree to turn the attachment node between them on or off. These local decisions have consequences that propagate through the constrained connectivity of the network, and ensure a global solution in which the set of active attachment nodes forms a valid parse tree.

In order to see how this propagation of the results of local competitions works, some

patterns that had to be established among them. This level of complexity within a connectionist network significantly decreases the chances of developing simple spreading activation methods that allow it to converge.

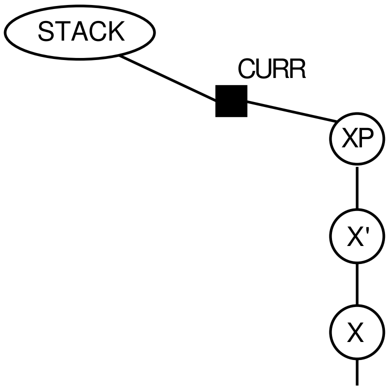


Figure 4.6: A syntactic phrase pushes itself onto the stack when its XP node activates the CURR attachment node connected to the stack node.

important properties of the stack must first be understood. The stack data structure is maintained through a single stack node in the network. It should be emphasized that this stack node is *not* a global control mechanism, but a simple network processing node. Like phrasal nodes, the stack node has connections to attachment nodes that it must decide to turn on or off. When a syntactic phrase is initially activated by some input word, it may either attach to the developing parse tree, or it may push itself onto the stack, in order to structure itself with later input. In the current implementation of the parser, all syntactic nodes—XP, X′, or X—attempt to activate exactly one attachment node. The way a phrase pushes itself onto the stack is for the XP node of the phrase to activate an attachment node between it and the stack node. This relationship is shown in Figure 4.6.[10] The attachment node connecting the stack node to the current input phrase will be referred to as CURR.

The stack node may have connections to additional attachment nodes as well; these attachments represent the result of previous push operations. There is thus a list of the attachment nodes that connect to pushed phrases, with the attachment node representing the most recent push operation referred to as TOS (top of stack), and the rest of the list referred to as REST.[11] Figure 4.7 shows the stack and its potential attachment node connections. The stack clearly must be able to simultaneously activate more than one attachment node in order to maintain the attachments to all of the pushed phrases. The activation of nodes on the REST list is straightforward because the phrases that these attachment nodes are connected to are not participating in current attachment decisions. The stack must maintain their status on the REST list simply by sending them constant activation.

The stack's potential activation of CURR and TOS is more complicated. The current

[10]Recall that fully activated attachment nodes are shaded black.

[11]The length of the REST list—and hence the depth of the stack—is not currently restricted; a plausible model would have to impose some restriction on the size of the stack. However, in the range of English examples tested, the REST list never had more than one element.

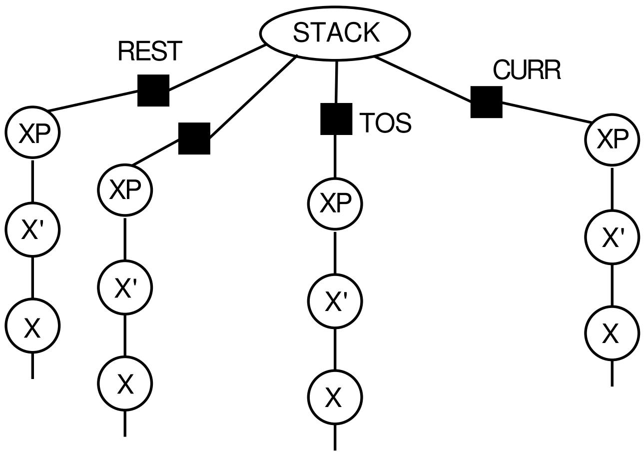


Figure 4.7: The stack node and its attachment nodes, which are used to maintain the stack structure.

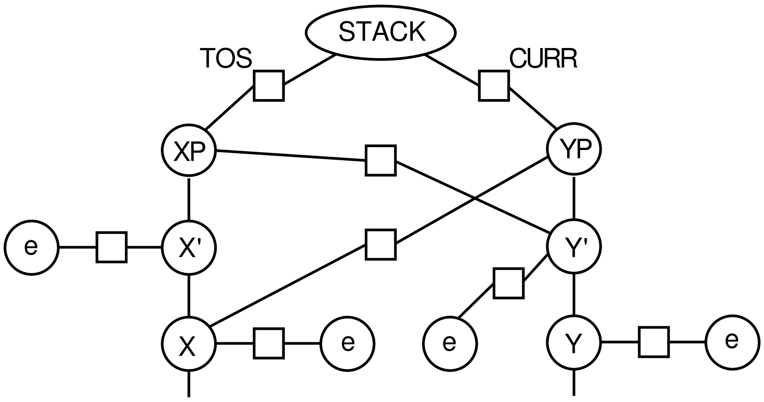


Figure 4.8: Potential attachments between the current phrase (CURR) and the phrase on the top of the stack (TOS).

input phrase may push itself onto the stack by activating CURR, or it may activate attachments to the partial parse tree on the top of the stack, which also affects the status of the CURR and TOS nodes. Consider the situation depicted in Figure 4.8, where a phrase of category X is on the top of the stack and a phrase of category Y is the current input. There are three possible attachment outcomes between these two phrases:

1. As shown in Figure 4.9, the YP may attach as the complement of the X node, while the XP remains on the top of the stack. In this case, the TOS attachment node remains active and the CURR node is inactive.

2. Alternatively, Figure 4.10 shows that the XP node may attach as the specifier of the Y′, entailing that the X phrase has popped itself from the stack. In this case, the TOS

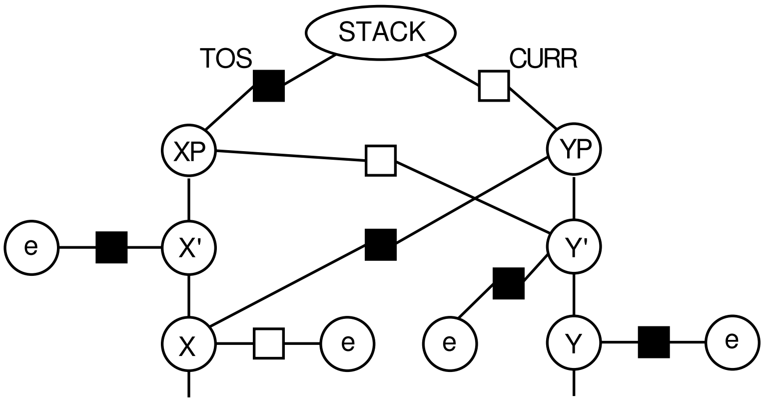


Figure 4.9: The CURR phrase attaches as the complement of the TOS phrase.

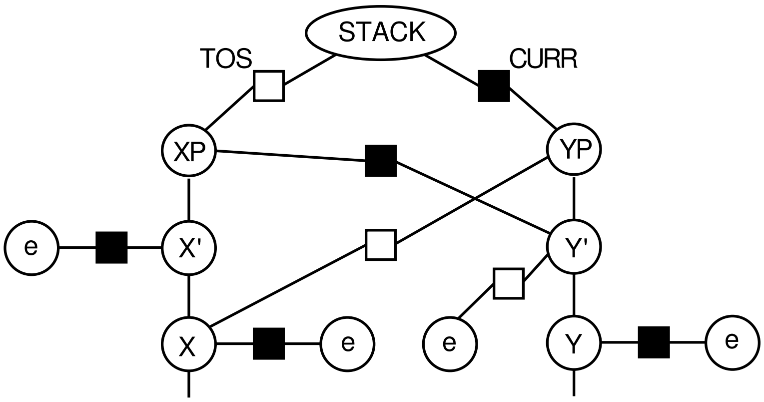


Figure 4.10: The TOS phrase attaches as the specifier of the CURR phrase.

attachment node becomes inactive and the CURR node becomes active.

3. Finally, there may be no valid attachments between the X phrase and the Y phrase, and the current phrase must take the default action of pushing itself onto the stack. In this case, both TOS and CURR are simultaneously activated, as in Figure 4.11.

In order to accommodate each of these attachment possibilities, the stack uses a CBSA function to allocate activation to TOS and CURR. But this output function, in contrast to that of the phrasal nodes, allows for multiple winners. So while at least one of TOS and CURR must receive output from the stack, both may receive output simultaneously, as needed for case (3) (shown in Figure 4.11).

Now that the basic stack mechanism has been presented, let us return to the example in Figure 4.5, repeated here as Figure 4.12, in which the stack attachment relationships are explicitly shown. (Attachments $a_2$ and $a_6$ correspond to the incompatible attachments $a_1$ and $a_2$ in Figure 4.5.) It is now possible to show how the propagation of local attachment

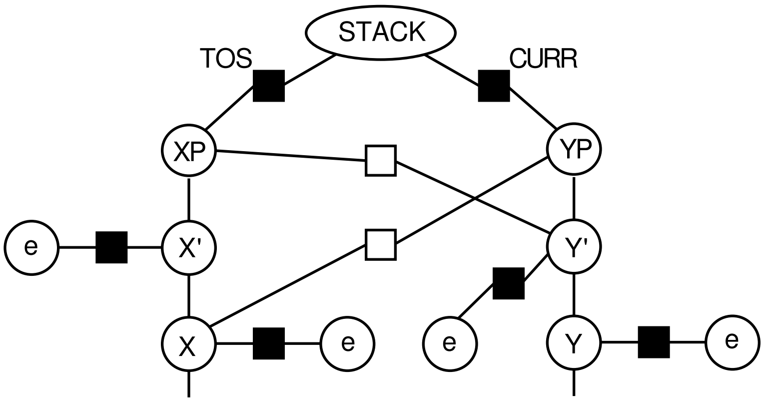


Figure 4.11: No valid attachments between the CURR and TOS phrases.

decisions ensures a globally consistent set of active attachment nodes. Although this example will present the attachment decisions as a series of discrete choices, the actual computation of these decisions in the parsing network is a process in which the nodes gradually adjust their activation levels over time. Thus, this is not a trace of how the network arrives at these decisions, but rather a determination of what the logical possibilities are for the simultaneous activation of attachment nodes within the network. In particular, let us consider the case of attachment nodes $a_2$ and $a_6$ being active simultaneously.[12] Assume that $a_2$ is active. This means that the XP node must be sending all of its output to $a_2$, and thus $a_1$ cannot be on. Since the stack must activate at least one of $a_1$ and $a_8$, then $a_8$ must be on. We now have the situation as demonstrated in Figure 4.13. Since the ZP node can activate only a single attachment and $a_8$ is on, both $a_3$ and $a_6$ must be off. Therefore $a_2$ and $a_6$ cannot both be active; the network now looks like Figure 4.14. To complete the example, it is easy to see that attachments $a_4$, $a_7$, and $a_{10}$ must be on, while $a_5$ and $a_9$ must be off. The final network is shown in Figure 4.15.

One additional constraint is necessary to ensure that the network represents a valid parse tree. Recall that when a syntactic phrase is activated, it may either attach to the tree on the top of the stack, or it may push itself onto the stack. At this point, before another phrase is processed, all "losing" attachment nodes (those which are off) must be deallocated. This prevents the inactive attachment nodes from being re-activated later and, at that point, invalidating the tree structure. Thus when each input phrase is activated, the network is guaranteed to be in a state in which the addition of new attachments will maintain a valid parse tree representation.

[12]This is just one example; the same process of propagating competitive relations ensures that any other pair of incompatible attachment nodes also cannot be turned on simultaneously.

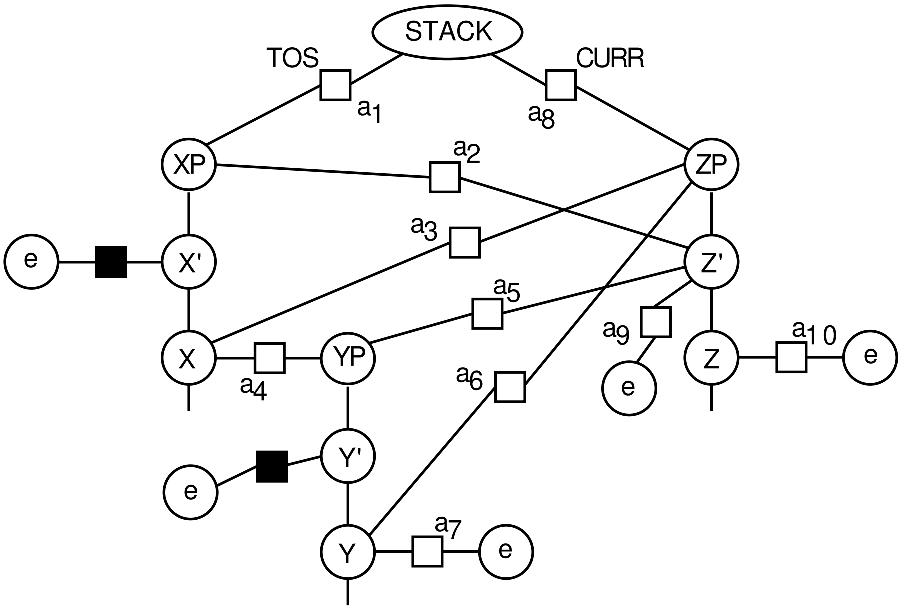


Figure 4.12: The set of potential attachments between CURR and TOS includes attachments, such as $a_2$ and $a_6$, that are incompatible but do not compete through CBSA.

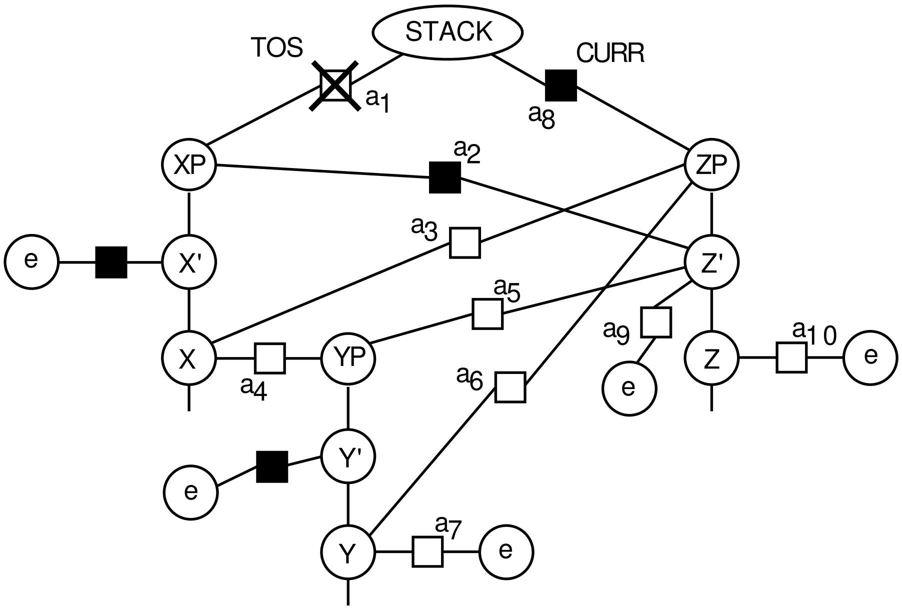


Figure 4.13: Propagation of local competitive decisions. If $a_2$ is on, then $a_1$ must be off. If $a_1$ is off, then $a_8$ must be on.

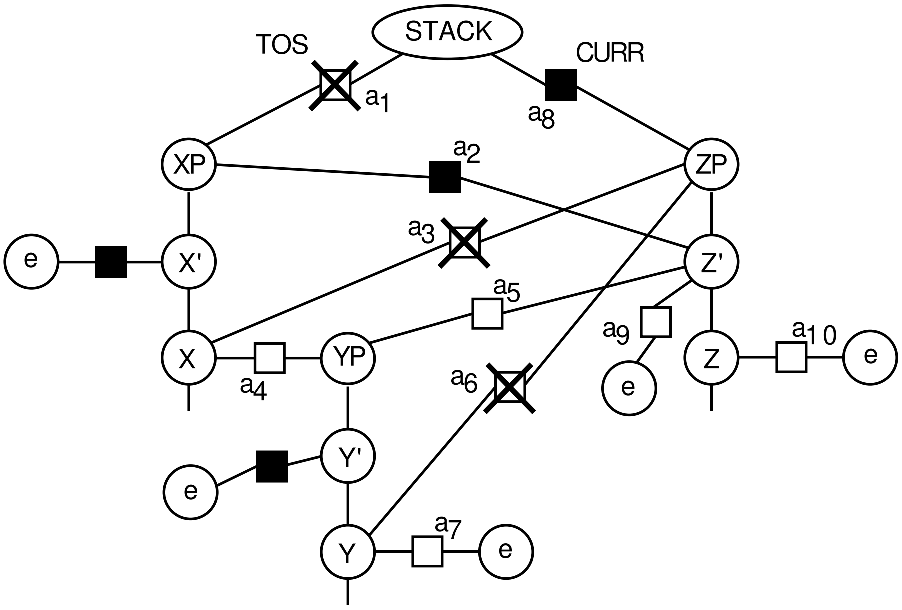


Figure 4.14: Propagation of local competitive decisions. If $a_8$ is on, then $a_3$ and $a_6$ must be off.

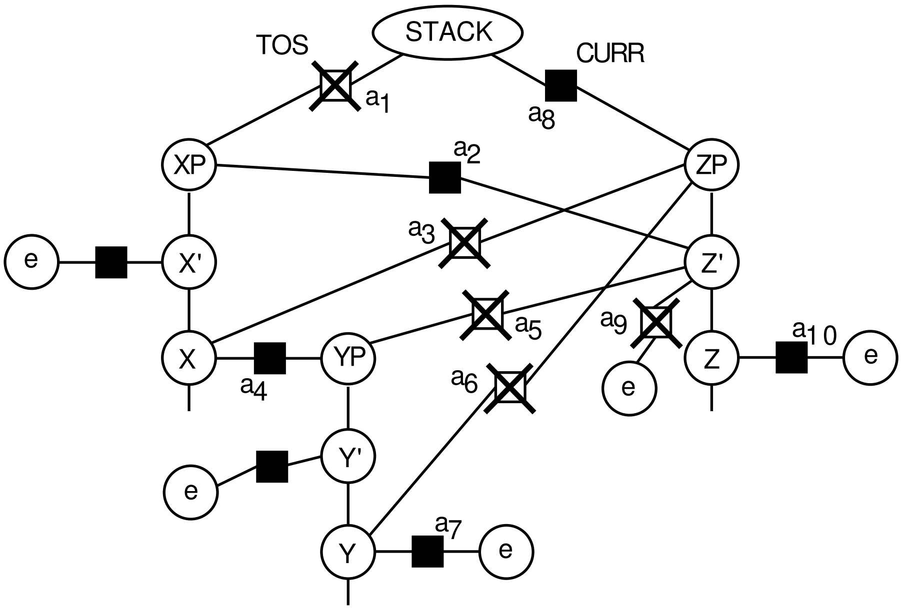


Figure 4.15: Propagation of local competitive decisions. Final results of assuming that $a_2$ becomes active: $a_2$, $a_4$, $a_7$, $a_8$, and $a_{10}$ are on; $a_1$, $a_3$, $a_5$, $a_6$, and $a_9$ are off.

## 4.2 A Parsing Example

The previous section provided an overview of the parser's structuring operations, and the symbolic and numeric processing that takes place in the resulting network. This section provides a high-level description of how the model parses an example sentence, *The woman knows Sara ran*. Details of the symbolic information underlying the attachment decisions of the parser, and the numeric processing that carries out the decision making, will be deferred to the following two chapters. The intent here is to give a higher-level view of how the parser proceeds step by step through the input, gradually building up a tree structure representation of the sentence by activating the appropriate attachment nodes. However, it should be noted that, although the description is at a high level, the behavior described here is precisely that of the implemented parser.[13]

The input to the parser is a list of tokens corresponding to the words of the sentence. For simplicity, a verb is specified by two tokens, one representing the tense features of the verb (for example, *present* or *past*), and one the root of the verb (for example, *know* or *run*).[14] This step is taken because tense features such as *present* are the "overt evidence" the parser requires to build a sentential clause; the prior analysis of the verb makes the parser's task easier by extracting these features ahead of time. The input token list for the example sentence is thus "(the woman present know Sara past run)."

When an input token is processed, the preprocessor of the parser looks up the word in the lexicon, then activates a phrasal template and instantiates it with the features from the lexical entry.[15] For example, the instantiation of a phrasal template in response to the word *know* is shown in Figure 4.16. Along with the activation of each phrase, two empty nodes are also activated; one of the empty nodes is connected with an attachment node to the X node, and one is connected with an attachment node to the X′ node.[16] (For this example, the reader may assume that an attachment to an empty node acts as the default attachment for an X or X′ node. If a non-empty XP node attaches to the site instead, then both the empty node and its associated attachment node are deallocated.) An attachment node is also set up between the XP node of the new phrase and the stack; this is the stack's CURR attachment node. Next, all the potential specifier and complement attachment nodes are established between the current phrase and the right edge of the tree attached to the stack's TOS node, as was shown in Figure 4.5 on page 37.[17] After the new input phrase is thus

[13]The parser is implemented in Allegro Common Lisp, using the CLOS object oriented package. The parser in actuality runs serially, simulating the parallel processing of the network.

[14]Note that this is done solely to simplify the implementation, and is not intended to be an implicit claim that such a preprocessing step is morphologically plausible.

[15]The parser is simplified by having a preprocessor perform certain tasks that are difficult to achieve in a strict connectionist framework, and whose behavior is essentially irrelevant to the ambiguity processing mechanisms of interest here.

[16]The initial configuration of a phrase with its empty nodes was shown in Figure 4.3 on page 34.

[17]Although at present this task is also performed by the preprocessor, it is easily accomplished in a

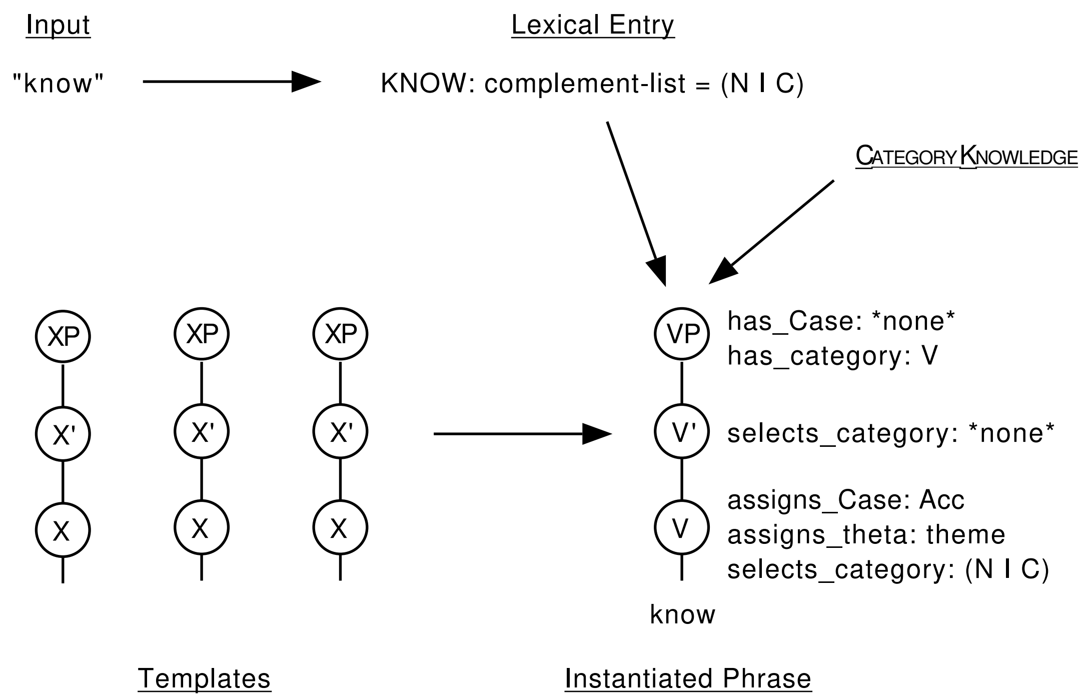


Figure 4.16: The lexical entry corresponding to an input word, along with associated category information, is used to instantiate a newly activated phrasal template.

connected to the stack and the developing parse tree, the iterative symbolic and numeric processing of the resulting network is triggered. The processing loop halts when the network settles on an acceptable state.[18] At this point, the losing attachment nodes are deallocated, and the stack's pointers are appropriately revised. The parser is then ready to process the next input token.

Let's turn now to the actions of the parser given the input "(the woman present know Sara past run)." The initial state of the parser is shown in Figure 4.17. The stack node has no connections to attachment nodes, so its CURR and TOS pointers are nil.[19] Figure 4.18 shows the network after the first token, *the*, is processed, and a determiner phrase DtP is

---

distributed fashion given the feature-passing capabilities of the parsing network.

[18]Recall that an acceptable state is one in which each attachment node is either fully activated or is inactive. In some rare cases discussed in Chapter 5, the network does not achieve such a state, and the processing loop is halted when the number of iterations surpasses some constant. In this case, the network is in a state in which at least one phrasal node has not made a clear choice between its attachment possibilities. This situation did not occur in any of the simulations on actual linguistic input; see Chapter 5 for further discussion.

[19]Since the stack's REST list is not needed in this example, it will not be depicted in the figures. The value of REST is nil throughout the parse.

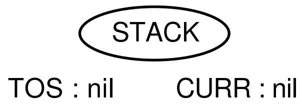


Figure 4.17: The initial state of the parser has a single stack node with no phrasal or attachment nodes activated yet.

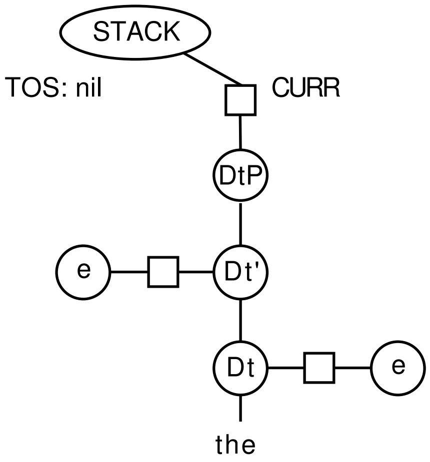


Figure 4.18: When *the* is processed, a determiner phrase is allocated and connected to the stack. (The phrasal nodes of a determiner phrase are labeled DtP, Dt′, and Dt.)

allocated for it.[20] Since there is no phrase already on the stack (its TOS pointer is nil), there are no other potential attachments to be created. Note that this means that each of the three attachment nodes shown in Figure 4.18 have no attachments with which they compete. At this point, the parser initiates the symbolic and numeric processing within the network. When the network settles, the three existing attachment nodes are active. The CURR node being active means that the current input phrase has pushed itself onto the stack; it now becomes the top of the stack, as shown in Figure 4.19.

The parser now activates a phrase for *woman*, and connects it to the stack through the CURR attachment node. Because there is a phrase on the stack, all potential attachments between it and the current phrase must be established. These are shown in Figure 4.20; there is a potential specifier attachment between the DtP node and the N′, and a potential complement attachment between the Dt node and the NP. Now there *are* sets of competing attachments: $a_1$ and $a_2$; $a_2$ and $a_7$; $a_4$ and $a_5$; $a_4$ and $a_6$. Once again the network's iterative

[20]A determiner does not actually trigger a full phrasal structure; the full structure is shown in this example just to make the determiner phrase the same as other phrases. This does not affect the attachment decisions made by the network.

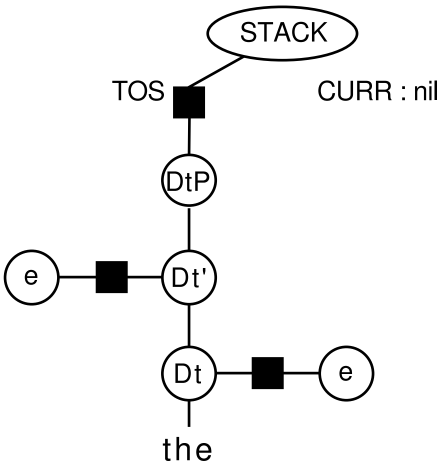


Figure 4.19: The DtP has pushed itself onto the stack.

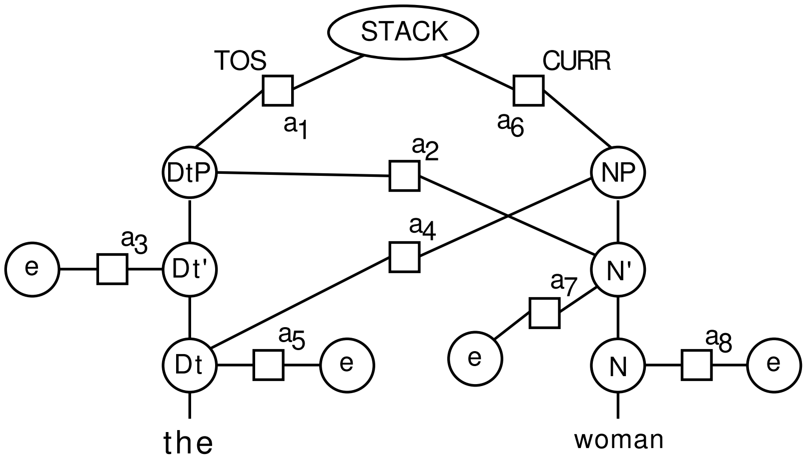


Figure 4.20: The parser allocates an NP for *woman*, and connects it to the stack and the TOS phrase.

processing is initiated; when it settles, the DtP has attached as the specifier of the NP.[21] Figure 4.21 shows which attachment nodes are now active. The DtP has popped itself from the stack, and the NP has pushed itself onto the stack. Once the losing attachment nodes are deallocated and the stack's pointers updated, the state of the parser is as shown in Figure 4.22.

The parser now reads the next input token, *present*. The lexical entry for *present* triggers

[21]Determiners of a noun occupy the specifier position in the noun phrase. There is nothing in the parser that prohibits the alternative DP analysis of the determiner/NP relationship, which is a common approach in GB theories of phrase structure (Abney, 1986; Speas, 1990).

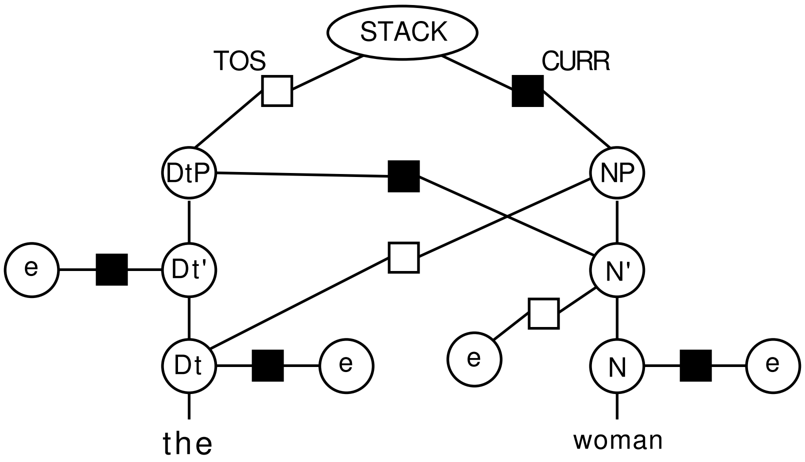


Figure 4.21: The attachment of the DtP to the N′ is active after the network settles.

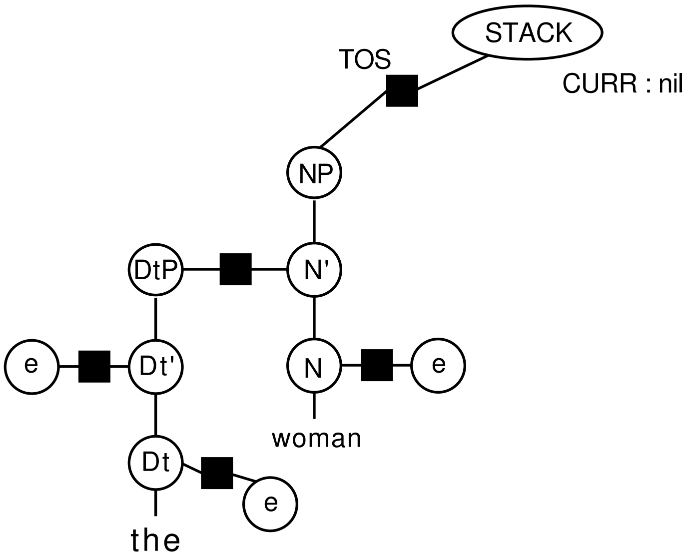


Figure 4.22: State of the parser after the losing attachment nodes are deleted and the stack pointers are updated.

the instantiation of a sentential phrase, which is known as an IP (inflection phrase) in $\overline{\text{X}}$ theory.[22] The IP is connected to the stack, and all logical attachment possibilities between it

[22] A full sentential phrase in $\overline{\text{X}}$ theory is actually a CP (complementizer phrase), of which the IP is a subtree. The parser *does* build a full CP and IP pair of phrases for tensed clauses, but this example is simplified to only display the IP. No attachment decisions are altered by this simplification. Chapter 7 presents examples with full sentential structure.

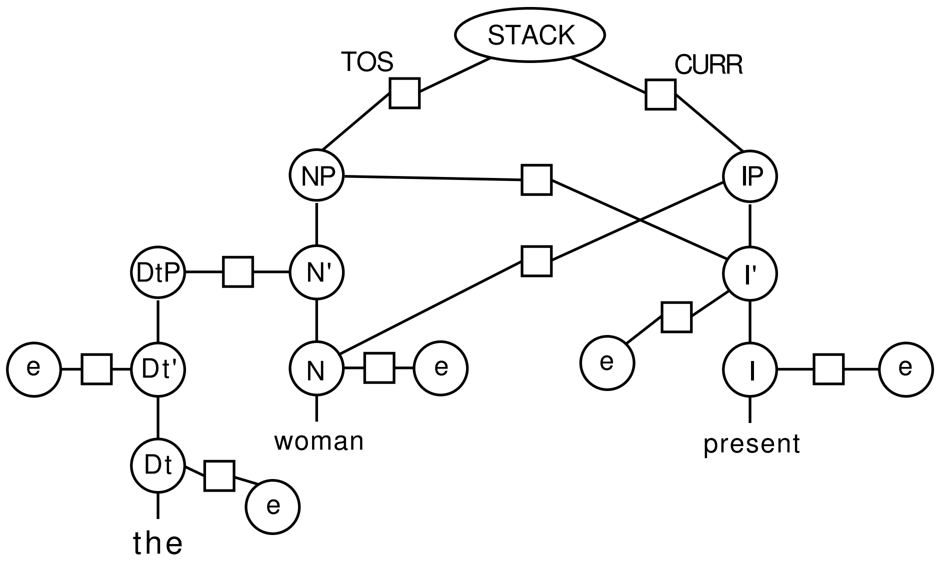


Figure 4.23: Attachment nodes established between the NP on the stack and the new IP.

and the NP on the stack are established; see Figure 4.23. Note that no attachment nodes are set up between the determiner phrase and the current input. Since the determiner phrase is not along the right edge of the tree on the top of the stack, it is unavailable for further attachments. The parser again triggers the iterative processing of the network. In $\overline{\mathrm{X}}$ theory, the subject of a sentence is represented as the specifier of the inflection phrase. Thus when the network settles, the NP has attached to the I′, and the IP has pushed itself on the stack. Figure 4.24 shows the network after the inactive attachment nodes are removed and the stack's pointers are updated.

The parser next activates a VP and connects this phrase as before to the stack. Once again, potential specifier and complement attachment nodes are established between the current phrase (the VP) and the phrase on top of the stack (the IP). The VP of a sentence attaches as the complement of the inflection phrase; the active attachments after the network settles are shown in Figure 4.25. This is the first time in the parse that the current phrase has not pushed itself onto the stack. Since the stack must activate at least one of TOS and CURR, the IP remains on the top of the stack.

The parser now processes the input token *Sara*, and connects its phrase to the existing network. Figure 4.26 shows that the parser establishes attachment nodes between the current phrase and the entire right edge of the tree on top of the stack. The NP has a valid attachment possibility as the complement of the V, and so it makes this attachment; the outcome is shown in Figure 4.27. However, this attachment cannot be part of the final parse of the sentence, in which the clause *Sara ran* is the complement of the V. When the parser processes the next token, *past*, it allocates the IP phrase that represents this clause. The new attachment nodes are shown in Figure 4.28. The appropriate parse of the sentence can be achieved by

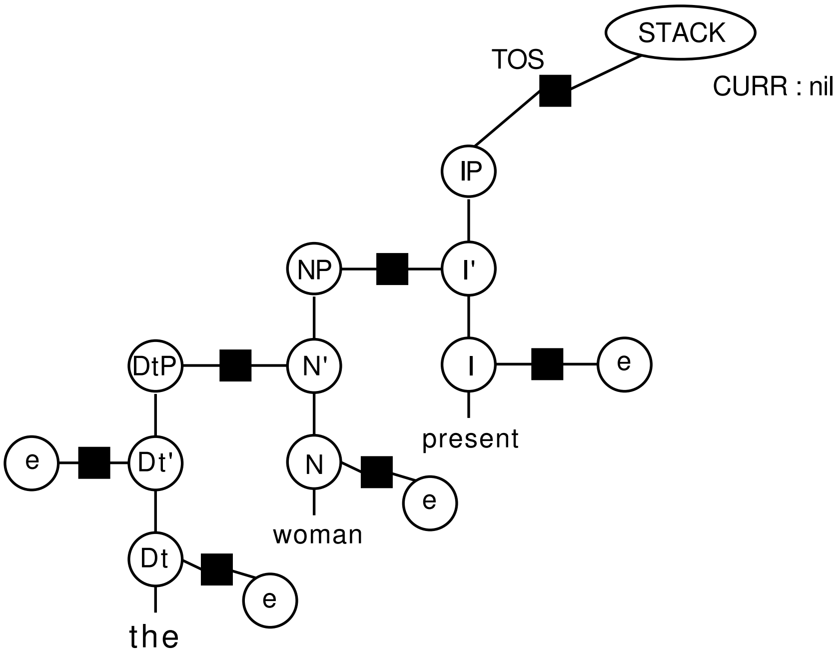


Figure 4.24: State of the parser after the losing attachment nodes are deleted and the stack pointers are updated.

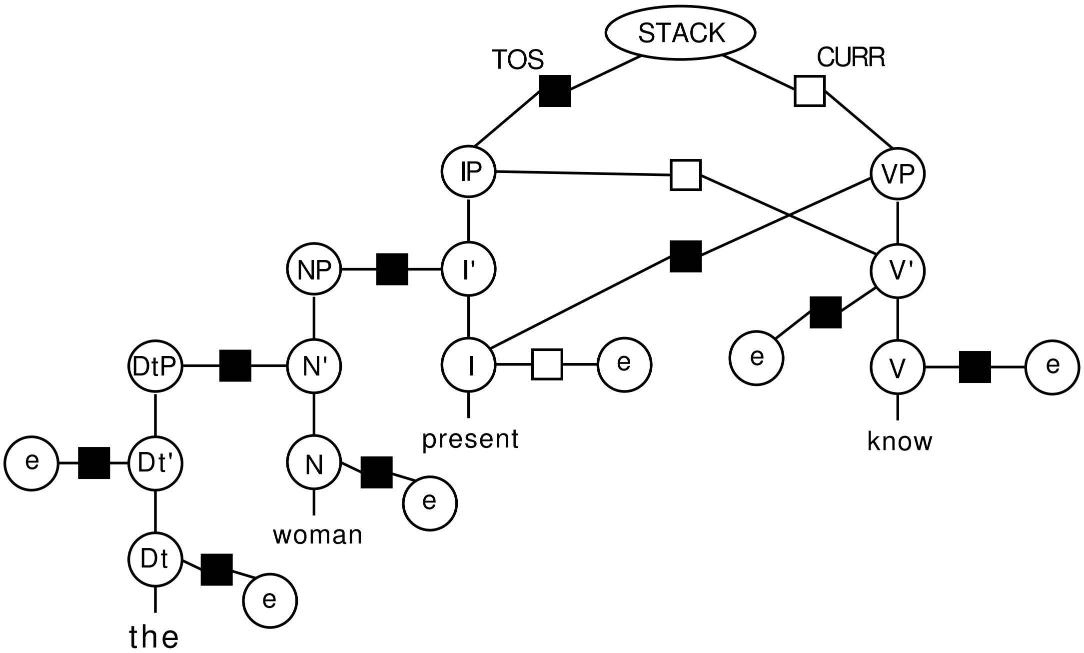


Figure 4.25: Active attachments after the network settles on the VP to I attachment.

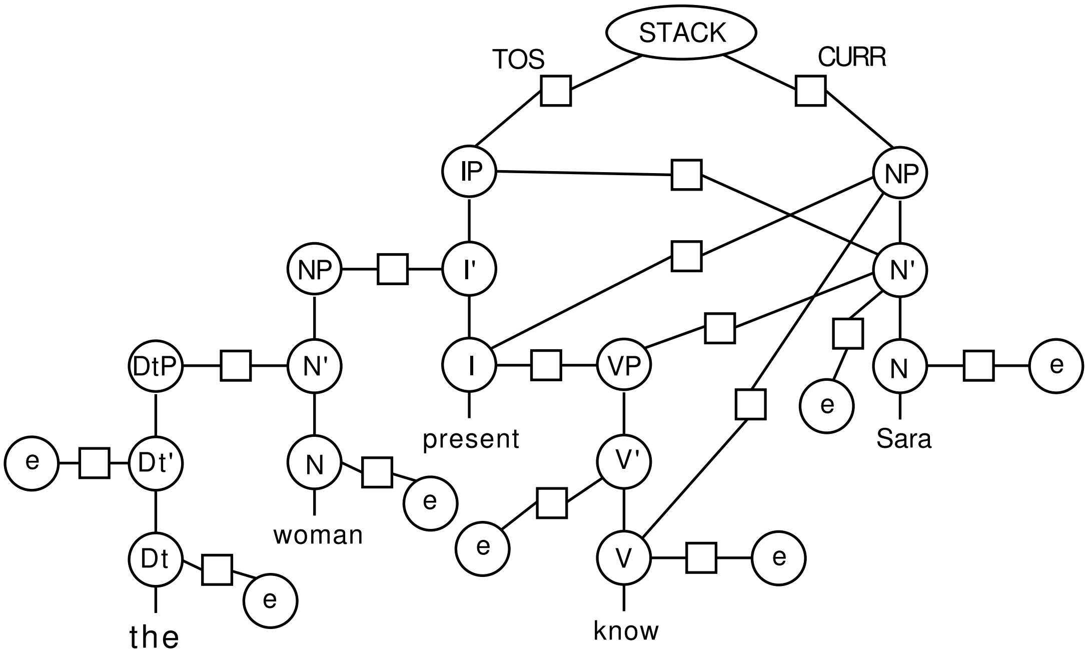


Figure 4.26: All potential attachments between the current NP and the phrase on top of the stack.

turning off the attachment of the NP to the V, and turning on the attachments of the IP to the V and the NP to the I′. The network settles on this solution, as shown in Figure 4.29.

The parser processes the final input token, *run*, and attaches the current VP to the most recent I node. Figure 4.30 depicts the final state of the parsing network. A parse is successful when, as in this case, both CURR and REST are nil, TOS is non-nil, and all phrasal nodes are connected to exactly one active attachment node. The TOS attachment node then points to the root of the parse tree for the input sentence. The parse tree represented by this network is shown in Figure 4.31.

## 4.3 Critical Attachment Behaviors

This section concludes the chapter with a discussion of the key aspects of the behavior of the competitive attachment model, which are a direct result of the design decisions presented in Chapter 3. Recall the three fundamental computational assumptions of the model: (1) The basic architecture is that of a hybrid connectionist network integrating symbolic and numeric computation. (2) Numeric decision-making is focused through competition-based spreading activation (CBSA), and the parser uses no inhibitory links. (3) The network is established through dynamic instantiation of generic template nodes, and top-down hypothesizing of structure is prohibited. The critical attachment behaviors of the parser will be presented

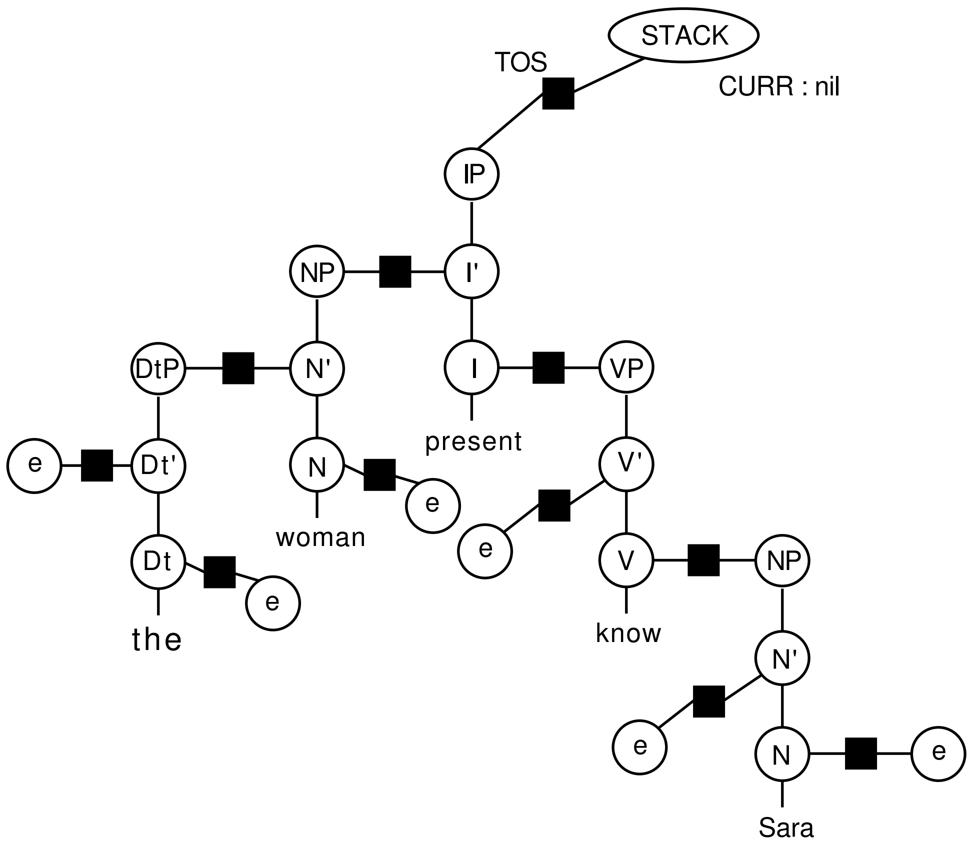


Figure 4.27: State of parser after the losing attachment nodes are deleted and the stack pointers are updated.

here in terms of the design assumptions from which they arise, referring to the example in Section 4.2 to illustrate the properties that are discussed.

The distributed network architecture gives rise to the parser's basic style of competitive attachment behavior, which follows from attachments being active, independent processing units. All potential attachment nodes that have been established are active simultaneously, and many of these attachments may compete with each other. Because there is no global decision-making mechanism in the model, the syntactic phrases must connect up with each other to form a network of communicating phrases—a phrase that is unconnected to the rest of the network can have no effect on the parse. Thus, if an attachment node is valid—that is, if its two phrasal nodes communicate compatible feature values to it—then the node actively competes for activation and tries to turn itself on. It is imperative that attachment nodes do so, since this establishes the connections in the parsing network. It is also crucial that the stack's ability to activate attachment nodes be somewhat weaker than that of phrasal nodes. Since the attachment to the stack is always a valid option for a new phrase, if that attachment node (CURR) were activated as strongly as parse tree attachments, processing

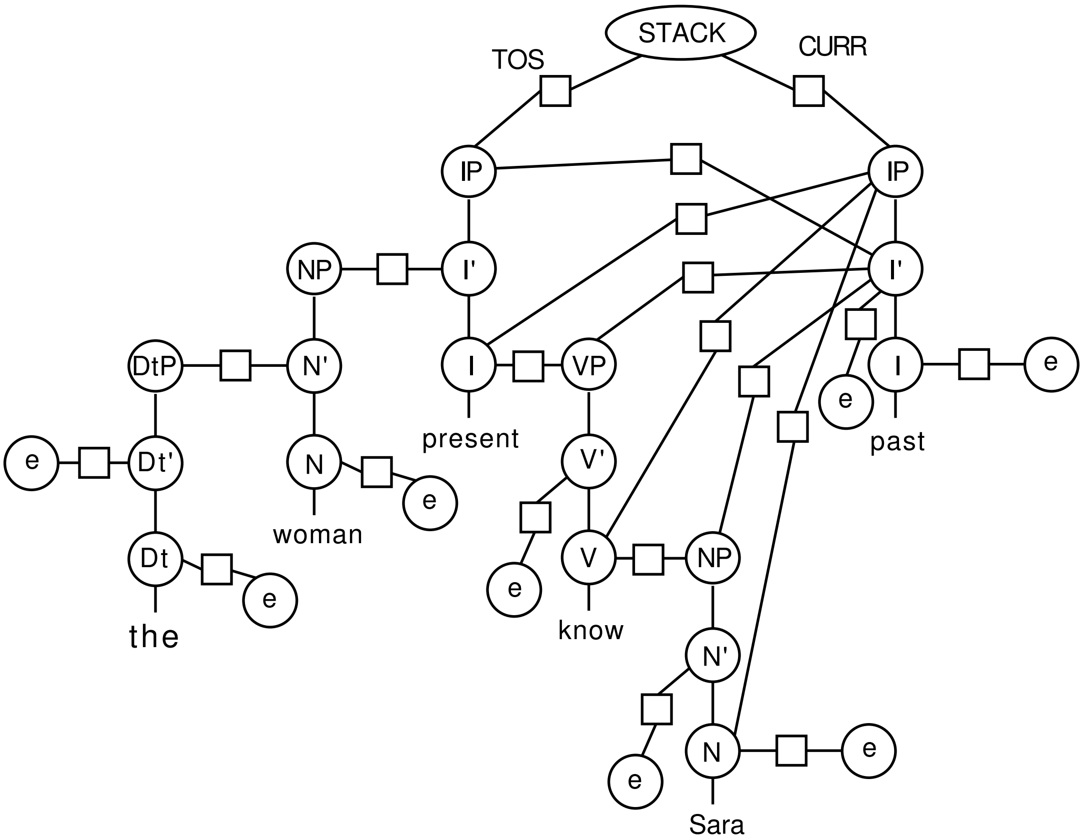


Figure 4.28: All potential attachments between the current IP and the phrase on top of the stack.

a sentence could consist of pushing each of the input phrases onto the stack and never attaching them to each other within a parse tree structure. The result of these competitive attachment behaviors is that phrases try to quickly attach themselves into an unambiguous syntactic structure; Chapter 7 will illustrate how this behavior leads to preferred resolutions of ambiguities that match human preferences.

The hybrid nature of the network is also crucial to its attachment behavior. Attachment decisions depend entirely on local, distributed decision-making among the stack, phrasal, and attachment nodes. Grammaticality is determined by symbolic feature-passing through the parsing network. Deciding between grammatical alternatives is made possible by the use of spreading activation to gradually settle on a globally consistent and preferred solution. For example, consider the point in the example of Section 4.2 at which the parser revises its initial attachment of the NP; see Figure 4.28 on page 53. There is more than one set of grammatically valid attachments that may logically be activated. The NP may maintain its current attachment to the V, while the IP pushes itself on the stack; alternatively, the NP may revise its attachment and become the specifier of the IP, which in turn replaces

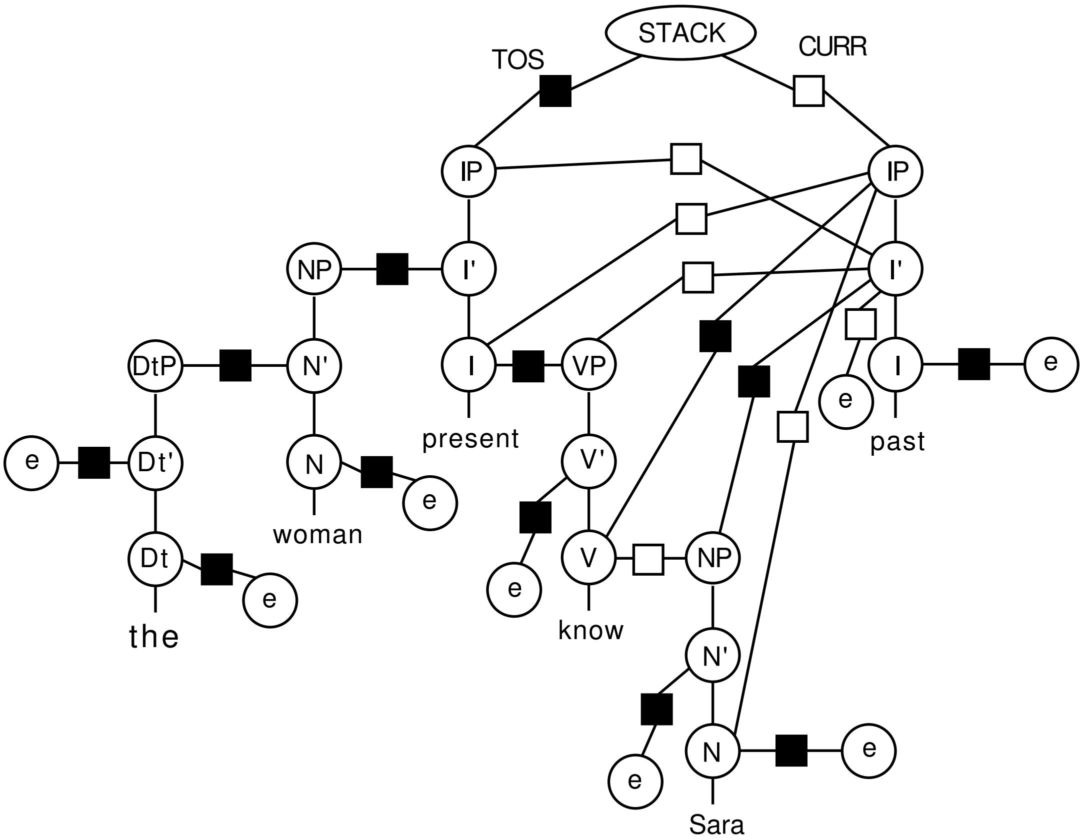


Figure 4.29: Active attachments after the network settles. The attachment of the NP *Sara* has been revised.

the NP as the complement of the V. The use of spreading activation allows the network to settle on the preferred solution in which the IP is incorporated into the current parse tree as the complement of the verb, as in Figure 4.29. This simultaneous attachment of the current phrase and revision of earlier attachments to accommodate it underlie the ability of the parser to model the reanalysis of structure in the human parser.[23]

The active attachment behavior of the parser is refined by the two additional computational assumptions that were adopted in the design of the model. The use of CBSA, and the adoption of a stack mechanism to support this, strongly restrict the attachments that can be considered by the parser. The only attachments that can compete simultaneously are those in the set of attachments between the current phrase and the tree on the top of the stack. In fact, the competitions among the allowed attachment nodes completely define a circumscribed set of logical attachment possibilities for both initial and revised attachments in the parser. These logical attachment possibilities are shown in Figure 4.32; they

[23]Again, the examples of Chapter 7 will substantiate this claim.

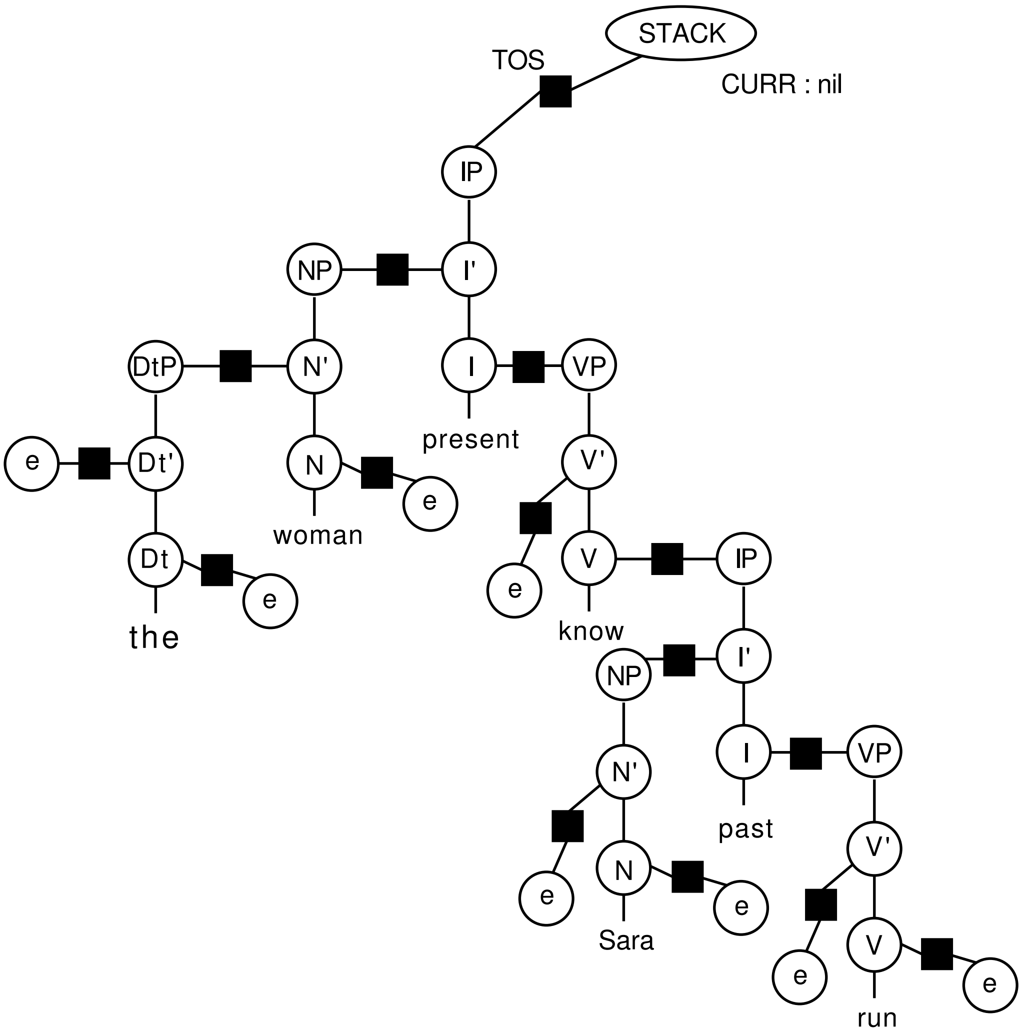


Figure 4.30: Final state of the parsing network.

follow directly from the propagation of local competitions among the attachment nodes, as discussed in Section 4.1.3. The direct and indirect effects of CBSA thus greatly constrain the attachment decisions made by the parser. For example, revising the attachment of the post-verbal NP in the example parse above is possible only because that revision involves one of these sets of logical attachment possibilities. In contrast, Chapter 7 will demonstrate cases where the parser makes initial decisions that cannot be revised as needed. In these sentences, the necessary attachments do not constitute one of the valid attachment sets defined by the competitive constraints; these are cases of "garden-path" sentences that people have difficulty parsing as well.

Another effect of the use of CBSA and the stack is to strengthen the property that a phrase must actively try to connect itself through attachment nodes to the developing parse

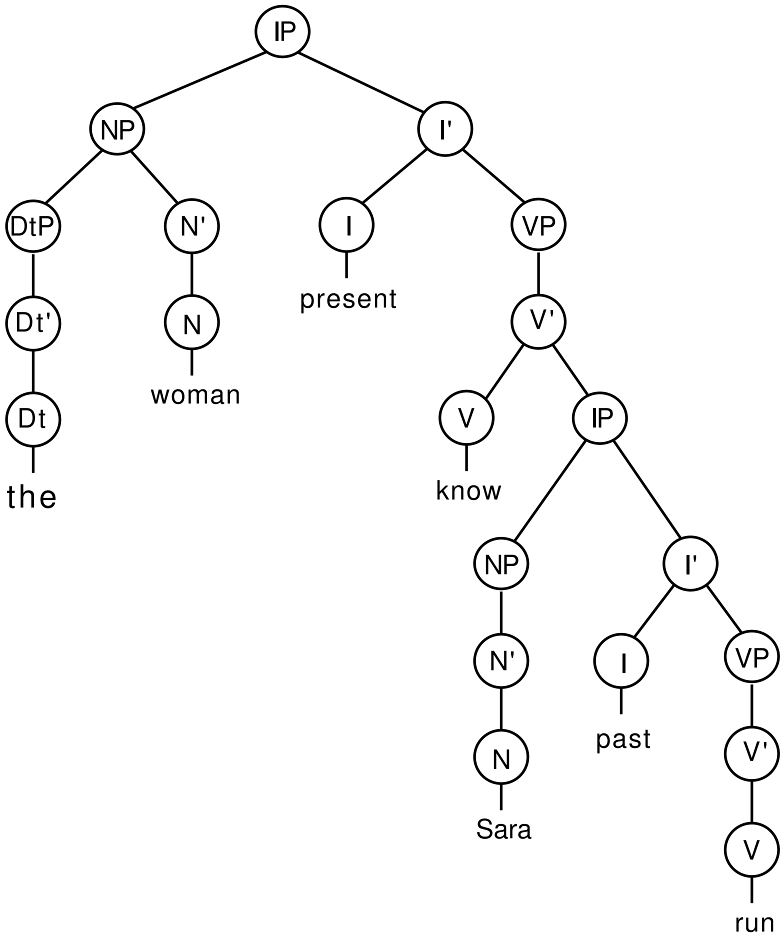


Figure 4.31: Parse tree corresponding to the network in Figure 4.30.

tree. In particular, an attachment node between two phrasal nodes must strongly compete for activation. This is because attachment nodes that lose their competition are deallocated; an attachment node that doesn't win *now* can never win at a later time in the parse, simply because it won't exist. For example, when the NP *Sara* in our example parse is first allocated, it *must* activate its initial attachment to the V node; problems will arise if it waits to see if a more preferred attachment will come along later. Suppose the attachment is *not* made, and the NP pushes itself onto the stack in a "wait-and-see" strategy, as in Figure 4.33. If the sentence now ends (as in *The woman knows Sara.*), the parse will fail because the NP is not connected to the parse tree for the rest of the sentence. There is no mechanism for re-establishing the necessary attachment node between the NP and the V. This property as well is key to the ability of the model to mimic human behavior; Chapter 7 will demonstrate how it accounts for some well-documented structural preferences.

The decision to dynamically instantiate the network leads to active memory management

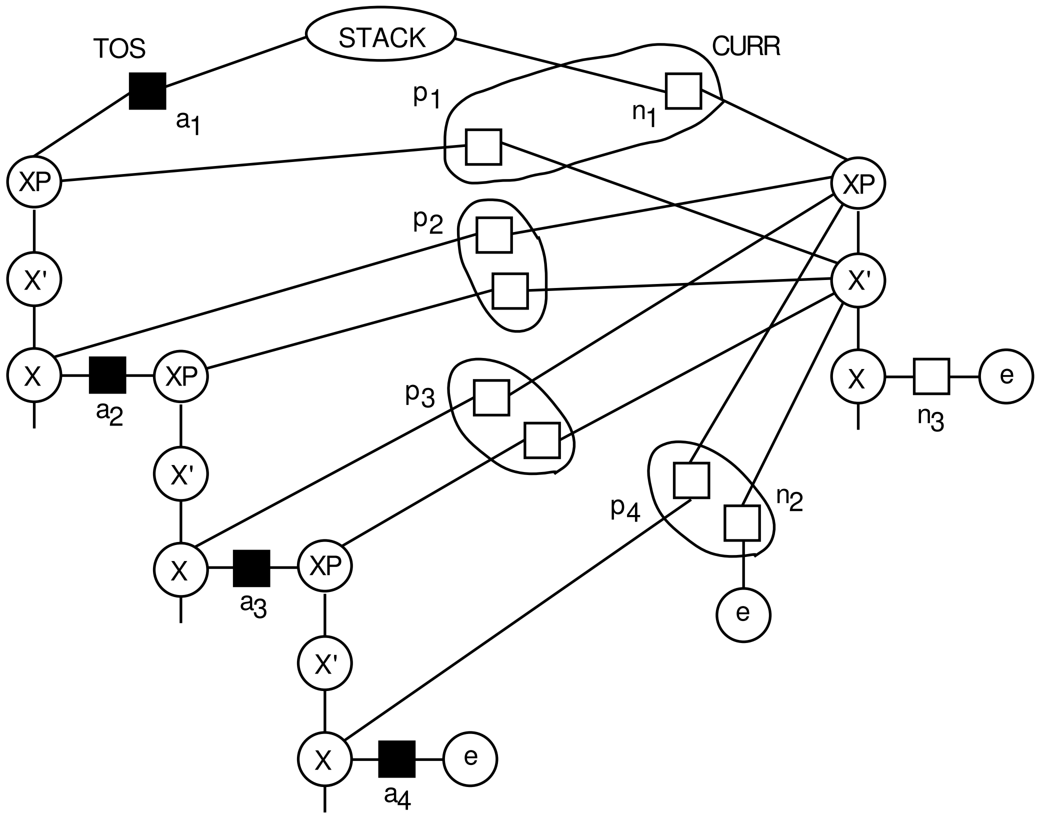


Figure 4.32: Logical possibilities for simultaneous activation of sets of attachment nodes, for an example TOS tree that is three phrases deep. (Only the right edge of the tree on top of the stack is shown, since the left side is unaffected by new attachments.) Attachments $a_1$ through $a_4$ were previously activated. If all of these attachments remain active, then $n_1$, $n_2$, and $n_3$ must become active; in this case, no attachments are made between CURR and TOS, and CURR pushes itself onto the stack. To attach CURR and TOS, the following must occur: exactly one of the prior attachments, $a_i$, must become inactive, and the corresponding pair of attachments, $p_i$, must become active. This relationship holds for a TOS tree of arbitrary depth.

techniques that further constrain the attachment behavior of the parser. For example, the decay of the activation of phrasal nodes is responsible for the parser's tendency to favor attachments to more recent parts of the parse tree. In the example above, the attachment (shown in Figure 4.30) of the final verb *run* to the most recent IP, rather than to the previous IP in the sentence, is aided by this recency effect. The fact that this behavior is a side-effect of an independently justified memory management mechanism allows the model to account for the robust recency effects in human parsing in a principled way.

Dynamic instantiation of the network also motivated the prohibition on top-down computation, which again restricts the parser's active attachment behavior. Structures that are

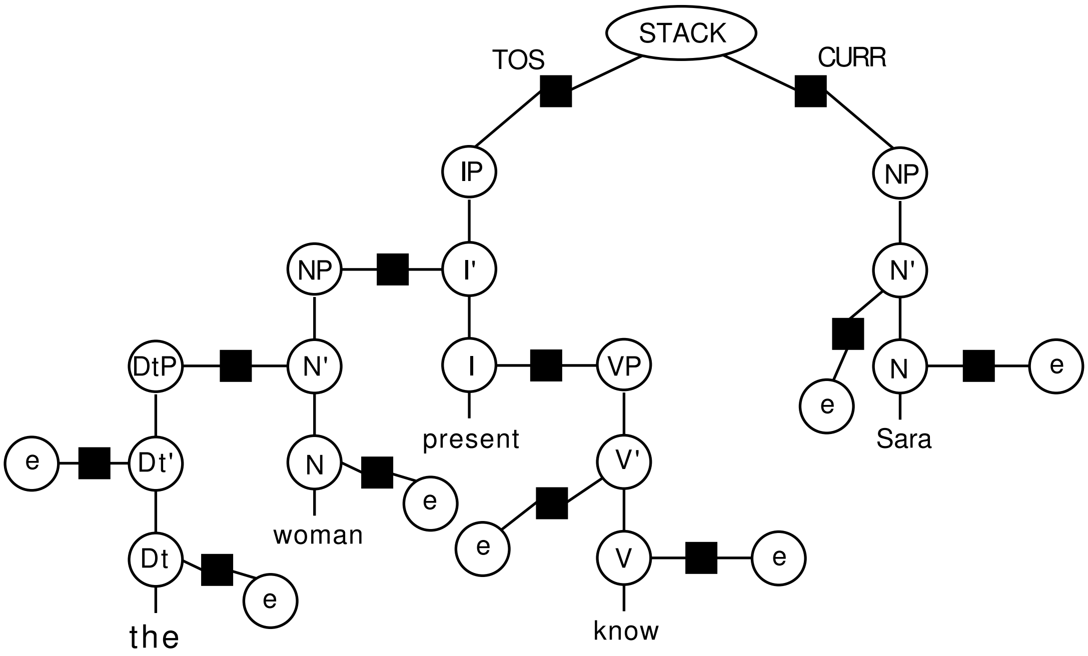


Figure 4.33: State of the parser if the NP *Sara* pushes itself onto the stack instead of attaching to *know*, given the input "(the woman present know Sara...)."

possible but not incontrovertible cannot affect the competitions between attachment nodes. Recall that the expectation for a phrase can be activated only by evidence in the input—that is, an attachment node can be created only between two existing phrases in the network, and a phrase can be allocated only given overt evidence. Thus in our example, at the point of processing the NP *Sara* in Figure 4.26, the verb's expectation for an IP complement as well as an NP complement cannot affect the competition of the NP node for that attachment site. No attachment node exists to represent the hypothetical possibility of an IP complement. Thus the NP attaches to the V, and when the evidence for an IP occurs later in the input, it necessitates a revision of this earlier attachment decision, as shown in Figure 4.29. Again, this basic parsing behavior mimics a number of human behaviors that will be discussed in detail in Chapter 7.

In conclusion, important aspects of the parser's attachment behavior strictly follow from a small set of well-motivated architectural assumptions underlying the model. These assumptions themselves directly rely on a set of mutually supporting computational and linguistic factors, so that in fact the attachment behaviors arise from these more fundamental principles.[24] Each of the behaviors discussed in this section is crucial to the results on struc-

[24]Thus, like the approaches of Gibson (1991) and Pritchett (1992), the competitive attachment model attempts to derive ambiguity resolution behaviors from properties of the grammatical theory. A difference in the approach taken here is that there is less emphasis on the precise effects of particular syntactic constraints

tural disambiguation to be presented in Chapter 7. Because each behavior can be traced back to basic assumptions about the natural language parsing process, we can directly evaluate how well these underlying principles are supported by the results of the model.

from the theory, and more emphasis on the architectural encoding of a constraint-based style of syntactic representation.

## Chapter 5

# Numeric Processing in the Parser

This chapter provides a detailed description of the numeric processing aspects of the competitive attachment parsing model. Parsing natural language within a massively parallel framework, without the use of rule-based knowledge to *a priori* constrain the network structure, is an unprecedented challenge to the competitive activation approach. The numeric activation techniques developed here represent a successful extension of competitive activation methods to the most complex and unstructured networks yet attempted. A large number of systematic simulations demonstrate the effectiveness of the techniques for controlling the spread of activation within a wide range of parsing configurations. The numeric functions enable the processing nodes to converge on a correct and consistent set of parse tree attachments in over 98% of the 1365 test configurations, establishing the competitive attachment model as a robust approach to natural language parsing.

Section 5.1 begins with a brief introduction to numeric processing in the parsing network. Section 5.2 describes the input, activation, and output functions that constitute the competitive spreading activation process in the network. Section 5.3 presents the results of the numeric functions within a large number of network configurations; these simulations demonstrate the basic attachment behaviors that arise from the numeric processing of the parser. Section 5.4 concludes with a discussion of some of the limitations of the numeric functions and parameters used by the parser.

## 5.1 Overview of the Numeric Processing

Each time the parser processes a new input word, by allocating its $\overline{\mathrm{X}}$ phrase and connecting the phrase to the existing network, the numeric processing of the network is triggered. Each node performs a numeric update/output processing loop, which re-computes the node's numeric functions. The nodes in the network are synchronized so that the update routines are performed simultaneously within each node, followed by the simultaneous computation of output. Updating a node consists of determining the amount of its numeric input, and calculating its new activation level accordingly. The output portion of the loop computes the amount of activation to be sent from a node to each of its neighbors. This spreading

```
begin {Process Input Sentence}
while there are more input tokens do
        {Each pass through this loop is a run of the network.}
        Get next input token.
        Allocate an X̄ phrase for the current input token.
        Initialize attachment nodes between the current phrase
            and the top of the stack.
        Reinitialize competing attachment nodes.
        until the network is an acceptable state do
                {Each pass through this loop is an iteration of the network.}
                for each node in the network do
                        Update(node).
                for each node in the network do
                        Output(node).
end {Process Input Sentence}
```

Figure 5.1: The iterative algorithm for processing the nodes of the network.

activation loop continues until the network reaches an acceptable state.[1] An acceptable state of the network is defined as one in which each phrasal node sends all of its activation to exactly one of its attachment nodes, and each attachment node is either turned on (fully active) or off (inactive) by its phrasal nodes. In such a state, the active portion of the network forms a valid parse of the input seen thus far.

In the following, a *run* of the network refers to one complete cycle of network processing, beginning with the creation of a new $\overline{\mathrm{X}}$ phrase and ending when the network achieves an acceptable state that incorporates the phrase into the parse. There is a run of the network for each token that is input to the parser. An *iteration* of the network is a single step through the update/output loop for all of the nodes in the network. Since the network iterates until it reaches an acceptable state, the number of iterations per run depends on properties of the competing attachment nodes within that run. An overview of the network processing is given in Figure 5.1. The remainder of this chapter will fill in the details of this numeric processing algorithm, and demonstrate the algorithm's effectiveness on a wide range of inputs.

[1]In some rare configurations, the network does not reach an acceptable state, and so the loop has a bound on the number of iterations it can perform. The cases in which the network does not converge on an acceptable set of attachments are discussed in detail in Section 5.3.4.

| Function | Definition |
|---|---|
| *ext-in* | The input to a node from a source that is external to the network. |
| *in* | The input to a node from other nodes within the network (referred to as the "within-network input"). |
| *act* | The activation level of a node. |
| *out* | The output of a node to other nodes in the network. |

Table 5.1: The four numeric activation functions used in the parser.

## 5.2 Spreading Activation Functions

As noted above, spreading activation in the parser is implemented by a loop of update and output functions called by each node in the network. The nodes of the network are defined by an object-oriented hierarchy in which different types of objects are processing nodes that have appropriate methods defined for computing the given functions.[2] There are four types of nodes: phrasal nodes (p-nodes), empty phrasal nodes, attachment nodes (a-nodes), and the stack node. Empty nodes are a subtype of p-node in the object hierarchy. Surprisingly, the stack node is also a subtype of p-node; the stack, both symbolically and numerically, behaves like a degenerate phrasal node. Attachment nodes are quite different from these other three types of nodes, although a-nodes and p-nodes share a common parent called "numeric node."

All numeric nodes call the same functions within the update/output loop of the network, but the particular computation performed for each function depends on the type of node. That is, each spreading activation function is a generic function whose effect in a given context depends on the type of node that calls it. There are four such numeric functions defined; see Table 5.1.[3] Since there are four types of node objects, each function name is subscripted with a symbol to indicate the appropriate method of computing the function. For example, $in_p$ refers to the within-network input calculation for a p-node; $out_{ap}$ refers to the method of computing the output from a p-node to an a-node. Symbols may be further subscripted to indicate the particular node performing the calculation: $out_{a_j p_i}$ refers to the output of p-node $p_i$ to a-node $a_j$.

The update portion of the spreading activation loop consists of a sequence of calls to the *ext-in*, *in*, and *act* functions; the output portion of the loop consists of calling the single

[2]As stated earlier, the parser is implemented using the CLOS object oriented package of Allegro Common Lisp.

[3]This does not mean that there are 16 different functions (4 functions $\times$ 4 node types) defined in the parser, since a subtype of a node often inherits a given function from its parent type. There are actually 9 different functions defined: the four basic functions for p-nodes, all of which are shared by the stack node and three of which are shared by empty phrasal nodes; a separate activation function for empty phrasal nodes; and the four basic functions for a-nodes. Three of the functions shared by the subtypes of p-nodes use a single different constant value in the inherited function.

function *out*. The loop is executed at each tick of the network's discrete time clock; thus, each function is recomputed at each time $t$. This time parameter is not explicitly shown in the following formulas, unless the calculation of a function refers to two different times (the current time $t$ and the preceding time $t-1$). In the default case, the time of each function within a formula refers to the same time $t$.

The value of each function is in the range 0 to 1.0, unless explicitly stated otherwise in the text.

### 5.2.1 Phrasal Nodes

Each phrasal node, or p-node, receives a fixed external input of 1.0; this external input represents the activation of the p-node by an input token.[4] The p-node also receives input from the a-nodes to which it is attached; this latter input is summed to yield the p-node's *in* function (whose result may be greater than 1.0):

$$in_{p_i}(t) = \sum_j out_{p_i a_j}(t-1) \tag{5.1}$$

where:

| | |
|---|---|
| $in_{p_i}(t)$ | is the within-network input to p-node $p_i$ at time $t$. |
| $out_{p_i a_j}(t)$ | is the output from a-node $a_j$ to $p_i$ at time $t$. |

The activation level of the p-node is set to the maximum of its external input and the summed input from its a-nodes, minus a decay factor:

$$act_{p_i} = \max[ext\text{-}in_p, in_{p_i}] - \delta_{p_i} \tag{5.2}$$

where:

| | |
|---|---|
| $act_{p_i}$ | is the activation level for p-node $p_i$. |
| $ext\text{-}in_p$ | is the external input for p-nodes (currently set to 1.0). |
| $in_{p_i}$ | is the within-network input for $p_i$. |
| $\delta_{p_i}$ | is the decay factor for $p_i$. |

Since the summed input from its a-nodes may be less than 1.0, the external input places a minimum of 1.0 on the value of a p-node's activation. This is important because it means that the amount of activation that a p-node has for dividing among its competing attachments is high even at the beginning of a run of the network. This strong input to the a-nodes allows them to more quickly increase in activation.

While the external input places a minimum on the activation level of a p-node, the summed input from the a-nodes has the effect that the p-node's activation level may in fact be greater, in proportion to the input from its a-nodes. This ensures that the p-node has

[4] Lexical ambiguity would lead to multiple projections for a single input word; the external input would then be divided among these projections. See Section 8.2.1.

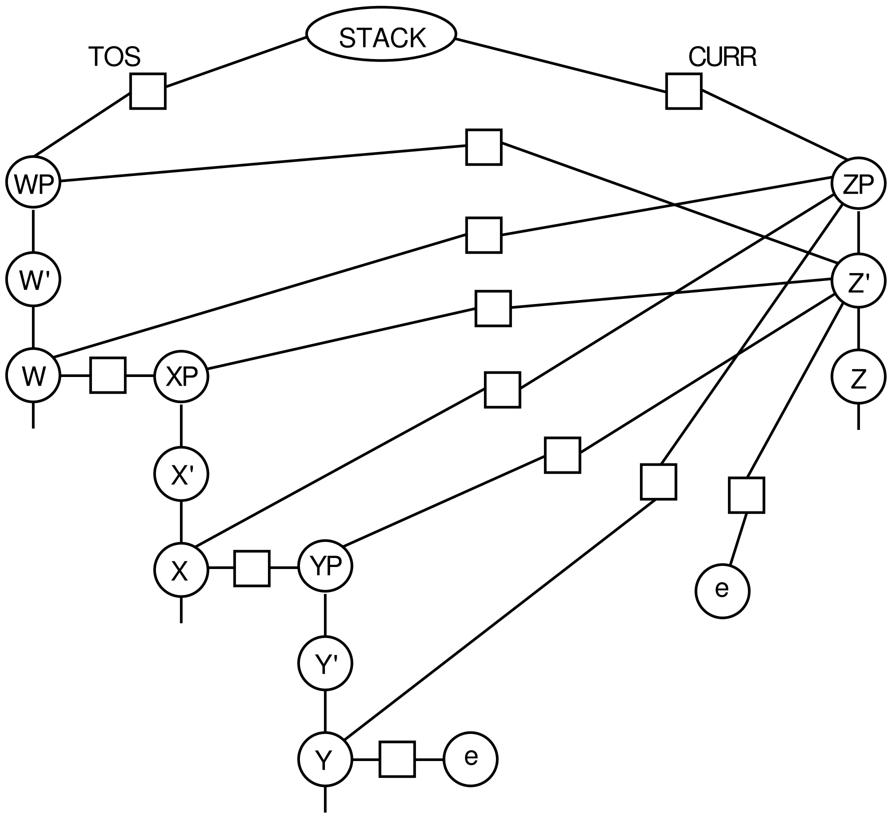


Figure 5.2: All previously processed X and XP nodes connect to exactly two a-nodes, but the number of a-nodes that the current X′ and XP connect to depends on the depth of the right edge of the partial parse tree on the top of the stack. In this example network, each of WP, W, XP, X, YP, and Y connect to two a-nodes, while each of ZP and Z′ connect to four a-nodes.

more activation to output in the case where it has more potential attachments that it is choosing between. In parsing, the p-nodes of the current phrase might have more than two attachment possibilities, while the previously processed phrasal nodes on the top of the stack will always have exactly two; this situation is demonstrated in Figure 5.2. Since two p-nodes must agree to turn on an attachment node, it is important that the output activation of the current p-nodes is not "diluted" by the fact that they may have more choices to decide between. If the current p-nodes had an equivalent activation level to spread among more a-nodes, their vote in turning on an a-node would be unfairly disadvantaged.

In addition to the external and within-network inputs, there is a final component of a p-node's activation function, the decay factor $\delta_{p_i}$. In the simple case, this decay factor is a function of the difference between the current time and the time at which the p-node was created. This is to ensure that a syntactic phrase will eventually become inactive, so that its nodes can be re-used if needed; the least recent portion of the parse tree will become inactive first. When a p-node participates in a new attachment, however, the amount of its decay needs to reflect that more recent participation in the parse tree. Thus the decay factor must be a function of the difference between the current time, $t$, and the time of the most recent attachment for a phrase, $t_{p_i}$. In the current implementation, the following linear function is used: $.05(t - t_{p_i})$.

The output function of a p-node is a little more complex than its input and activation functions. As described in Chapter 4, a p-node must proportionally allocate its activation among its a-nodes using a competition-based spreading activation (CBSA) mechanism. The precise function used by p-nodes is the following:

$$out_{a_j p_i} = \frac{({act_{a_j}}^v + q) wgt_{ji}}{\sum_k ({act_{a_k}}^v + q) wgt_{ki}} act_{p_i} \tag{5.3}$$

where:

| | |
|---|---|
| $out_{a_j p_i}$ | is the output from p-node $p_i$ to a-node $a_j$. |
| $act_{a_j}$ | is the activation of a-node $a_j$. |
| $act_{p_i}$ | is the activation of p-node $p_i$. |
| $wgt_{ji}$ | is the (positive) weight on the connection between p-node $p_i$ and a-node $a_j$ (weights are symmetric). |
| $k$ | ranges over all a-nodes connected to p-node $p_i$. |
| $v$ | is the competitive exponent (explained in the text). |
| $q$ | is the competitive additive factor (explained in the text). |

This function is a variant of the CBSA function given in equation 3.1 on page 21; it states that a p-node divides up its output activation to its a-nodes in proportion to their weighted activation. The weighted activation:

$$({act_{a_j}}^v + q) wgt_{ji} \tag{5.4}$$

includes the constants $v$ and $q$, which determine the degree of competition induced by the $out_{ap}$ function.[5] Increasing the value of the exponent $v$ makes the competition more pronounced, since any difference in the activation of the competing a-nodes is magnified by, for example, squaring or cubing the activation values. Increasing the value of the additive factor $q$, on the other hand, makes the competition less pronounced, by "swamping out" the differences between a-node activations with this additive factor.

In the parser, $v$ is gradually increased from 1 to 3 (in steps of 1) during each run of the network, so that competition is increased over time. This scheme provides an initial time period in which all of the a-nodes have a chance to amass some activation under conditions that are not extremely competitive. This less competitive period is crucial because it allows the network the opportunity to find a globally satisfying set of attachments, rather than

[5]With $v$ having the default value of 1 and $q$ having the default value of 0, equation 5.3 reduces to equation 3.1.

just immediately focusing on the initially preferred set of a-nodes (which might not form a consistent set of attachments). The competitiveness of the a-nodes must be gradually increased, however, in order to force the p-nodes to eventually choose a single a-node to activate, and thus ensure that the network reaches an acceptable state.

The value of $q$ in the parser is effectively 0; it is set to a very small number ($10^{-15}$) solely for the purpose of avoiding errors of division by 0 in the case where all of the a-nodes connected to a p-node are inactive.

Not shown in equation 5.3 is the fact that the output is thresholded; if output to some a-node is less than the threshold (currently set to .15),[6] then the p-node sends an output activation of 0. This thresholding contributes to cleaner competitive behavior, since the result is that a weakly competing a-node will be turned off more quickly. Output from p-nodes is also capped; it cannot be greater than 1.0.

### 5.2.2 Empty Nodes

Empty nodes are a special kind of phrasal node; in the parser's object hierarchy of nodes they are a subtype of p-node. The differences in the numeric processing of these two types of nodes arise from the following fundamental properties. Ordinary phrasal nodes are allocated in response to evidence in the list of lexical items that are input to the parser, and must participate in the final parse tree for that sentence. By contrast, empty nodes are automatically allocated with each phrase, but in fact may not be used in the final parse. Evidence for the existence of an empty node in the parse tree must be gathered from other parts of the tree. This basic difference in the licensing of the syntactic objects represented by these two types of nodes is reflected in their spreading activation functions.

It was noted above that the external input of 1.0 to a p-node represents its activation by a token in the input. Since an empty node has no direct evidence in the input, its external input is set to a minimal value, and its input from within the network thus takes on greater significance. The function for computing the within-network input to an empty node is exactly the same as that for a non-empty p-node, which was shown in equation 5.1. An empty node has only one potential attachment—a single a-node to which it is connected—giving the sum a single term.

The activation function for an empty node reflects the primary difference between it and an ordinary p-node—the fact that while a p-node is strongly activated and gradually decays over time, an empty node is weakly activated and must gradually amass activation over time in order to not become inactive (and thus be deallocated). The activation function for an empty node is similar to that of a p-node in that it involves taking the maximum of its external input and a function of its within-network input, and incorporates an element of decay. But the precise formulation is different from that of the p-node activation function in equation 5.2, and this difference allows the activation level of an empty node to reflect its

[6]The value of this threshold, and all other constants mentioned in this chapter, were determined empirically.

strength of evidence within the tree. In order to keep the activation level from oscillating, the activation function is one in which the new activation level is computed as a small change from the old activation level:

$$act_{e_i}(t) = \max[ext\text{-}in_e, \\ (1-\delta_{e_i})act_{e_i}(t-1) + \delta_{e_i} in_{e_i}(t)] \tag{5.5}$$

where:

| | |
|---|---|
| $act_{e_i}(t)$ | is the activation level for empty node $e_i$ at time $t$. |
| $ext\text{-}in_e$ | is the external input for empty nodes (currently set to .33). |
| $\delta_e$ | is the decay rate for empty nodes (currently set to .1). |
| $in_{e_i}(t)$ | is the within-network input for $e_i$ at time t ($in_e \equiv in_p$). |

The current activation level of an empty node is thus the decayed value of its old activation plus a small portion of the input from its a-node, with a minimum result of $ext\text{-}in_e$. The percentage of decay of the activation level and the percentage of input that is added are the same, $\delta_e$.

Finally, the output function for empty nodes is the same as the output function for p-nodes. As with the input function, this function greatly simplifies due to the fact that empty nodes have a single potential attachment—an empty node outputs all of its activation to the single a-node.

In summary, the input and output functions for empty nodes are inherited from p-nodes, although their effect is simpler due to the fact that empty nodes connect to a single attachment node. The external input to an empty node is a fraction of the amount of external input to an ordinary p-node, because an empty node needs additional evidence to justify its participation in the parse tree. Although sharing some of the same elements, the activation function for an empty node must be different from that of a p-node, because the activation of an empty node reflects the gradual amassing of evidence for its existence in the tree.

### 5.2.3 The Stack Node

The stack node in essence behaves like a degenerate p-node; it is defined as a subtype of p-node, and its numeric processing is quite similar to that of a non-empty p-node. The external input to the stack is the same as for p-nodes—it is a fixed input of 1.0. Its within-network input is computed with the same function as that used by p-nodes (equation 5.1)—that is, $in_s$ is the summed input to the stack from its a-nodes. The stack's activation function is also inherited from p-nodes (equation 5.2). However, a difference between the stack and other p-nodes that affects its activation level is that the stack's decay factor $\delta_s$ is set to 0. Because the stack node must remain active throughout the parse, there is no decay of the stack's activation; besides, there is no need, as there is with p-nodes, to make its node available for re-use.

The output function used by the stack is also the same as the output function for p-nodes (equation 5.3). The only difference is that the stack's output threshold is 0, so that output from the stack is not thresholded. An additional difference in behavior arises from the fact that during a run of the network, the competitive exponent $v$ for the stack node increases from 1 to 2 only, instead of from 1 to 3 as for syntactic p-nodes. Both of these factors—the lack of thresholding and the limit on increasing the level of competition—serve to decrease the level of competition between the TOS and CURR a-nodes connected to the stack. These two differences in the stack's output computation are motivated by the need for the stack to be able to activate both TOS and CURR simultaneously. Unlike p-nodes, the stack node does not have to choose exactly one a-node to activate, it simply must activate *at least* one, and sometimes the necessary behavior is to activate both. The surprising result is that with only these minor differences between the numeric functions of the stack and p-nodes, there is a major difference in the resulting behavior. The stack may appropriately activate either or both of TOS and CURR, while a p-node always chooses a single a-node to activate.

### 5.2.4 Attachment Nodes

Phrasal nodes, empty phrasal nodes, and even the stack node, all have very similar numeric activation functions. This similarity arises from the fact that, although they have important differences between them, all of these node types have the same basic role within the context of numeric processing in the network: their primary purpose is to decide whether or not to activate some attachment node. This section will describe the numeric functions of these attachment nodes. These functions are quite dissimilar from the ones presented above, because a-nodes have the very different role within the network of weighing alternative evidence and boosting or lessening their activation level accordingly.

An a-node has no external input, because its activation depends entirely on the evidence for its existence that it receives from the phrasal nodes to which it is connected. An a-node $a_i$ has three numeric values that are used in determining its within-network input function: $input_{a_ip_j}$, $input_{a_ip_k}$, and $state_{a_i}$. The values $input_{a_ip_j}$ and $input_{a_ip_k}$ are the numeric inputs that it receives from its two p-nodes $p_j$ and $p_k$; $input_{a_ip_n}$ is equivalent to $out_{a_ip_n}$ described above—that is, the output of p-node $p_n$ to a-node $a_i$. For those a-nodes that are connected to an empty node or the stack, $input_{a_ip}$ refers to input from those nodes as well, since they are subtypes of p-node. The value $state_{a_i}$ is computed based on the symbolic features that are passed to a-node $a_i$. The more grammatical constraints that the a-node's features satisfy, the higher the state value; invalid features cause the state value to drop to 0. These three numeric values, $input_{a_ip_j}$, $input_{a_ip_k}$, and $state_{a_i}$, are combined in numerical versions of AND and OR operations to produce the input function for an a-node. The $in_{\mathrm{AND}}$, $in_{\mathrm{OR}}$, and $in_a$ functions described below were inspired by the corresponding functions in Reggia, Marsland, & Berndt (1988).

The AND function is as follows:

$$in_{\mathrm{AND}_i} = input_{a_ip_j} \cdot input_{a_ip_k} \cdot state_{a_i} \tag{5.6}$$

Taking the product of the three values indicates that the input to an a-node should be high to the extent that all three of these values are high. That is, an a-node must receive activation from both its p-nodes, as well as reasonably satisfying the grammatical constraints on it, in order to get strong numeric input. This function alone is not a sufficient input function, however; the a-node must be given a chance to get some activation even if only one of these components is high. For example, very strong input from one of the p-nodes may convince the other p-node to increase its output to that a-node.

In order to allow for these numeric components of the a-node to affect the input more individually, there is an OR function taken into account as well:

$$in_{\mathrm{OR}_i} = 1 - (1 - input_{a_ip_j})(1 - input_{a_ip_k})(1 - state_{a_i} \cdot state\text{-}weight) \tag{5.7}$$

This function is a non-linear accumulative function of the a-node's three numeric values, and allows the individual components to have a greater impact on the combined value than does the AND function. The AND function says that in order for the input to be high, *all three* of the components must be high; by contrast, the OR function says that in order for the input to be high, *at least one* of the components must be high. Only a portion of $state_{a_i}$ is used in the OR function (currently $state\text{-}weight = .25$) to ensure that the state value won't dominate the result; it is more important for the p-node inputs to have an individual effect.

The within-network input function for an a-node is a weighted sum of the AND and OR functions:

$$\begin{aligned} in_{a_i} &= in_{\mathrm{AND}_i} \cdot act_{a_i} \\ &\quad + in_{\mathrm{OR}_i}(1 - act_{a_i}) \\ &\quad - in_{\mathrm{OR}_i} act_{a_i} k \end{aligned} \tag{5.8}$$

where:

$in_{a_i}$ is the within-network input for $a_i$.
$act_{a_i}$ is the activation level for a-node $a_i$. (This function is described below.)
$k$ is a constant between 0 and 1.0 (currently set to .5).

Consider the first two lines of equation 5.8. These two lines have the effect of weighting the OR component of the function more when the activation of the a-node is low, and weighting the AND component of the function more when the activation of the a-node is high. This essentially means that, initially, an a-node receives activation as long as at least one of its numeric components (the inputs from its p-nodes or the state) is active. However, as the a-node increases in activation, it is necessary that all of its numeric components are highly active for it to continue to receive activation. Thus as the evidence for an attachment increases, it becomes crucial for this evidence to agree. The higher weighting of the AND function when the activation of the a-node increases captures the critical semantics of an active attachment—that it must reasonably satisfy its symbolic constraints (its state value is high), *and* it must receive all of the output from each of its two p-nodes (both of its input values are high).

The last line of the function further emphasizes the centrality of the AND component in capturing the semantics of an a-node: it subtracts off a fraction of the OR function, in order to ensure that the OR component of the input combining function is not weighted too strongly. The amount subtracted is in proportion to the level of activation; again, this allows for incomplete evidence for an attachment to initially have some effect, but forces the evidence to be complete as it becomes stronger.

Finally, the input function has one further component not shown in equation 5.8: if either of the two input values or the state is 0, then $in_{a_i}$ is set to 0. If the state value is 0, that means that the attachment violates some grammatical constraint and is therefore invalid; it must be turned off. If either of the two inputs is 0, that means that the p-node that is sending no output to the a-node has made a choice not to activate this attachment; again, the a-node should be turned off.

The initial version of the activation function that employed this input was the following:

$$\begin{aligned} act_{a_i}(t) &= in_{a_i}(t) \\ &\quad + (1 - in_{a_i}(t))[2act_{a_i}{}^2(t-1) - act_{a_i}(t-1)] \end{aligned} \tag{5.9}$$

where:

$act_{a_i}(t)$ is the activation level for a-node $a_i$ at time t.
$in_{a_i}(t)$ is the within-network input for $a_i$ at time t.

This is the same function as that used for computing activation levels in the print-to-sound network of Reggia, Marsland, & Berndt (1988).[7] The effect of the function is that the current activation level of an a-node $a_i$ is set to the current input $in_{a_i}$, plus or minus a fraction of $(1-in_{a_i})$ that is equal to twice the previous activation squared minus the previous activation (the term $2act_{a_i}{}^2(t-1) - act_{a_i}(t-1)$). The fraction of $(1 - in_{a_i})$ that is added to $in_{a_i}$ is positive when the previous activation is greater than .5, and negative when the previous activation is less than .5. Thus when the activation level of an a-node is greater than .5, the new activation will be set to the proportionately increased input value; when the activation level is less than .5, the new activation will be set to the proportionately decreased input value. As long as the input level is less than or equal to 1.0, the activation level will also have a maximum of 1.0, since it equals the input plus or minus a fraction of 1 minus the input.[8]

In the parser, an additional term was added to equation 5.9 to yield the following final activation function for a-nodes:

$$\begin{aligned} act_{a_i}(t) &= in_{a_i}(t) \\ &\quad + (1 - in_{a_i}(t))[2act_{a_i}{}^2(t-1) - act_{a_i}(t-1)] \\ &\quad + \rho_{a_i} act_{a_i}(t-1) \end{aligned} \tag{5.10}$$

[7]The activation function in that work is stated as the change in activation of a node, and therefore appears different in form.

[8]While it is possible for this function to take on values less than 0, this does not happen in practice; it requires the input to be quite small, and the thresholding of output to the a-nodes prevents this.

where:
$act_{a_i}(t)$ is the activation level for a-node $a_i$ at time t.
$in_{a_i}(t)$ is the within-network input for $a_i$ at time t.
$\rho_{a_i}$ is the *reinforcement* factor for $a_i$ (explained in the text).

This function reflects the fact that an a-node is self-reinforcing over time; that is, in addition to its input function, an a-node receives additional activation from itself in proportion to its age. The amount of additional activation added is the old activation level times $\rho_{a_i}$, which is the rate of reinforcement for that a-node. The reinforcing term is necessary to counteract the gradual decay of the p-nodes to which an a-node is attached; it is desirable for an active a-node to have a fairly stable activation level, and without this self-reinforcement it will decay too quickly. In the current implementation, $\rho_{a_i} = .025(t - t_{a_i})$, where $t_{a_i}$ is the time that $a_i$ was allocated.[9]

There are some final details of the activation function that are not shown in equation 5.10. First, the reinforcement term that has been added means that the activation level may exceed 1.0; to avoid this, the function has an explicit ceiling of 1.0. Second, the activation level is set to 0 if the input is 0, regardless of the result of the above function. This ensures cleaner competitive behavior among the a-nodes.

The output function for an a-node is very straightforward; an a-node simply sends its weighted activation to its p-nodes:

$$out_{p_j a_i} = wgt_{ji} act_{a_i} \tag{5.11}$$

where:
$out_{p_j a_i}$ is the output from a-node $a_i$ to p-node $p_j$.
$wgt_{ji}$ is the (positive) weight on the connection between a-node $a_i$ and p-node $p_j$ (weights are symmetric).
$act_{a_i}$ is the activation of a-node $a_i$.

A-nodes that are connected to the stack use this same output function to send activation to it. The TOS and CURR a-nodes have a weight of 1.0 on their links, so they send their entire activation to the stack. The a-nodes on the REST list have a weight of 0 on their links, so that they send no activation to the stack.

Given the numeric functions described above, a-nodes are very unlikely to gain in activation once their activation level falls below .05, and are very unlikely to decrease in activation once their activation level rises above .4. Thus, these values are used as the thresholds in determining when the network is in an acceptable state—defined as one in which the activation level of each a-node is either below .05 or above .4.

[9] In fact, $\rho$ can be stated in terms of the decay factor for p-nodes, directly reflecting the fact that the self-reinforcement of a-nodes counteracts the decay of p-nodes: $\rho_{a_i} = .5\delta_{p_j}$, where $p_j$ is the more recent of the two p-nodes that $a_i$ connects to.

| Property | Definition |
|---|---|
| Convergence | The network must reach an acceptable state within a reasonable number of iterations. |
| Correctness | The solution set of active a-nodes must form a valid set of parse tree attachments. |
| Reasonableness | If there is more than one correct solution, the parser must make a choice that is appropriate and predictable. |
| Consistency | The network must exhibit similar behavior in the solution on which it converges across a range of inputs. |

Table 5.2: The four criteria according to which the attachment behavior of the parser is evaluated.

## 5.3 Numeric Processing Experiments

This section presents an evaluation of the parsing behavior induced by the spreading activation functions presented above. The goal of the parsing network is to incorporate each input phrase into a valid parse. To consider the parser to have achieved this goal, its behavior in response to an input must be shown to have the properties listed in Table 5.2. This section begins with a presentation of the motivations for testing the parser at this stage, and a discussion of the development of the appropriate test cases. The results of running the parsing network on these test cases are then presented. The results demonstrate that the numeric functions presented in Section 5.2 produce network behavior that meets the four criteria of convergence, correctness, reasonableness, and consistency.

### 5.3.1 Motivations

Two properties of the network make it possible to test the numeric processing of the parser, before introducing the further complexity of its symbolic capabilities. First, the attachment behavior relies purely on numeric processing; the numeric state value of the a-nodes captures all of the symbolic information that is relevant to attachment decisions. Because the state value provides the sole bridge between symbolic and numeric processing, it is straightforward to test the numeric processing behavior that determines attachments by manually setting the state value to reflect the desired test cases.

The second fact that simplifies testing is the following. Before each run of the network, all existing attachment nodes are reinitialized—that is, their activation level is set to 0. (See Figure 5.1 on page 61.) This step is taken in order to simplify numeric processing by having all competing attachment nodes, both new and old, begin with the same level of activation. The reinitialization has the side effect that, for each run, the only relevant information from prior runs is the set of discrete attachment decisions that were made, not the precise activation levels of the a-nodes. If activation persisted from one run to the next, the results to evaluate would be the series of attachment decisions that the parser makes on a given input sequence. But since the only relevant information is the result of prior attachment decisions, which are encoded in the structure of the network, each test case can consist of a single network configuration. In this case, a result is much simpler to evaluate, since it consists of the attachment decision made by the parser in a single run.

It is not only possible to test the attachment behavior of the parser at this point, it is highly desirable as well. The introduction of actual linguistic knowledge into the parser makes the state of the network very complex. It is extremely difficult to devise a set of thorough, methodical tests based on real lexical items. Even if it were possible, the test cases risk being too dependent on the particular lexical items that are chosen. The approach taken here is to provide a more systematic set of tests by abstracting away from the precise symbolic input. The network behavior will be demonstrated on a highly structured set of sample inputs. The goal is to demonstrate that the network's properties of convergence, correctness, reasonableness, and consistency hold across a broad range of input conditions; they do not arise only in a hand-picked set of example sentences.

This is not to deny the importance of evaluating the behavior of the complete parser on actual linguistic input. Chapter 7 is devoted to such results, which demonstrate how the parser processes syntactic ambiguities within a range of example sentences. The numeric experiments reveal the basic attachment behavior of the parser, while Chapter 7 relates this behavior to relevant examples from the psycholinguistic literature. By showing that the attachment decisions that mimic human behavior are not merely an artifact of the particular examples, these numeric simulations provide a foundation for the later discussion. Chapter 7 is not only supported by, but in turn reinforces, the conclusions of this section: Since it is impossible for the numeric simulations to test every configuration of the parser, the simulations of Chapter 7 will provide evidence that the attachment behaviors observed here do in fact hold under conditions arising from actual linguistic input.

### 5.3.2 Test Cases

In order to reasonably test the numeric behavior of the parser, it must be determined what the relevant inputs are to a run of the network, so that these can be varied in a meaningful way. Each run of the network results in a set of attachment decisions that encode how the newest input phrase is incorporated into the existing parsing network. The nodes of interest are those a-nodes that represent attachments that are competing for activation in this run. The set of competing a-nodes consists of all the new a-nodes established for the current input phrase, plus any existing a-nodes that these new ones compete with. An example network with the competing a-nodes highlighted is shown in Figure 5.3.

One relevant input to the network is the state value of each of these competing a-nodes. The state value encodes important symbolic information, and is a strong contributor to the activation level of an a-node. The other relevant factor in a run of the network is the actual structure of the parsing network, which is determined by the result of any prior attachment decisions. Since the current input phrase is only connected to the right edge of the partial parse tree on the top of the stack, the only pertinent aspect of the existing network is the

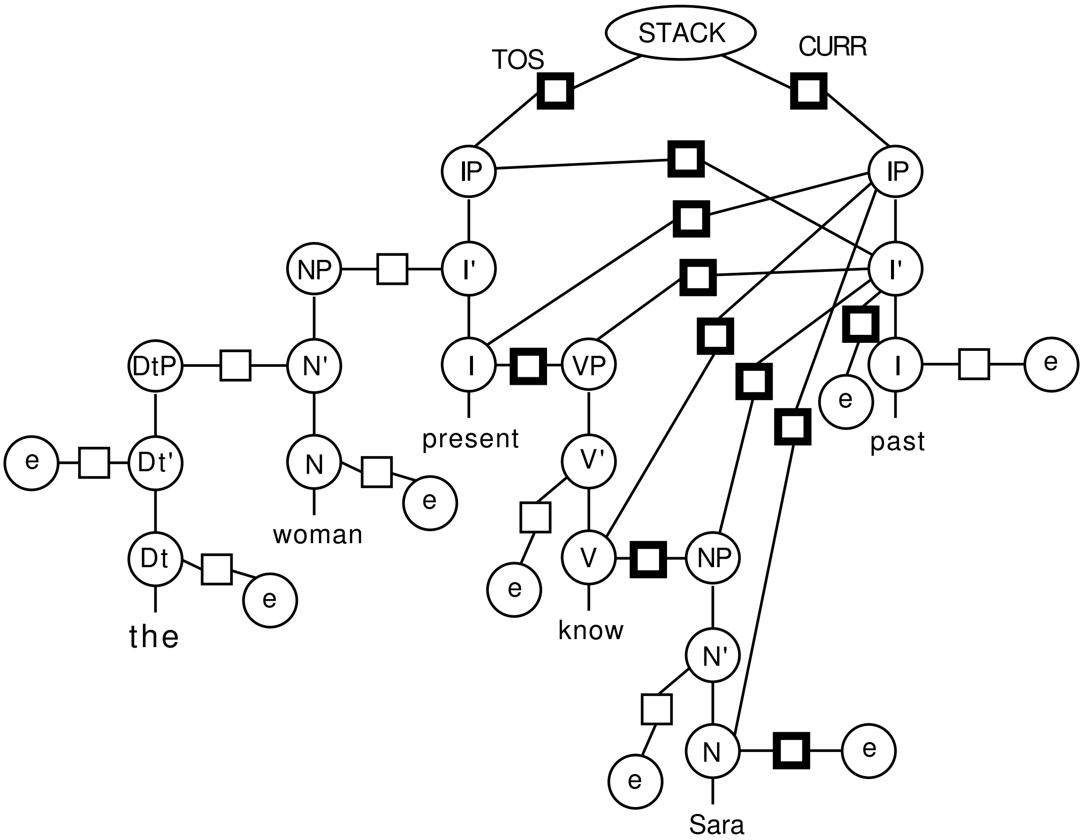


Figure 5.3: Example network with the set of competing a-nodes highlighted.

structure of this right edge. Furthermore, the only structural aspect of this right edge that can vary is its depth. The depth of the right edge of the partial parse tree on the stack is important because it determines the number of competing a-nodes.

The test cases for the set of numeric simulations will thus be enumerated by varying both the state values of the competing a-nodes and the depth of the tree on the stack. Depth of a tree in the following will mean the number of phrases (*not* the number of nodes) along the right edge of the tree. Let $n_s$ denote the number of different state values that the a-nodes will be allowed to take on, $d$ denote the depth of the tree on top of the stack, and $f(d)$ denote the number of competing a-nodes. Then the number of possible initial configurations of the network for a tree of a particular depth on the top of the stack is ${n_s}^{f(d)}$. The value of $f(d)$ is determined by the structure of the network: $f(d) = new(d) + old(d)$, where $new(d)$ is the number of new a-nodes allocated for the current input phrase, and $old(d)$ is the number of pre-existing a-nodes that the new ones compete with. The current XP has a new a-node for each of the $d$ X nodes on the stack, plus a new a-node connecting to the stack (CURR); the current X′ node has a new a-node for each of the $d$ XP nodes on the stack, plus a new a-node connecting to an empty node. Thus, $new(d) = 2(d+1)$. Each of the $d$ X nodes on the stack connects to an existing a-node, plus the root XP node connects to the existing TOS a-node, yielding $old(d) = d+1$ when $d > 0$. If $d = 0$, then there are no existing phrases or attachments, and $old(d) = 0$. Thus, $f(d) = 3(d+1)$ if $d > 0$, and $f(d) = 2(d+1) = 2$ if $d = 0$. We must therefore determine values of $n_s$ and a maximum depth $D$ for which $\sum_{d=0}^{D} {n_s}^{f(d)}$ is a reasonable number of test cases.

In the current implementation of the parser, the state of an a-node can take on the values 0, .5, .6, .7, .8, or .9; the algorithm for computing these particular values will be discussed in Chapter 6. If all of these values are to be tested, then $n_s = 6$; the number of possible initial configurations of the network for a tree of depth $d > 0$ on the top of stack would be $6^{3(d+1)}$. For example, for a tree 5 phrases deep, the number of test cases would be $6^{18}$. This is clearly not a reasonable number of simulations to run for even a single depth value. Given the exponential nature of the function for the number of test cases, it is necessary to decrease the number of state values that are to be tested, as well as the number of a-nodes whose state value is varied.

The precise values given above for the state of an a-node are not so important; what is important is that the state can take on the value 0 to represent an invalid attachment, plus a range of non-zero values to represent the degree of grammatical constraint satisfaction of a valid attachment. In light of this, it was decided to limit the numeric simulations here to vary the state between two values only: 0 for an invalid attachment, and a fixed, non-zero value for a valid attachment. The non-zero value chosen was based on applying the state computation algorithm of Chapter 6 to a typical symbolic configuration. Since an attachment that is valid normally satisfies its grammatical constraints to a fairly high degree, the simulated symbolic features were chosen to reflect this type of situation. Under these conditions, the algorithm yields the value .9 for a complement attachment and .8 for a specifier attachment; these values are .7 and .6 respectively if one of the p-nodes is an empty node.[10] The properties of the stack are such that the value for a stack attachment is always .6. Figure 5.4 illustrates these possible state values in an example network. Since each a-node can now take on either the value 0 or the appropriate non-zero value, the value of $n_s$ has been reduced from 6 to 2.

The decision to limit the variation of state values to a binary choice permits the number of test cases to be further reduced by decreasing the number of a-nodes whose state value may be varied. The only allowable state variation is between two values that represent the choice of a valid or invalid attachment. Under these conditions, it only makes sense to vary the state values of the a-nodes that connect the current input phrase to the tree on the stack; these are the new a-nodes that connect to non-empty, non-stack p-nodes. Pre-existing a-nodes, as well as those that connect to empty nodes, *must* be valid; the stack a-nodes not only are always valid, but always have the same value. Thus, any pre-existing a-nodes, as well as those that connect to the stack or to empty nodes, are given the fixed, non-zero state

[10] After these simulations were conducted, the state computation algorithm was refined in an effort to make it simpler and more consistent. The current algorithm used by the parser would return .5 instead of .6 for a specifier attachment involving an empty node; this change is not large enough to significantly affect the results presented here.

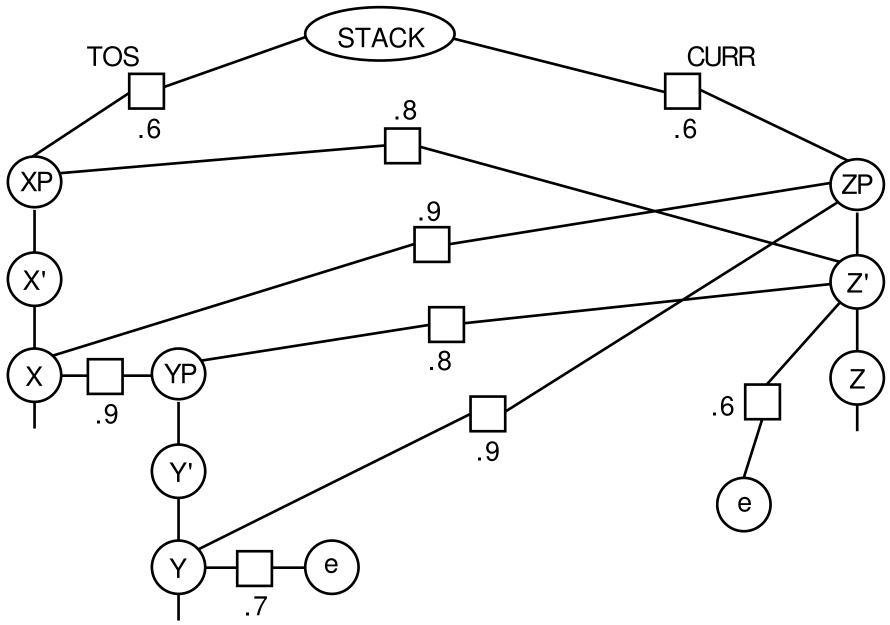


Figure 5.4: Example network with the initial settings of non-zero state values for a-nodes.

values described above. Figure 5.5 shows which a-nodes are given fixed and varied state values in an example network. Since the number of new a-nodes that connect the current phrase to the tree on the top of the stack is $2d$, the number of initial conditions is then ${n_s}^{2d}$ rather than ${n_s}^{3(d+1)}$. So for a partial parse tree of depth 5, the number of test cases has been reduced from $6^{18}$ to $2^{10}$.

In conclusion, the parser will be tested on networks in which the partial parse tree on the top of the stack has depth varying from 0 to 5 phrases. A depth of 0 means that there *is* no partial parse tree on the top of the stack; that is, the current input is the first phrase allocated by the parser. The maximum depth of 5 was chosen because it is large enough to cover a wide range of parsing configurations, and a maximum depth larger than 5 would entail too great a number of simulations to be run. The state values of the a-nodes between the current phrase and the tree on top of the stack will be systematically varied between a state value of 0, and a non-zero state value as outlined in detail above. The results of the $\sum_{d=0}^{5} 2^{2d} = 1365$ numeric simulations will be summarized below.

### 5.3.3 Procedures and Assumptions

Before discussing the actual results, some procedures and assumptions should be explicitly stated. As presented in Figure 5.1, a run of the network consists of allocating an $\overline{\mathrm{X}}$ phrase for the current input token, connecting the new phrase to the existing parsing network with the appropriate attachment nodes, (re)initializing all of the a-nodes that will be competing during the run, and then performing the update/output spreading activation loop until the

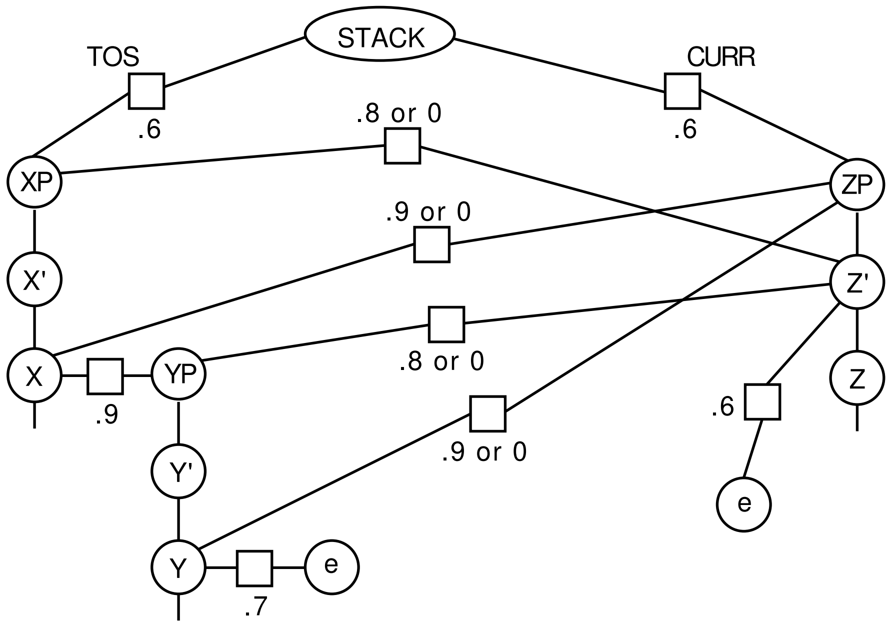


Figure 5.5: Example network with final choice for possible state values for a-nodes.

network reaches an acceptable state representing its current attachment decision. Each numeric simulation of this section imitates a single run of the parser, on a network of a given size, and with symbolic input simulated by a given assignment of state values to the attachment nodes.

For each numeric simulation, a current phrase will be created, as well as a tree on the top of the stack; this tree will have its number of phrases determined by the depth $d$ being tested. All a-nodes needed to represent the branches of the tree on the stack, as well as the "new" a-nodes for the current phrase, will be created and initialized. The state value of each a-node in the simulation will be determined by the systematic variation described above. The network then runs by calling the update and output routines synchronously on all nodes; the simulation terminates when an acceptable state is achieved, or a maximum number of allowable iterations is surpassed. The spreading activation functions, parameters, and thresholds for these numeric simulations are exactly the same as those that were used in the simulations of the complete parser described in Chapter 7.

### 5.3.4 Results

The results of the numeric simulations will now be discussed in terms of each of the four properties presented above: convergence, correctness, reasonableness, and consistency. Because the numeric processing of the network does in fact exhibit these properties to a high degree, it is possible to concisely summarize the results of the 1365 simulations.

### Convergence

The network converges on an acceptable state for over 98% of the initial configurations tested. In the case of the simplest attachment decision that the network faces, in which no new a-nodes have a non-zero state value and the current phrase must push itself onto the stack, the number of iterations to reach an acceptable state is 10–15, for all sizes of the network. The number of iterations for all other cases is consistently in the 30 to 70 range.[11] The number of iterations tends to increase as the number of a-nodes with non-zero state values increases, especially if those a-nodes form multiple compatible sets of attachments. This behavior makes sense: the more correct possibilities that the network is choosing between, the longer it takes to settle on one, particularly in these "canned" simulations where all of the attachment alternatives are equally strong. The exception to this behavior occurs when the "easiest" attachment possible is a valid one. The easiest attachment, for reasons to be discussed below, is the attachment of the XP node of the current phrase to the X node of the lowest phrase in the existing tree; this configuration is shown in Figure 5.6 for a network of depth 3. In these cases, the number of iterations actually *decreases* as the number of valid a-nodes increases; the overwhelming tendency for the easy attachment to dominate is just heightened by the spreading out of available activation to the other nodes.

Another property that affects the number of iterations to convergence is the distance between the valid attachments that the network is choosing between. Phrases that are higher up in the tree on the stack were allocated less recently, and so their activation has decayed more than that of phrases lower in the tree. Attachments to the higher phrases will therefore get less input. One effect of this is that the current phrase prefers lower attachments to the existing tree; this behavior is discussed below under the property of "Reasonableness." The decay of p-nodes not only affects which attachment decision the network makes, but the number of iterations it takes for it to converge on this choice as well. If the lowest possible attachment is competing with an attachment that is only slightly higher in the tree, it will take longer to converge than if the competing attachment is much higher in the tree—the closer attachment gets more input from its p-nodes and therefore competes more strongly. Figures 5.7 and 5.8 illustrate an example of this situation.

The network does not converge on an acceptable state for 17 of the 1365 initial configurations. In each instance, there is at least one p-node that is partially activating multiple a-nodes; that is, the p-node does not exhibit a clear choice of a single attachment to activate. Sixteen of those cases occur in the set of simulations in which the tree on the stack has a depth of 5, which was the maximum depth that was systematically tested. For this size network, there are $2d = 10$ new a-nodes whose state value will be varied across the simulations. In each of the cases of non-convergence, at least 6 of these 10 new a-nodes had a non-zero state value; most of these cases (11 out of 16) had 8–10 new a-nodes with non-zero state values. The high percentage of new a-nodes with non-zero state values means that there are a large number of a-nodes competing for the output from a single p-node. When the

[11]It is perhaps worth noting that this is within the limit of "less than a hundred time steps" discussed by Feldman & Ballard (1982).

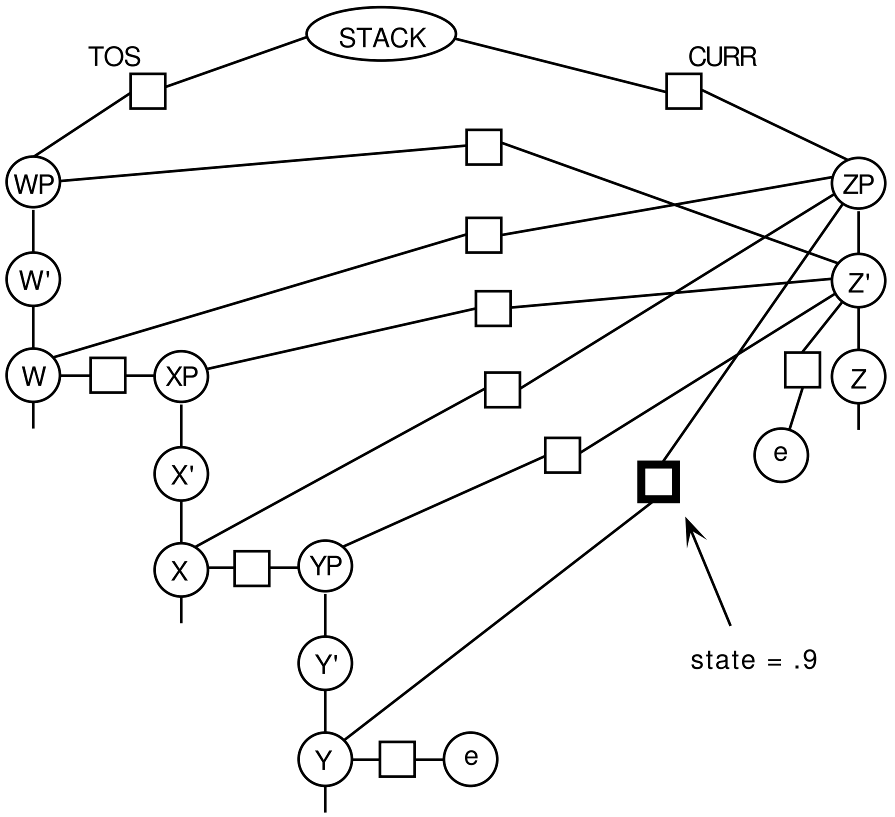


Figure 5.6: If the lowest complement attachment is valid, it is the easiest attachment in the network.

amount of output from a p-node is spread too thinly, the network behavior becomes less predictable. Correcting this behavior would require developing a new activation function for p-nodes that would more effectively avoid this diffusion of output. This approach was not pursued because in real linguistic input these particular configurations are quite improbable. Since it is linguistically implausible that such a large number of attachments between the current phrase and the existing tree will be grammatically valid attachments, any such non-convergent behavior in an actual parse is highly unlikely.

The final case in which the network does not converge is at first more unexpected because it seems like such a simple initial condition. In this configuration, shown in Figure 5.9, the tree on the stack consists of a single phrase, and both new a-nodes between it and the current phrase have non-zero attachments. The fact that the network cannot decide between these two valid attachments is actually not so surprising when considered in light of the effect of decay discussed above. In this case, the two competing attachments are "too close"—because the two a-nodes connect to the same two phrases on the stack, they receive very similar inputs from their p-nodes, and their resulting activation levels are too similar in value for the competitive mechanism to force a choice between them. Adjusting the schedule by which

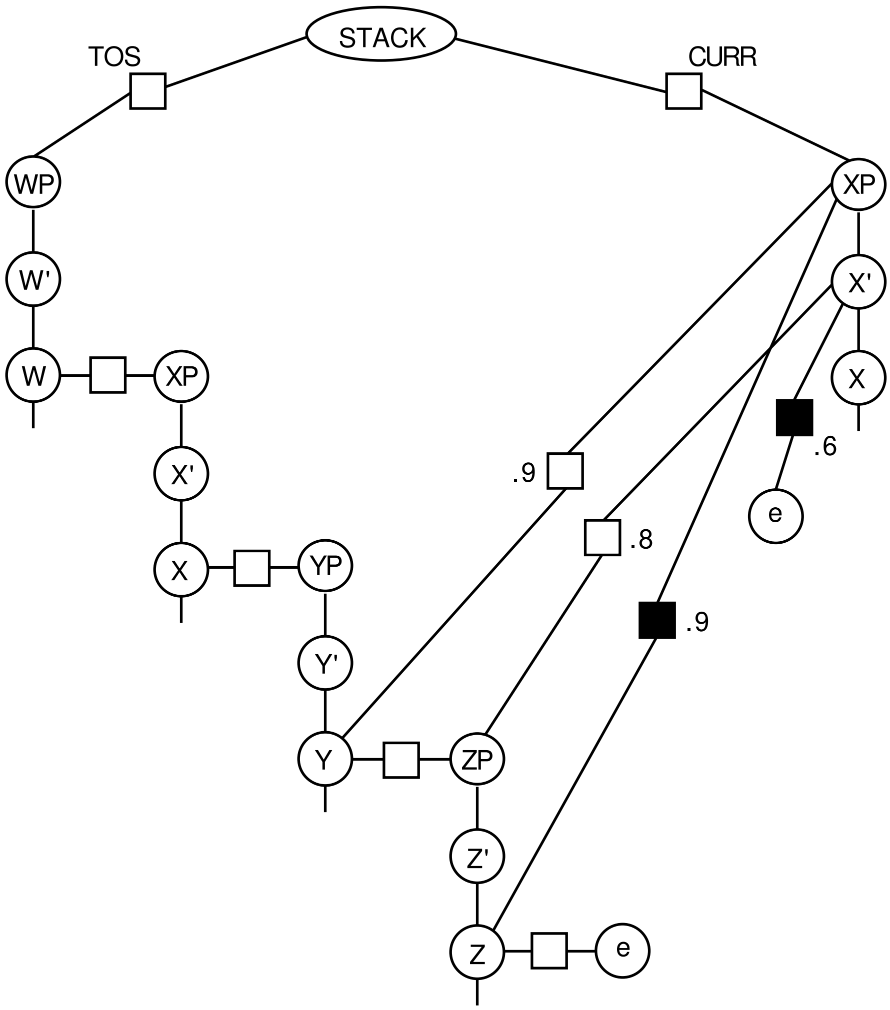

Figure 5.7: Difference in level of competition between higher and lower attachments. (Only a-nodes with non-zero state values are shown.) The network in this figure will settle in 55 iterations, while the similar network in Figure 5.8 takes only 51 iterations.

the competitive constant $v$ is increased corrects this problem. However, since the current schedule works better in a larger number of cases, the change was not adopted in the final parser. In fact, the situation of Figure 5.9 is so linguistically implausible that correcting the problem was not a priority. This configuration can only arise under the following conditions: there are a pair of linguistic categories, X and Y, such that XP can be a specifier of Y$'$ and YP can be a complement of X, the current phrase is of category Y, and there is a single phrase of category X on the stack.

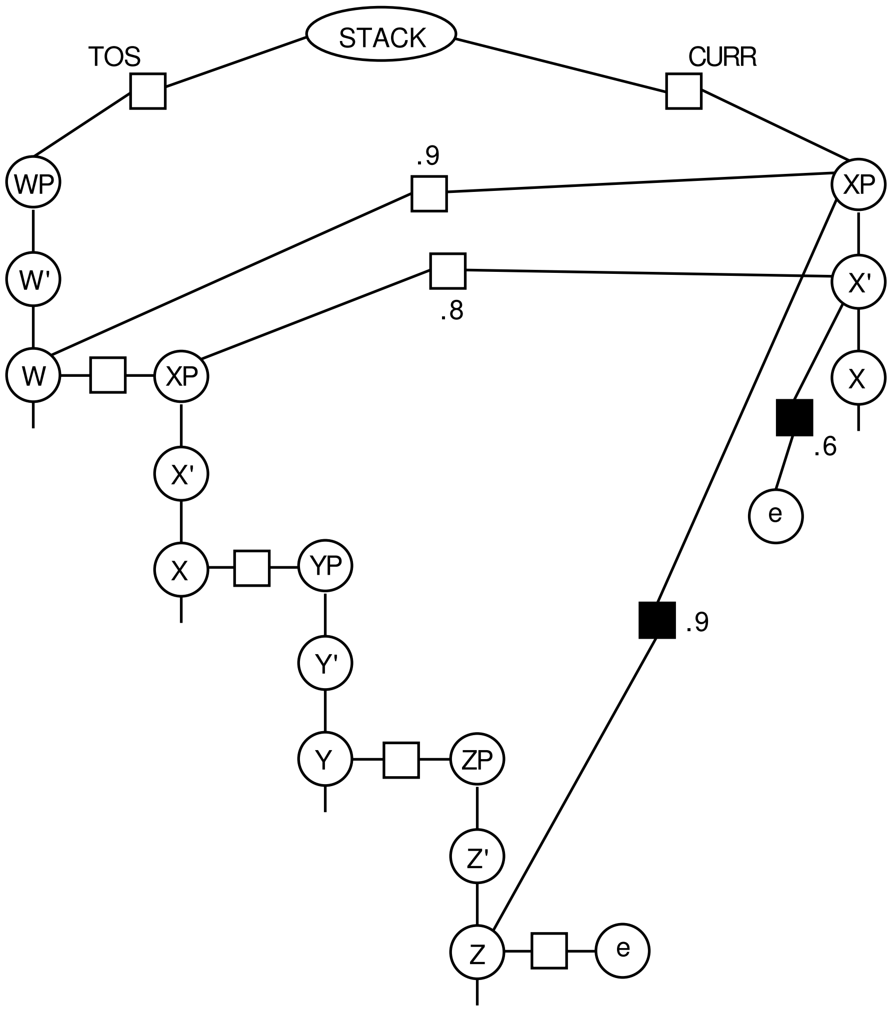


Figure 5.8: Difference in level of competition between higher and lower attachments. (Only a-nodes with non-zero state values are shown.) The network in this figure will settle in 51 iterations, while the similar network in Figure 5.7 takes 55 iterations.

### Correctness

Given that the network converges in almost all situations, the next question is whether the acceptable state at which it arrives represents a valid parse state. Section 4.3 presented the logical attachment possibilities for a correct attachment of the current input phrase to the existing network. Figure 4.32, repeated here as Figure 5.10, visually summarizes these attachment possibilities. For the state of the network to be a correct solution, it must represent one of the sets of active a-nodes described in Figure 5.10. In fact, the final state of every one of the 1348 simulations in which the network converges conform to these rules that specify which a-nodes may be simultaneously active in a valid parse state. The network

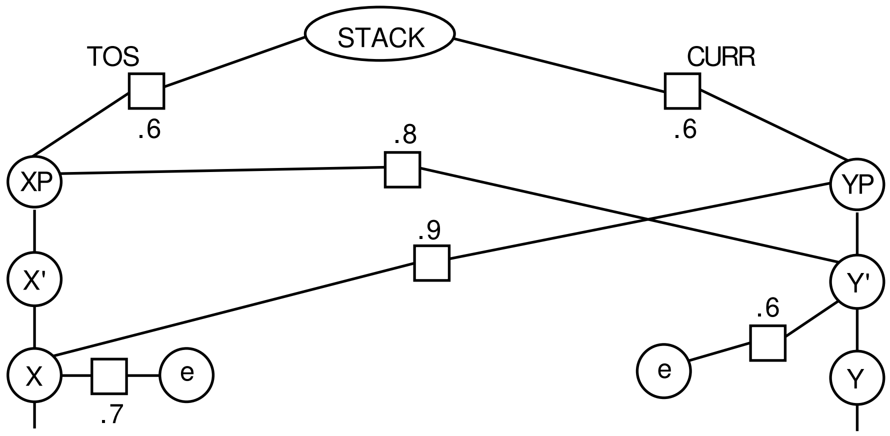


Figure 5.9: Tree of depth one with both a-nodes valid.

never settles in a state in which multiple incompatible attachments are active simultaneously, nor one in which *no* attachment decision is made. Thus, the network not only converges on over 98% of the cases tested, but when it converges, it settles on a globally consistent set of attachments.

**Reasonableness**

The next question is whether the consistent solution that the network chooses is in fact a reasonable action for the parser to take. For example, pushing the current input phrase onto the stack is *always* one of the correct acceptable states at the end of a run of the network. However, it would be completely unreasonable for the parser to parse an input by pushing every phrase onto the stack. Of course if it did this, each individual action would be "correct," but the parser could not arrive at a correct final state. Another factor that must be taken into account is the amount of evidence for one alternative solution over another. Although the network may have several valid solutions to choose from, the choice should not be an arbitrary one; the acceptable state should reflect a reasonable assessment of the relative strengths of the competing attachment possibilities. Some expectations of the network along these lines must be spelled out in order to evaluate the reasonableness of its behavior.

First, if there is a valid attachment of the current input phrase to the existing tree on the stack, that attachment must be made, rather than pushing the current phrase onto the stack. Pushing a phrase onto the stack must be an action of last resort, only occurring when no other attachment possibility is available. This guideline is actually a necessity for correct parsing, since the parser cannot "change its mind" about an attachment not made—once an a-node loses and is de-allocated, it is never created again. The network behaves reasonably in this regard in every simulation—the only time the current phrase is pushed onto the stack is when there is no way to attach it to the existing parse tree.

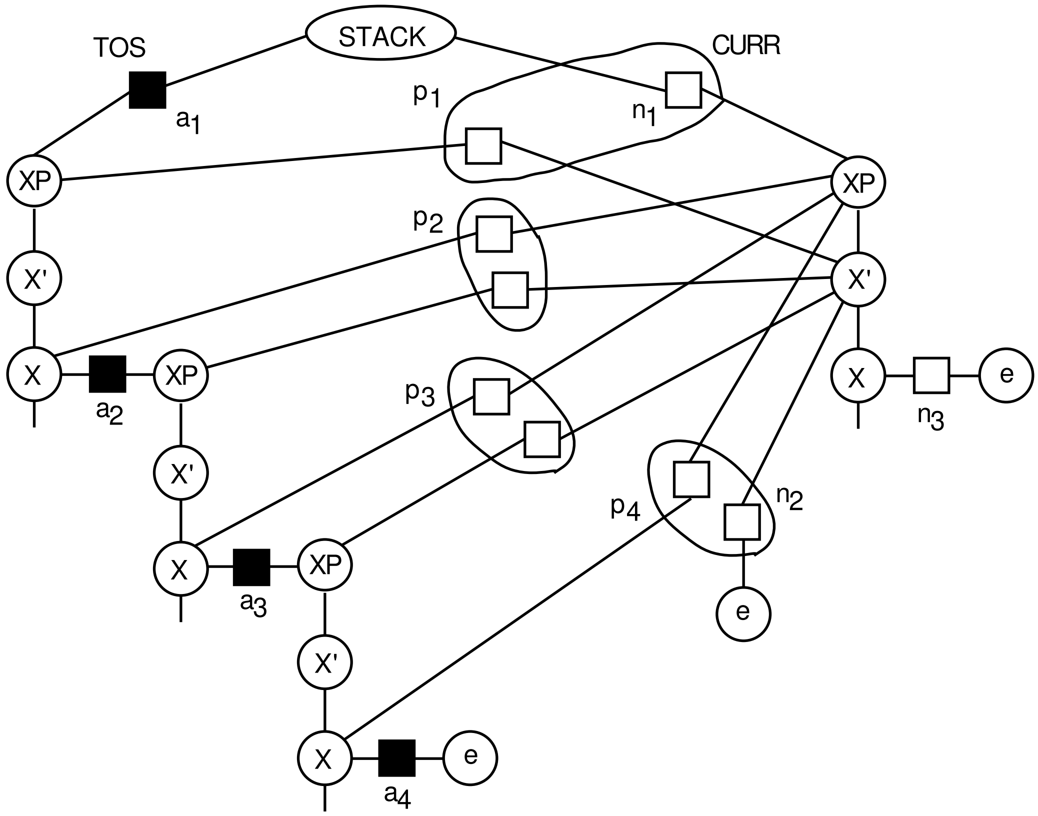


Figure 5.10: Logical possibilities for simultaneous activation of sets of attachment nodes, for an example TOS tree that is three phrases deep. (Only the right edge of the tree on top of the stack is shown; the left side is unaffected by new attachments.) Attachments $a_1$ through $a_4$ were previously activated. If all of these attachments remain active, then $n_1$, $n_2$, and $n_3$ must become active; in this case, no attachments are made between CURR and TOS, and CURR pushes itself onto the stack. To attach CURR and TOS, the following must occur: exactly one of the prior attachments, $a_i$, must become inactive, and the corresponding pair of attachments, $p_i$, must become active. This relationship holds for a TOS tree of arbitrary depth.

This behavior is a direct consequence of two properties of the stack that are independently motivated. One relevant property is the uniform treatment of parse tree attachments and attachments to the stack. Because the stack cannot satisfy grammatical constraints on its a-nodes, their state value is always very low. This low state value has the desirable effect of making these a-nodes less competitive than a-nodes that represent actual parse tree branches. The second relevant property is that the stack's output function must support the simultaneous activation of more than one a-node. Setting the stack's CBSA parameters to be less competitive not only makes the TOS and CURR a-nodes less competitive with *each other* (so that they can be active simultaneously), but has the additional effect of making them less competitive with other a-nodes that represent real attachments. It is interesting to note that the behavior of the stacking mechanism as a last resort action is not directly built into the parser, but rather "falls out" from these independently motivated properties of the stack.

The second guideline necessary to ensure that the individual attachment decisions of the parser are reasonable is that the network must take into account varying degrees of evidence or preference for the different alternatives it is faced with. How well the network weighs evidence is a complex phenomenon to evaluate. One of the motivations for using spreading activation in the first place was that it provides a uniform mechanism for comparing different sources of evidence; however, numerically combining evidence in this way can hide the underlying sources of the effects. In spite of this, the simulation results show clearly that the network combines and compares evidence in a reasonable way. It has already been shown that the a-nodes connected to the stack are less favored than other a-nodes, and that their lower state value is in part responsible for this.[12] While in the 1365 simulations all non-stack attachments of the same type were given the same state value, additional simulations were performed that demonstrated that differing state values (on a-nodes other than TOS and CURR) have a noticeable effect on the attachment decision that is made. The set of 1365 simulations are also highly uniform in that the weights on all of the connections are set to 1.0; this property was varied in further testing as well. The effects of different state values and weights will be exhibited on concrete examples in Chapter 7.

One extremely important source of evidence for an attachment that does show variation in this set of simulations is the amount of input from the phrasal nodes. The simulation results show that the network reasonably and consistently combines and compares this source of evidence. Because p-nodes that are lower in the tree have decayed less, they have more activation to output to their a-nodes. This varying degree of support for an attachment causes the network to consistently choose the lowest of the possible attachments to the parse tree. In every one of the 1342 converging simulations in which there is at least one valid attachment between the current phrase and the tree on the stack, the attachment decision that the network settles on is the lowest set of a-nodes that represent a logical attachment possibility. The strong, consistent recency effect displayed by these results is a direct consequence of the decay mechanism that is necessary for managing the pool of phrasal nodes. Because of evidence in the psycholinguistic literature for the role of recency, Chapter 7 will return to this issue.

### Consistency

The detailed aspects of the behavior of the network that were discussed above hold across all of the simulations in which an acceptable state was reached. The number of iterations to convergence is always within the range of 10–70, and the exact number within this range

[12]Simulations were performed to directly support this claim. For example, as the state value of the CURR a-node is increased, the current phrase becomes more likely to attach to the stack even when there are valid attachments to the existing parse tree.

varies consistently with the size and state settings of the network. The attachment decisions that the network settles on are not only correct in all cases, but further conform to the properties of reasonable behavior outlined above. The consistency of these results is quite important; in spite of the fact that 1365 seems like a large number of simulations, the size of the networks tested is still limited. The fact that these simulations display such consistency gives support to the hypothesis that the same attachment behaviors will hold in larger networks (that is, those with deeper trees on the stack). The parser was in fact tested on various networks with deeper trees on the stack—up to a depth of 8—and these simulations always converged on a solution that was both correct and reasonable.[13] Furthermore, the examples in Chapter 7 that involve trees of depth greater than 5 also behave in a manner consistent with the results presented here.

## 5.4 Limitations of the Numeric Processing

The biggest limitation of the approach is that the numeric functions and parameters lack solid theoretical motivation. In each case, there is justification for the approach that is taken, but the precise settings are largely empirically motivated. However, considering the complexity of the problem, the spreading activation functions that were developed are surprisingly simple and uniform. The phrasal nodes and the stack node use almost exactly the same functions, even though they play a very different role in the processing of the network. Although the attachment nodes use a distinct set of functions that are more complex, they are quite similar to the functions used by Reggia, Marsland, & Berndt (1988). The fact that these functions could be easily adapted across diverse applications supports the view that they capture a generally useful mechanism for weighing evidence from different sources.

Another limitation of the numeric processing is that the network is very sensitive to certain changes in the functions and parameters. Most of the functions and parameters are interrelated, so that it is difficult to isolate the effects of one, or to try out new values for one without adjusting several others. On the other hand, the gross behavior of the network is fairly robust. Under conditions that allow the network to make attachments, the behavior under different parameter settings is quite consistent.

Other issues are not actually problems in principle, but are just limitations arising from practical constraints on the implementation. The parser was not tested on all combinations of state values for the attachments because the number of simulations required is prohibitive. However, Section 5.3.2 motivated the subset of test configurations that was chosen, and the robust behavior of the parser over this large number of simulations is encouraging. One problem that this set of tests does not reveal is that the nodes of the network cannot handle a situation in which there are ties among all of the competing alternatives. If the evidence for the a-nodes that a p-node attaches to is very close to being equal, the p-node will "get stuck" splitting its output evenly among them. A non-deterministic component

[13]These results were not presented here since they were not tests of systematically varying state values and depths, but rather were "spot checks" of plausible linguistic configurations.

needs to be integrated into the output function of p-nodes to enable them to recover from this situation. However, since a tie between competing a-nodes never arose in the set of simulations presented here, or any simulations on actual linguistic input, the development of a tie-breaking scheme was not a high priority.

In conclusion, the results of the numeric processing simulations support the claim that, in spite of the concerns discussed here, the competitive attachment parsing approach has promise. Although the precise numeric formulations were derived experimentally, the functions and parameters are easily explained, and their effects are simple and understandable. Furthermore, they lead to overall behavior that is quite elegant and robust, indicating that their empirical nature, and their sensitivity to change, are not significant drawbacks for the approach.

# Chapter 6

# Symbolic Processing in the Parser

This chapter provides details about the symbolic features and message-passing methods used by the parser.[1] The proposed techniques underlie a novel approach to natural language parsing using constraint-based grammatical knowledge within a massively parallel network. Previous approaches in connectionist parsing have built rule-based knowledge into the structure of the network to determine the space of possible syntactic analyses. With a constraint-based linguistic theory, this straightforward approach of recognizing built-in structure is not available. The techniques described here achieve a direct encoding of a constraint-based syntactic theory using only local operations on very simple symbolic features. A key component of the approach is the development of a message-passing algorithm that utilizes appropriate grammatical restrictions to limit the path of a feature through the network. Using only local distributed communication among the network nodes, the message-passing method successfully ensures that even long-distance syntactic relations are correctly established within the parse tree represented by the network. Thus, the numeric and symbolic techniques of the competitive attachment parser together yield a robust distributed parsing mechanism using only simple and uniform processing nodes.

Section 6.1 gives an overview of symbolic processing in the parsing network. Section 6.2 first briefly introduces the linguistic constraints that the parser encodes, then describes the symbolic features of the different types of nodes in the network. The section explains how the local processing of these features enables the parser to capture the declarative constraints imposed by the linguistic theory. Section 6.3 explains the symbolic output routine in which grammatical knowledge constrains the path of features through the parsing network. It is this novel feature-passing method that ensures that only valid syntactic configurations are established. Section 6.4 concludes the chapter with a discussion of the limitations of the symbolic processing component of the parser.

[1]The communication of symbolic features through the parsing network will be referred to as feature-passing or message-passing. However, it is important to note up front that the parser does not have the symbolic capabilities to support the creation and propagation of general messages or feature structures. In fact, although the symbolic network communication is currently implemented as a feature-passing method, the simplicity of the features that are used here would allow a strict marker-passing implementation, using a small set of fixed markers.

## 6.1 Overview of the Symbolic Processing

Symbolic processing determines to what degree each of the potential attachments in the network satisfies its syntactic constraints. While numeric processing consists of an update/output spreading activation loop, symbolic processing consists of an update/output message-passing loop. As with the numeric functions, the symbolic update and output routines are synchronized so that the update routines are performed simultaneously by all nodes in the network, followed by the simultaneous computation of output. The symbolic updating of a node consists of processing its symbolic input—that is, the features that were just passed to it. The symbolic output routine determines which of its own features, as well as those just input to it, a node should pass on to each of its neighbors.

A feature is passed in a *packet*, which gives the feature and its value, the node that created the packet (the original source node in the message-passing path), and the last node that passed the feature packet (the most recent source node). Whether or not a feature packet is passed from a node to each of its neighbors is determined by the grammatical properties of the feature itself, as well as those of the potential source and destination nodes. There are two major types of features, local and long-distance; this categorization is a direct encoding of the grammatical distinction between local and long-distance relationships in Government-Binding theory. In the parser, local features can be communicated only between the phrasal nodes (p-nodes) of a single $\overline{\mathrm{X}}$ phrase, or between a phrasal node and its attachment nodes (a-nodes), which represent potential sisterhood relations in the parse tree. Long-distance features can be communicated between more distant nodes in the tree along an appropriate path through the parsing network. Different types of long-distance features have different restrictions on their communication from one node to another; this restricted marker-passing process will be discussed in detail in Section 6.3.

In the parsing network, symbolic and numeric computation proceed concurrently, with numeric activation "gating" the communication of symbolic features. A feature packet can be passed to its neighbor only by a node that has sufficient activation—that is, an activation level above a certain threshold $\theta$. This gating mechanism has the desirable effect of focusing symbolic processing within the active portion of the parsing network. In the current implementation, the level of the gating threshold $\theta$ permits any p-node to pass symbolic features, but an a-node must be fully active in order to propagate the features that it receives. Thus, a p-node can initially pass any feature packets among the p-nodes of its own phrase, as well as to its a-nodes. However, an a-node cannot propagate these features further until the numeric activation of the network is focused onto a set of attachment nodes that represent the attachment decisions of the parser. The effect is that only local features can reach their destination before the numeric processing yields an acceptable network state. The local features determine the initial state values of the competing a-nodes, which are crucial in the numeric activation functions. Once the network settles, and the gating threshold for a-nodes is reached, then full symbolic processing (including long-distance features) can occur. Figure 6.1 shows the network processing algorithm of Figure 5.1 with the symbolic components added in.

```
begin {Process Input Sentence}
while there are more input tokens do
        {Each pass through this loop is a run of the network.}
        Get next input token.
        Look up lexical entry for the current input token.
        Allocate an X̄ phrase for the current input token, with
            symbolic features initialized based on the lexical entry.
        Initialize attachment nodes between the current phrase
            and the top of the stack.
        Reinitialize competing attachment nodes.
        until the network is an acceptable state do
                {Each pass through this loop is an iteration of the network.}
                for each node in the network do
                        UpdateSym(node).
                        UpdateNum(node).
                for each node in the network do
                        OutputSym(node).
                        OutputNum(node).
        for 1..n do
                {Continue symbolic processing for a fixed number of iterations.}
                for each node in the network do
                        UpdateSym(node).
                for each node in the network do
                        OutputSym(node).
end {Process Input Sentence}
```

Figure 6.1: The complete iterative algorithm for processing the nodes of the network.

## 6.2 Symbolic Knowledge

The symbolic features used by the parser are a direct encoding of a subset of the grammatical features of Government-Binding theory (GB).[2] GB is divided into a number of "modules" that deal with distinct aspects of the grammar. The constraints imposed by each of these grammatical subsystems interact to determine the validity of a syntactic structure. For example, one module deals with predicate/argument relations, while another determines whether two phrases can be coreferential. While it is infeasible to implement all of GB, the division of knowledge into modules makes it relatively straightforward to implement a

[2]This section presents information about GB that is necessary for understanding the implementation of the competitive attachment parsing model. For a brief introduction to the motivation and development of GB as a linguistic theory, non-linguists may find Sells (1985) helpful.

coherent subset of the theory. The modules that were chosen to be included in the parser are those that determine the basic attachment structure of the input: $\overline{\text{X}}$ theory, $\theta$ theory, Case theory, and (a subset of) the binding theory. A brief overview of the relevant syntactic constraints imposed by each of these modules will be presented in Section 6.2.1, followed in Section 6.2.2 by a description of how this knowledge is represented within the parser.

### 6.2.1 Constraints from the Linguistic Theory

The $\overline{\text{X}}$ module determines the structure of the parse tree, by specifying the content of syntactic phrases and the basic relations that hold between the phrases within a tree.[3] $\overline{\text{X}}$ theory states that every phrase has a *head* (the X node), a *maximal projection* (the XP node), and some number of *intermediate projections* (the X′ node(s)). Some properties of these phrases are very general and may hold for all types of phrases in a language. For example, in English, specifiers always precede the head of a phrase, while complements follow the head. Other properties of a phrase are determined by more specific syntactic information, such as the category of the phrase. For example, categorial information establishes which types of XPs may be in a specifier relation to a given phrase. There are additional properties that are determined by even more specific information given in individual entries in the lexicon. For example, while the types of complements allowed for a phrase in some cases may be determined by the category of the head, many are determined by an explicit *subcategorization* list given in the lexical entry for the head. In addition to the general conditions on phrase structure, $\overline{\text{X}}$ theory imposes the selectional requirement that two potential sister phrases must be of the appropriate categories to be attached to each other.

Theta theory is the module of GB that is concerned with the relations among the phrases that participate in predicates of the sentence. Each predicate has a *theta grid*, which lists the roles, such as Agent, Theme, or Instrument, that phrases may play in an interpretation of that predicate; these roles are called *theta roles*. For example, the verb *know* has a theta grid that includes the theta roles Agent (the one who knows) and Theme (that which is known). The assignment of theta roles is constrained by the Theta Criterion, which is the central principle of Theta Theory. The Theta Criterion states that each phrase in an *argument* position in the parse tree must be assigned exactly one theta role, and that each theta role of a theta grid must be assigned to exactly one *argument* phrase. The category of a phrase determines which of its specifier and complement positions are argument positions. The roles of a theta grid may be assigned to phrases only under certain syntactic configurations. There may be one special theta role, specified as the *external* theta role, that is assigned under the condition of *specifier/head agreement*; this structural relation will be described in Section 6.3. The typical case of external theta role assignment occurs when the verb phrase

[3]GB theorists have attempted to describe many properties of the basic tree structure as side effects of other modules (Abney, 1986; Stowell, 1981). For example, the categories of allowable complements for a phrase may be derivable from the lexical semantic properties of its head. For simplicity of presentation, all basic tree structure and categorial relations will be described here as part of the description of $\overline{\text{X}}$ theory.

assigns the Agent theta role to the subject position of the sentence.[4] All other *internal* theta roles in the theta grid must be assigned from an X node to a sister node in the parse tree.

Case theory imposes an additional constraint on any argument positions that are filled with noun phrases. The Case Filter states that all NP arguments must receive *Case*. Case in GB is an abstract grammatical feature that is a generalization of the overt case markings, such as Nominative or Accusative, that are used in many languages to explicitly mark noun phrases in particular configurations. GB assumes that these are the overt manifestation of the abstract feature Case that is assigned to noun phrase arguments in all languages. An X node of category verb, inflection, or complementizer assigns *structural* Case to an NP in a *head government* relation (to be described below), but the other X nodes assign *inherent* Case to a sister NP.

The binding theory specifies the required structural relationships that must hold between phrases that are coreferential. Such phrases are *coindexed*—assigned the same referential index—under certain structural configurations. The parser implements only the portion of the binding theory that concerns the licensing of traces.[5] Recall that a trace represents a phrase that has been displaced from its underlying position in the sentence. For example, in *Who did Mary kiss e?*, the trace *e* in the object position of *kiss* is a place-holder for the displaced object, *who*. The binding theory states that a trace must be *bound* by being coindexed with an appropriate binder (the displaced phrase). This coindexation relation must meet the structural restrictions of *antecedent government*, which will be described in Section 6.3.[6]

The grammatical information described above comprises the symbolic knowledge that must be represented by the limited symbolic processing capabilities of the parser. Part of the knowledge is the structure of individual $\overline{\text{X}}$ phrases and the direction of attachments between them; these aspects are captured in the parser's mechanism for allocating phrases and attachment nodes. In addition, the four modules of GB contribute to the knowledge of the parser the four explicit constraints shown in Table 6.1.[7] As stated earlier in Section 4.1.2, these constraints in the parsing network are verified at the attachment nodes. Thus, although they are stated as constraints on the properties of XP nodes, the constraints actually do not apply directly to an XP node, but rather to an XP node *in a certain attachment relation* to another node. The next section describes the symbolic features that are needed within

[4]In the parser, the external theta role is passed from the VP to its sister I node, which then assigns the theta role to its subject by specifier/head agreement. This formulation is not standard in GB, and is adopted partly for convenience and partly to demonstrate use of the specifier/head agreement relation. See the discussion in Section 6.3.1.

[5]The parser incorporates the passing of the necessary binding features between an overt phrase and its underlying trace. However, the competitive coindexation nodes that are needed to explicitly represent a binding relation are not included in this version of the parser. See Stevenson (1993a) for a discussion of competitive relations among multiple potential coindexation nodes.

[6]The portion of the binding theory encoded here is an adaptation of the version of the Empty Category Principle (ECP) developed by Rizzi (1990).

[7]The first three of these constraints were presented in Chapter 4.

| Constraint | Definition |
|---|---|
| Selection Constraint | The category of an XP must match the category expected by its sister X or X′. |
| Theta Criterion | An XP in an argument position must be assigned exactly one theta role.[a] |
| Case Filter | An NP in an argument position must be assigned Case. |
| Binding Constraint | Every XP must be bound.[b, c] |

[a]The other part of the Theta Criterion, which states that a predicate assigns each of its theta roles to exactly one phrase, is accomplished indirectly in the parser through constraints on the number of attachments to a node.

[b]A non-empty XP is "bound" by itself.

[c]The binding theory also prohibits *vacuous quantification*; for example, a fronted WH-word, as in a question, must bind some trace. This is accomplished in the parser by requiring quantifiers, such as WH-words, to participate in a binding relation.

Table 6.1: The four syntactic constraints used in the parser.

each node type to support the verification of these constraints. Section 6.3 then presents the message-passing algorithm that ensures that the structural configurations under which the features must be assigned are respected.

### 6.2.2 Symbolic Features and Their Values

The parser relies on a small number of syntactic features to represent the grammatical knowledge described above. The features are implemented as slots that take on a limited range of values specified by the grammatical theory. For example, the feature Case can take on the value Nominative, Accusative, Oblique, or Genitive; a category feature can take on the value of Noun, Adjective, Preposition, Verb, Inflection, Complementizer, or Determiner. The value of a feature can also be a disjunction of a subset of its allowable atomic features, which is indicated by a list; for example, Case = (Accusative Oblique) means that the Case of the node can be Accusative or Oblique.

Section 6.2.1 presented two types of constraints that apply to symbolic features: restrictions on the values of the features, and restrictions on the configurations under which the features can be assigned from one node to another. This section explains how the parser captures the first type of constraint by requiring that certain features of attachment nodes take on particular values. The symbolic features of each node type will be described, including how their values are initialized and updated during the parse. Most feature slots of a node are filled with atomic values or lists of these atomic values; these slots will be referred to as the *simple* features of a node. In addition to its simple features, one of the most important symbolic feature slots of a node holds a list of *feature packets* (feature slot/value pairs) that the node potentially passes to its neighbors. Part of the update procedure for an attachment node is to determine how well its grammatical constraints are satisfied, based on the features that are input to it from its phrasal nodes. The final step of the update process for a node of any type is to transfer to its output list the symbolic features that it received during the previous iteration of the network, so that they can be further propagated during the current iteration.

### Phrasal Nodes

Symbolic features in the parsing network originate in the lexicon. Lexical entries are defined within an object-oriented network of categorial features, in which each object determines the values of some subset of symbolic features. The set of categorial features and their effects on the settings of other symbolic feature values are shown in Table 6.2. The leaf objects in the object-oriented network are the syntactic categories Noun, Adjective, Preposition, Verb, Inflection, Complementizer, and Determiner. The settings of the categorial features for each of the syntactic categories is shown in Table 6.3.[8] A lexical entry is defined as an instance of a category leaf object within the feature network, and inherits most of its symbolic feature settings from its ancestors. The only information that must be explicitly given in the lexical entry is that which is idiosyncratic to the particular word or morpheme being defined. In fact, for now, the only features determined by the individual lexical entries are the number of a noun (singular or plural), the subcategorization of a verb (a list of possible categories of its complement), and the tense of an inflection (past or present). All other features are inherited from the categorial network.

The symbolic features of the p-nodes constituting an $\overline{\mathrm{X}}$ phrase are determined by the lexical entry of the head of the phrase. A p-node has only three simple features, plus the slot for its list of potential output feature packets. Each of the three simple features is initialized when the p-node is created, and remains unchanged throughout the parse. One simple feature, with value TRUE or FALSE, tells whether or not the phrase is a *lexical* category. The lexical categories are, loosely, those with more semantic content (noun, adjective, preposition, and verb), while the non-lexical (or *functional*) categories are those that play a more purely syntactic role (inflection, complementizer, and determiner). The second simple feature, which also takes on the value TRUE or FALSE, indicates whether or not the sister of an X′ node is an argument position. The argument positions are the complements of lexical X nodes, and the specifiers of N′, I′, and C′. Both of these features are used in determining whether certain feature packets that land at a node can be propagated beyond it. The third feature is a list of the frequency information for the allowable categories of the sister of the p-node in the parse tree. The frequency list is used to set the weights on the links between an X or X′ node and the a-nodes that it connects to.[9]

[8]The breakdown of the syntactic categories into these features and the meanings of the features are adapted from proposals within GB theory, but the particular formulation was developed in the research here.

[9]Chapter 7 will demonstrate how setting weights based on the frequency information leads the parser to

| Feature | Setting | Meaning |
|---|---|---|
| Nominal | + | The sister of an X′ node is an argument. |
| | − | The sister of an X′ node is not an argument. |
| Verbal | + | An X node assigns structural Case.<br>The sister of an X node is selected. |
| | − | An X node assigns inherent Case.<br>The sister of an X node is not selected. |
| Lexical | + | The sister of an X node is an argument.<br>An X node assigns a theta role to its complement. |
| | − | The sister of an X node is not an argument. |
| Case Assigner | + | An X node can directly assign Case. |
| | − | An X node cannot directly assign Case.[a] |
| Degenerate | + | Only an XP node is projected from the input. |
| | − | A full X phrase is projected from the input. |

[a]This feature refers to the necessity of a noun or adjective in English to be accompanied by the Case-assigning morpheme *of* in order to discharge Case to its complement, as in *queen of England* or *proud of Sara*.

Table 6.2: Categorial features and their effects on the settings of other symbolic feature values.

| Syntactic Category | Categorial Feature | | | | |
|---|---|---|---|---|---|
| | Nominal | Verbal | Lexical | Case Assigner | Degenerate |
| Noun | + | − | + | − | − |
| Adjective | − | − | + | − | − |
| Preposition | − | − | + | + | − |
| Verb | − | + | + | + | − |
| Inflection | + | + | − | − | − |
| Complementizer | + | + | − | + | − |
| Determiner | − | − | − | − | + |

Table 6.3: The syntactic categories with their settings for each of the categorial features defined in Table 6.2.

The features included on the output feature packet list of a p-node are those that are required to support the constraint-checking carried out by attachment nodes. To enable the checking of the selection constraint, each p-node outputs the category of itself and/or a list of the possible categories of its sister in the parse tree. The list of possible categories may include the special value NONE, which indicates that the p-node is allowed to have no sister. Since a p-node must always activate exactly one attachment, this is accomplished by activating an attachment to an empty node and assigning it the category NONE. To support the verification of the Theta Criterion, each X node outputs whether or not it assigns a theta role to its sister XP.[10] The Case Filter depends on knowing from X nodes which Case they assign, if any, and from XP nodes which Case they expect.[11] The binding theory directly relies on two features, one from an XP node that specifies whether or not that phrase is bound, and one from an X or X′ node that specifies whether or not its sister in the parse tree is an argument position. The latter feature is used in determining an appropriate binder for an empty node. A final feature from an X or X′ node indicates whether or not its sister is *selected*; the property of being a selected position is, like the property of being an argument position, determined by details of GB that are irrelevant here. The only selected positions are the complements of V, I, and C nodes (Cinque, 1990; Rizzi, 1990). This is again a feature that is used solely to determine how features can be propagated through the network.

Table 6.4 summarizes the features initially on the output list of each type of p-node (XP, X′, and X), and sample values for nodes of different categories. Like all network processing nodes, a p-node updates this output feature list at each iteration, by adding to it the symbolic feature packets that it just received on its input list.

### Empty Nodes

Chapter 5 noted that empty nodes are a subtype of p-node. In fact, they are further specified as a subtype of XP node, and inherit the symbolic features of an XP. The only simple feature that is applicable to an empty node is the one that states whether or not a node is lexical (in the technical sense described above); this feature is FALSE for all empty nodes. Like all other nodes, an empty node has an output list of feature packets, and the features included on this list are also inherited from the XP node object. When the empty node is first created, the value of the output feature is-bound is FALSE. Because the empty node is not yet bound, much of its remaining symbolic information is undetermined. The XP-category, X′-category, and has-Case output features are assigned the special value UNSPECIFIED to indicate that a more precise statement of the value is unknown. This value allows constraints that refer to category and Case values to assume that an empty node may take on values that are relevant

take lexical preferences into account in disambiguation.

[10]An XP node might also assign the *external* theta role of the predicate of the head of the phrase; for example, the VP assigns the external theta role of the verb to the subject of the clause.

[11]In English, the latter is usually not relevant, since only pronouns restrict which Case they can be assigned; for example, the pronoun *we* must receive Nominative Case, and *us* Accusative or Oblique.

| XP Output Features | Noun | Verb | Infl | Comp |
|---|---|---|---|---|
| XP-category | N | V | I | C |
| X′-category | (N I C) | (NONE) | (NONE) | (NONE)[a] |
| assigns-external-theta | FALSE | TRUE | FALSE | FALSE |
| has-Case | ANY[a] | NONE | NONE | NONE |
| is-bound | TRUE | TRUE | TRUE | TRUE |

| X′ Output Features | Noun | Verb | Infl | Comp |
|---|---|---|---|---|
| XP-category | (D A NONE) | (NONE) | (N I C) | (WH C NONE) |
| X′-category | N | V | I | C |
| sister-is-argument | TRUE | FALSE | TRUE | FALSE |
| sister-is-selected | FALSE | FALSE | FALSE | FALSE |

| X Output Features | Noun | Verb | Infl | Comp |
|---|---|---|---|---|
| XP-category | (N)[a] | (N C)[a] | (V) | (I) |
| assigns-theta | TRUE[a] | TRUE[a] | FALSE | FALSE |
| assigns-Case | GEN | ACC | NONE | NOM |
| sister-is-argument | TRUE | TRUE | FALSE | FALSE |
| sister-is-selected | FALSE | TRUE | TRUE | TRUE |

[a] Example value only; the precise value will depend on the lexical entry.

Table 6.4: Initial output features of p-nodes, with sample values for XP, X′, and X nodes of various categories: Noun, Verb, Infl (inflection), and Comp (complementizer).

to the constraint. These output features might change during the parse, since the features of the empty node will be determined by the corresponding features of its binder. When a valid binding relation is established for the empty node, its is-bound output feature is set to TRUE, and its XP-category, X′-category, and has-Case output features will be updated to be equal to those of the binder.[12] The output feature assigns-external-theta is always FALSE for empty nodes, because there is no head of the phrase from which to assign a theta role. Table 6.5 summarizes the initial values of the output features of an empty node.

### The Stack Node

Just as an empty node is further specified symbolically as an XP node, the stack node is a subtype of p-node that is further specified as an X node. Analogous to an X node activating

[12] In the results described in this dissertation, every empty node will remain unbound throughout the parsing process. See Stevenson (1993a) for a presentation of how the competitive attachment model supports the binding of empty nodes in a manner that accounts for the psycholinguistic data on filler/gap processing from Carlson & Tanenhaus (1988), Frazier (1987), and Stowe (1986).

| Empty Node Output Features | Initial Values |
| --- | --- |
| XP-category | UNSPECIFIED |
| X′-category | UNSPECIFIED |
| assigns-external-theta | FALSE |
| has-Case | UNSPECIFIED |
| is-bound | FALSE |

Table 6.5: Initial output features of empty nodes.

| Stack Node Output Features | Initial Values |
| --- | --- |
| XP-category | UNSPECIFIED |
| assigns-theta | FALSE |
| assigns-Case | NONE |
| sister-is-argument | TRUE |
| sister-is-selected | N/A |

Table 6.6: Initial output features of the stack node.

a complement attachment to an XP, the stack node activates attachments to XPs that push themselves onto the stack. However, the stack is purely a computational mechanism, and has no meaningful identity within the linguistic theory; thus, it is a degenerate X node with regard to its symbolic features. Its features are given initial values reflecting this, and remain unchanged during the parse. The simple symbolic features of the stack state that it is not lexical, and that it assigns equal frequency to the categories of nodes that can attach to it. The features on its output list are set as follows. The XP-category feature has the value UNSPECIFIED (any phrase can push itself onto the stack), assigns-theta is FALSE, and assigns-Case is NONE. However, since the final parse tree on the stack is presumably input to semantic and discourse processing mechanisms, the sister-is-argument output feature of the stack node is set to TRUE. This extends the notion of argument position from GB to encompass arguments in the discourse. The value of the final output feature of X nodes, sister-is-selected, is irrelevant for the stack because it is used only in determining how to pass features through the tree.[13] Table 6.6 summarizes the initial values of the output features of the stack node.

[13]Since the stack is at the top of the tree, features from within the tree cannot be passed up beyond it; since the stack has only sister features to communicate, features from it need not be passed down into the tree beyond its a-nodes.

### Attachment Nodes

The important symbolic work of a p-node is to communicate features to the a-nodes that represent its potential attachments. Thus, the list of its potential output feature packets is the seat of the real information of a p-node. It is the a-nodes that process these features to determine the syntactic validity of any given attachment. Not surprisingly, the a-nodes are symbolically complementary to the p-nodes: they create no output feature packets of their own, but they have a large number of feature slots for recording the values of all of the symbolic features relevant to an attachment. The features that are created and output by the p-nodes are combined by the a-nodes to determine to what degree its grammatical constraints are satisfied. The set of symbolic features of a-nodes thus must be the union of the features communicated by the p-nodes:

- XP-category
- X′-category
- receives-theta[14]
- has-Case
- receives-Case[15]
- is-bound
- sister-is-argument
- sister-is-selected

When an a-node is created, all of its features are initialized to the default value ANY, and its output feature packet list is empty. To update its symbolic information, an a-node processes any features that were input to it during the previous network iteration, recomputes its numeric state value based on its updated symbolic information, and, like all nodes, transfers its list of input features to its output feature list. The remainder of this section will describe the processing of input features and the recomputation of the state value.

When an a-node receives an input feature, it unifies its current value for that feature with the new value. Unifying the new value with the initial value ANY gives the new value as the result. Since GB requires that the theta and Case assigners for a node be unique, assigns-theta and assigns-Case features can be assigned only once. This is ensured by having those features able to unify successfully only with the value ANY; thus, once one value has been assigned, additional attempts to assign a new value will not unify with the current value. For all other features, the current and new values unify successfully if they are the same atomic value, or if one is an atomic value and the other is a list of atomic values that contains the first (the result being the atomic value). The only exception is unification involving the special value UNSPECIFIED. If one of the terms being unified is the value UNSPECIFIED or a list containing the value UNSPECIFIED, the result of unification is the value UNSPECIFIED. If the

[14]The receives-theta feature at an a-node corresponds to the theta assignment features of p-nodes.

[15]The receives-Case feature at an a-node corresponds to the Case assignment features of p-nodes.

current and new values for a feature do not unify, then the feature is given the value INVALID and the a-node becomes inactive. Table 6.7 shows an example of an a-node representing a complement attachment, and the result of unifying the input features from its X and XP nodes with its default values. Table 6.8 similarly shows an example of a specifier a-node, and the result of unifying the input features from its X′ and XP nodes.

After an a-node updates its feature settings by unifying its input features with its current feature values, the a-node then updates its numeric state value by applying the constraint algorithm shown in Figure 6.2. The algorithm encodes the four constraints on XP feature values that were presented in Table 6.1 on page 92. In order to arrive at the new state value for an a-node, the constraint-checking algorithm takes the simple approach of assuming a high initial state value and then deducting a constant value for each constraint that is unsatisfied. For example, given the constants currently used in the algorithm, the a-node of Table 6.7 computes its state value to be 0.9, because none of its constraints are violated. The a-node of Table 6.8, on the other hand, currently violates both the Theta Criterion and the Case Filter, and thus determines its state value to be 0.7.

Note that the computation of a constraint violation is conservative when given indeterminate values. This is shown explicitly for the Selection Constraint: if the category of the XP is UNSPECIFIED (that is, unknown), then the algorithm assumes that the attachment might not satisfy the constraint. The Case Filter is also conservative, since a value of UNSPECIFIED can "match" the value N. Since only the stack node and empty nodes can give an XP-category feature the value UNSPECIFIED, the effect of this conservative approach is to decrease the state value of an attachment to the stack or to an empty node (in the latter case, until the empty node is bound).

### Summary

Recall that in Chapter 5, phrasal nodes and attachment nodes were shown to play complementary roles in the numeric processing of the parsing network. A phrasal node distributes the numeric evidence that indicates how strongly it prefers each of its potential attachments. An attachment node combines its numeric input to arrive at its activation level, which indicates how strongly that attachment is preferred to be part of the parse. Here the two node types have been shown to play the corresponding complementary roles in the symbolic processing of the parser as well. A phrasal node creates and outputs features that are used to determine the grammaticality of potential attachments. An attachment node unifies the features it receives and applies a constraint-checking algorithm to compute its state value, which indicates the degree to which its grammatical constraints are satisfied.

The constraint-checking process at the attachment nodes implements one type of constraint imposed on symbolic features by the grammatical theory, achieving part of the goal of distributed, constraint-based parsing. The next section describes how the message-passing algorithm of the parser enforces the other type of grammatical constraint, which restricts the structural configurations under which features can be assigned.

| A-Node Features | Input Values from V Node | Input Values from NP Node | New Values |
|---|---|---|---|
| XP-category | (N I C) | N | N |
| X′-category | | (I) | N/A |
| receives-theta | TRUE | | TRUE |
| has-Case | ACC | (NOM ACC OBL) | ACC |
| receives-Case | TRUE | | TRUE |
| is-bound | | TRUE | TRUE |
| sister-is-argument | TRUE | | TRUE |
| sister-is-selected | TRUE | | TRUE |

Table 6.7: The sample a-node represents a complement attachment between the verb *know* and the NP *Sara*. The initial value of each of its features is ANY. Since it is a complement a-node, the X′-category feature is not applicable.

| A-Node Features | Input Values from I′ Node | Input Values from NP Node | New Values |
|---|---|---|---|
| XP-category | (N I C) | N | N |
| X′-category | I | (I) | I |
| receives-theta | | | ANY |
| has-Case | | (NOM ACC OBL) | (NOM ACC OBL) |
| receives-Case | | | ANY |
| is-bound | | TRUE | TRUE |
| sister-is-argument | TRUE | | TRUE |
| sister-is-selected | FALSE | | FALSE |

Table 6.8: The sample a-node represents a specifier attachment between the I′ of a phrase headed by the tense morpheme *to* and the NP *Sara*. The initial value of each of its features is ANY. Note that since an untensed inflection phrase cannot assign Case, the receives-Case feature retains the value ANY, indicating that the NP does not yet receive Case in this attachment relation. Also, the NP does not yet receive a theta role, since that must be passed to it from the VP.

```
begin {Constraint-Checking and State Computation}

if the a-node has any invalid features then
   Set the state value of the a-node to 0.
   exit
endif

Set the state value of the a-node to 0.9.

if XP-category = UNSPECIFIED then        {Selection Constraint}
   Decrement state value by 0.1.
endif

if sister-is-argument = TRUE             {Theta Criterion}
   and receives-theta ≠ TRUE then
   Decrement state value by 0.1.
endif

if sister-is-argument = TRUE             {Case Filter}
   and XP-category = N
   and receives-Case ≠ TRUE then
   Decrement state value by 0.1.
endif

if is-bound = FALSE then                 {Binding Constraint}
   Decrement state value by 0.1.
endif

end {Constraint-Checking and State Computation}
```

Figure 6.2: The grammatical constraint-checking algorithm that determines the state value for an a-node. Since the category value UNSPECIFIED can match any category, "XP-category = N" is true when the feature has the value N or UNSPECIFIED.

## 6.3 Restricted Feature-Passing

Section 6.2.2 showed how certain constraints from GB are encoded as simple equality tests on the values of the attributes of an attachment node. However, to ensure grammaticality, it is clearly not sufficient to pass features indiscriminately through the network and then run the constraint-checking algorithm at the a-nodes. The constraint-checking algorithm can ensure that each XP in a given attachment relation has certain required features, but it cannot ensure that the attachment node received those features in an appropriate manner. Nor is it sufficient to control the passing of features purely by distance or degree of activation, as in many previous marker-passing approaches (for example, Charniak, 1986; Hendler, 1987), since it is the *structure* of the path between two nodes that must constrained, rather than its length. Approaches that constrain paths through a network according to a regular expression specification are also inappropriate (Norvig, 1989; Yu & Simmons, 1990); here, dynamic properties of both nodes and links must be taken into account in determining a valid path through the network.[16]

The challenge then is to verify the structural configurations from GB that must hold between two nodes in a given syntactic relation—for example, that structural Case is assigned within a head government relation. Features assigned under sisterhood can be easily made to obey this constraint by prohibiting those features from being further propagated after leaving their source node. However, the other syntactic relations involve longer-distance structural configurations that are not directly described as relations between neighboring nodes in the parse tree. Since a connectionist network has no global perspective on its own structure, even non-local parsing decisions such as these must be made solely on the basis of local communication.[17]

A solution for achieving local verification of structural constraints in the parser exploits two facts: (1) A syntactic relation between two nodes involves features that must be assigned or shared between them; and (2) Features passed between nodes must travel through the network, which is a direct representation of the parse tree structure. Thus, the parser can enforce structural constraints on a syntactic relation by ensuring that the feature-passing path along which the relevant features are passed conforms to those structural restrictions. Because the network is limited to local interactions among the nodes, this must be achieved by constraining *each segment* of the feature-passing path to adhere to the grammatical restrictions that apply to the particular feature being communicated. The success of the approach relies on the insight that the structural constraints on any relation between two nodes in a parse tree can be broken down into local components. The set of local restrictions can then be verified entirely between pairs of directly neighboring nodes along the path in the tree between the two dependent nodes.

Section 6.3.1 presents the structural configurations from GB that must be verified in

[16]The solution adopted here could be viewed as a generalization of the regular expression approach, using a distributed definition of the allowable path components.

[17]The limited symbolic capabilities of the parser prevent it from building in global information; nodes are unable to create feature structures that could encode the history of a feature-passing path.

| Module | Feature | Structural Constraint |
|---|---|---|
| Theta Theory | internal theta role | sisterhood |
| | external theta role | specifier/head agreement[a] |
| Case Theory | inherent Case | sisterhood |
| | structural Case | head government |
| Binding Theory | binding of a trace | antecedent government.[b] |

[a] The use of specifier/head agreement for external theta role assignment is explained in the text.

[b] The antecedent government relation is further specified depending on the type of movement that gave rise to the trace; this is explained in the text.

Table 6.9: Structural constraints on feature assignment.

order to support the grammatical constraints implemented by the parser. It is worth noting that while the parser encodes only a portion of the linguistic theory, this subset of structural restrictions covers the major syntactic relations of GB. Section 6.3.2 follows with a description of the message-passing algorithm of the parser, which ensures that these structural constraints are upheld.

### 6.3.1 Structural Constraints from GB

Section 6.2.1 noted that different grammatical features must be assigned from one node to another under different structural configurations, as summarized in Table 6.9. Detailed explanations of each of the structural relations relies on an exposition of GB theory that is beyond the scope of this discussion. Furthermore, GB is an evolving theory in which the precise set of definitions underlying these relations is continually being refined. It is not the intention of the research here to propose the definitive version of the theory that should be implemented. Rather, the goal is to demonstrate the ability of the computational techniques to implement a constraint-based theory of this style. Hence, a brief definition of each structural relation will be given, along with an illustration of the typical configuration(s) within which it applies. Note that the definitions of the government relations are derived from the proposal of Rizzi (1990), because it provides the most uniform account of these long-distance relations, and therefore supports a straightforward implementation. However, the success of the parser's approach does not rely on Rizzi's precise linguistic arguments.

The government relation plays a key role in the linguistic theory, reflected in the name of the theory itself, "Government-Binding." Since government underlies both the head government and antecedent government relations, its definition will be given first.[18]

[18] Here I have extracted the common pieces of Rizzi's head government and antecedent government definitions and labeled them *government*, although Rizzi (1990) does not separate them out in this way.

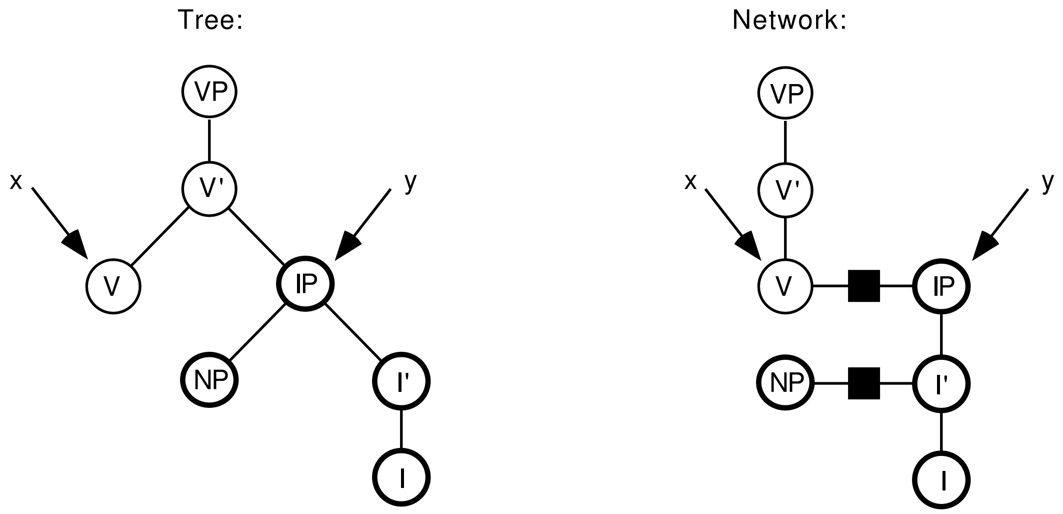


Figure 6.3: $x$ c-commands $y$ and every node below it; c-commanded nodes are highlighted.

**Government:** $x$ governs $y$ iff
i. $x$ c-commands $y$,[19]
ii. no barrier intervenes, and
iii. Relativized Minimality is respected.

The three conditions on government are defined as follows; the definitions of barrierhood and minimality are taken directly from Rizzi (1990).

**C-command:** $x$ c-commands $y$ iff every node that dominates $x$ dominates $y$.[20] (See Figure 6.3 for an illustration of the c-command relation.)

**Barrier:** $x$ is a barrier iff it is not selected as a complement of a verbal head.[21]

**Relativized Minimality:** Relativized Minimality is respected in the relation $x$ $\alpha$-governs $y$ only if there is no $z$ such that
i. $z$ is a typical potential $\alpha$-governor for $y$, and
ii. $z$ c-commands $y$ and does not c-command $x$.

The term "$\alpha$-government" ranges over head government and three types of antecedent government. Informally, the Relativized Minimality condition states that a node $x$ can only

[19]Rizzi (1990) uses *m-command* for head government and c-command for antecedent government. For simplicity, c-command is used for both types of government here. The extra condition of not being able to head govern through a lexical projection must then be added to rule out certain problematic cases that arise from using c-command instead of m-command for head government. This restriction on head government is an adaptation of proposals from Chomsky (1986b) and Rizzi (1990).

[20]Note that this definition, although a common version of c-command, differs slightly from that of Rizzi (1990).

[21]The verbal heads are V, I, and C.

have a certain government relation to another node $y$ if there is no closer node $z$ that can have that relation to $y$. For example, the binder of a trace must be the node at a minimal distance from the trace that satisfies the other syntactic constraints on the binding relation.

Given the above statement of the government relation, the head government and antecedent government definitions are quite simple.[22]

**Head Government:** $x$ head governs $y$ iff

i. $x$ is a head,[23] and

ii. $x$ governs $y$.

**Antecedent Government:** $x$ antecedent governs $y$ iff

i. $x$ and $y$ are coindexed, and

ii. $x$ governs $y$.

There are three subtypes of antecedent government: A-antecedent government for traces of NP-movement (as in passive and raising); $\bar{\text{A}}$-antecedent government for traces of WH-movement (as in WH-questions); and X-antecedent government for traces of head movement (that is, movement of the head of a phrase).

Figures 6.4, 6.5, and 6.6 illustrate the typical government configurations yielded by the above definitions. (Since X-antecedent government isn't currently used in the parser, it is not shown.)

The final relation of specifier/head agreement is much simpler than the government relations. Specifier/head agreement simply states that the head of a phrase (the X node) and the specifier position of the phrase must share certain features. For example, this relation is usually used to account for subject/verb agreement in English: the person and number features of the specifier of the inflection phrase and the corresponding features of the I node must agree.[24] Here specifier/head agreement is used to assign the external theta role from the I node (which it receives from its sister VP) to the phrase in subject position. This is a non-standard explanation of the transmission of the external theta role. As noted in Section 6.2.1, this is not intended to make a claim about the linguistic theory, but rather is meant to demonstrate the ability of the parser to implement specifier/head agreement within the framework developed for the other long-distance relations it uses. The other possibility for implementing external theta role transmission is to use a version of head government based on *m-command*. This would also be easily specified within the parser's message-passing algorithm.

---

[22]Again, the definitions are from Rizzi (1990), with the modifications to refer to *government*.

[23]Technically, one of the following heads: A, N, P, V, Agr, or T. For the purposes here, the fact that it is a head—that is, an X node—is enough. In the parser, only X nodes of the appropriate types have head government features to assign.

[24]There are a number of formulations of how these features are communicated between the verb and the I node; these issues are not addressed here.

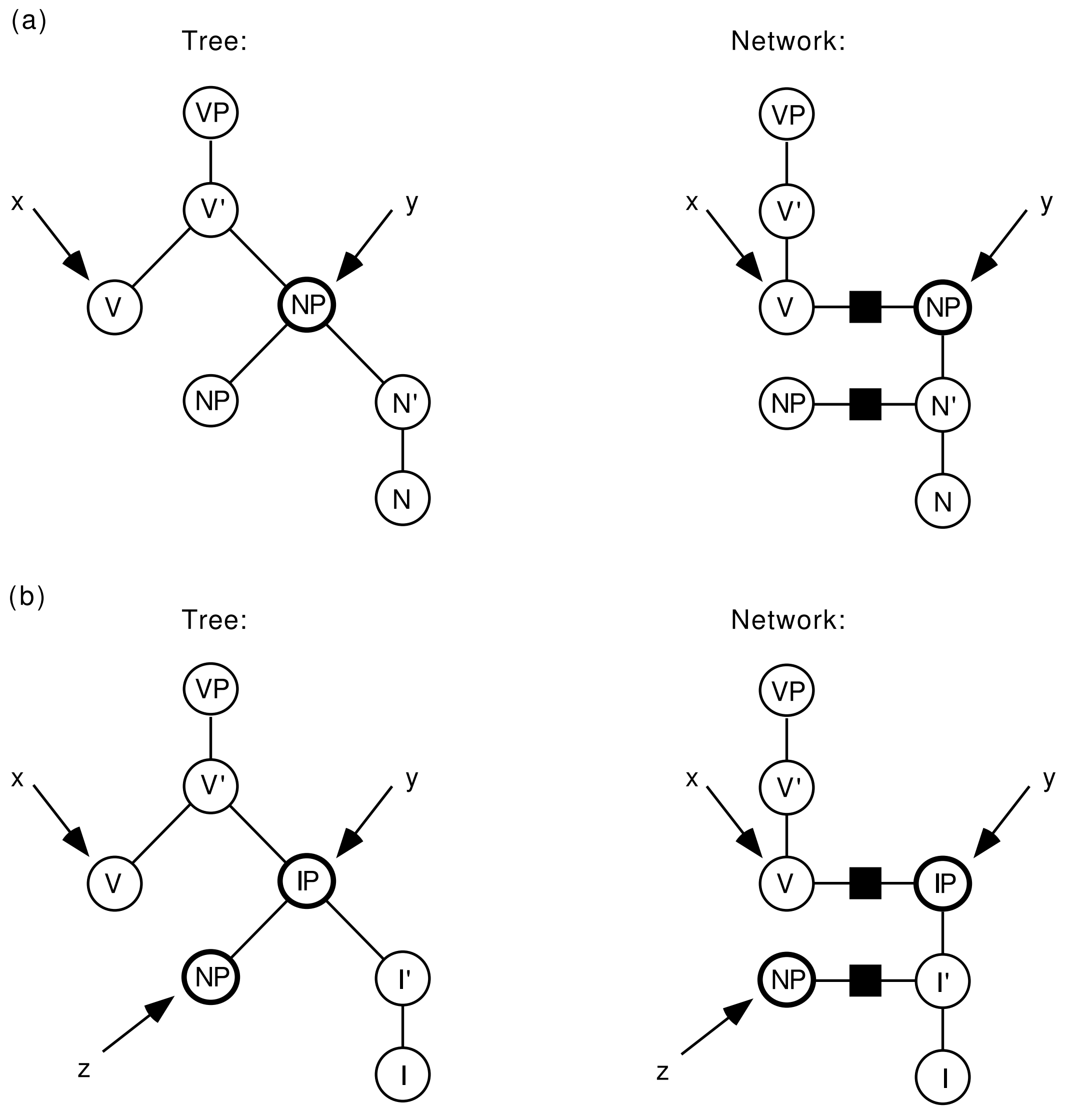


Figure 6.4: $x$ head governs $y$ in (a), and $x$ head governs both $y$ and $z$ in (b). Head-governed nodes are highlighted.

### 6.3.2 Message-Passing Restrictions

The feature-passing restrictions of the parser were derived directly from the linguistic theory, by analyzing the structural constraints described above into locally applicable primitives. The analysis of the structural configurations into more primitive elements yields a grammatical hierarchy of syntactic relations, shown in Figure 6.7. The most basic structural relations are *projection*—the relation between nodes in the same $\overline{\mathrm{X}}$ phrase, and *sisterhood*—the relation between an X node and its complement, or an X$'$ node and its specifier. All other structural relations are defined in terms of these primitives. For example, the c-command

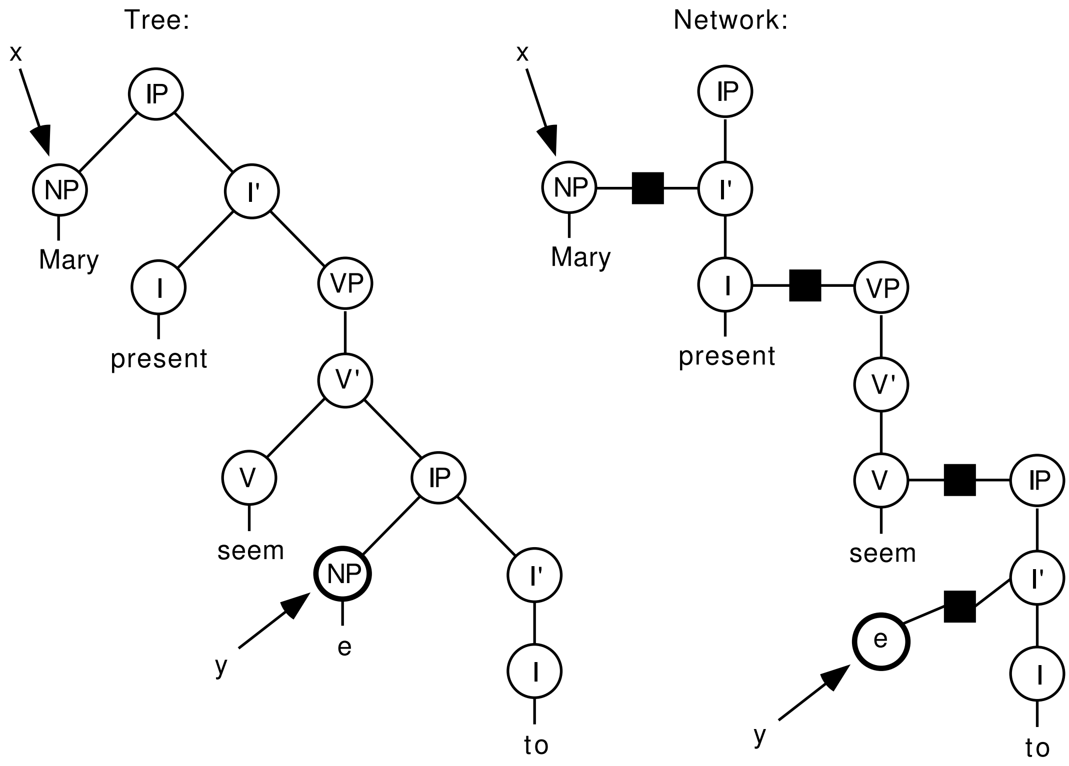


Figure 6.5: $x$ A-antecedent governs $y$.

relation can hold across any sister links, but only across downward projection links. The government relation is then defined as a further refinement of the c-command relation, which states that in addition the relation cannot hold across a "barrier." In this way, each relation in the hierarchy places some restriction on the relations from which it is constructed.

The grammatical hierarchy developed here provides more finely grained definitions of the structural restrictions than those given within the linguistic theory. In the parser, this hierarchy of syntactic relations has been formulated as an object-oriented network. Each object (node) in the grammatical hierarchy can have attached to it some constraint on how nodes in a parse tree are related if that structural configuration applies. For example, the grammatical node labeled "government" is where the restriction of not crossing barriers is attached, and the $\alpha$-relation nodes are where the Relativized Minimality restrictions are attached. Deeper nodes in the hierarchy inherit all the structural restrictions of their ancestor nodes. Thus, for example, the node for $\bar{\text{A}}$-antecedent government inherits the restrictions relevant to an $\bar{\text{A}}$-specifier relation, government, and c-command.[25] Because the more complex

[25]As can be seen in Figure 6.7, the definition of the grammatical hierarchy ensures that any government relation inherits the c-command restriction. In order to use c-command for the antecedent government relations and m-command for head government, it would be sufficient to make the c-command node a parent of the antecedent relation node instead of the government node, and to add an m-command node as a parent

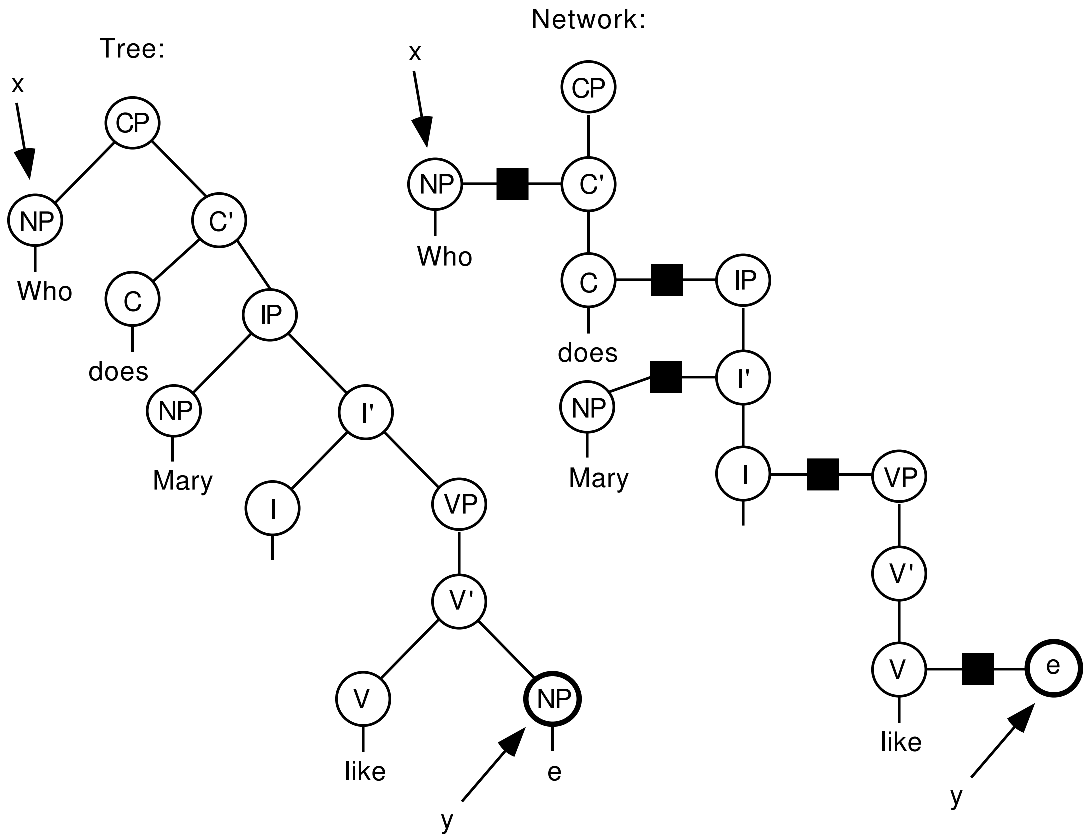


Figure 6.6: $x$ $\bar{\text{A}}$-antecedent governs $y$.

syntactic relations are built up from the primitive relations of projection and sisterhood, all of the restrictions are stated solely in terms of how they constrain a relation across individual projection and sisterhood links. This allows each of the structural restrictions to be implemented in the parser as a feature-passing constraint that is verifiable between any two neighboring nodes in the parsing network.

In order to ensure that the correct feature-passing restrictions apply to a particular symbolic feature, the features are defined as instances of the appropriate syntactic relation object within the grammatical hierarchy. In this way, the features inherit the appropriate local communication constraints for controlling how they are passed through the parsing network. A feature defined as an instance of a relation at the top of the hierarchy—that is, sisterhood or projection—is a local feature of the parser. A sisterhood feature can be passed only along a sisterhood link in the network (that is, from a p-node to a directly neighboring a-node); a projection feature can be passed only along a sequence of projection links. A feature defined as an instance of a relation that is deeper in the grammatical hierarchy—

of the head relation node.

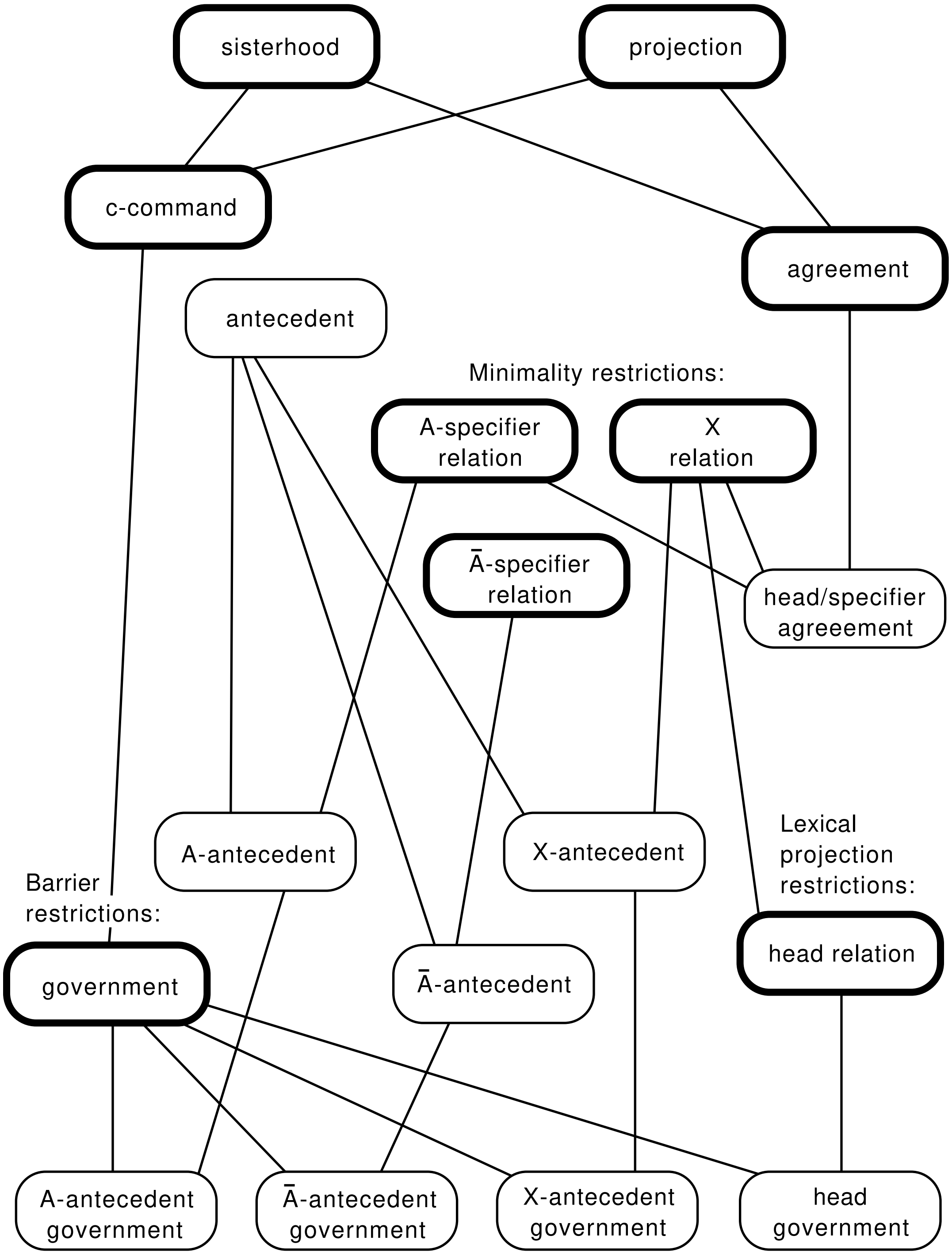


Figure 6.7: The object-oriented grammatical hierarchy of structural relations. Highlighted nodes impose relevant feature-passing restrictions.

such as $\bar{A}$-antecedent government—is a long-distance feature of the parser. Such a feature can be passed along any number of sisterhood or projection links in the network, subject to each of the communication constraints imposed by the restrictions on the grammatical relations that it inherits.

Since the definition of a feature includes the restrictions on how it may be communicated, the feature packets that are passed through the network implicitly include this information. All of the nodes in the parsing network use the same feature-passing algorithm for all features. The difference in how a feature is passed through the network arises solely from the particular communication restrictions that it inherits. At each node where the feature lands, the restrictions determine whether the node can pass on the feature to each of its neighbors. This uniform feature-passing mechanism ensures both local and non-local structural constraints on syntactic relations, using only local information at each node along a feature-passing path. The resulting method has been successfully applied to correctly pass features according to each of the structural restrictions described above in Section 6.3.1.

To examine the feature-passing mechanism in more detail, consider Figure 6.8 in which a WH-phrase must bind a trace that it antecedent governs. The binding feature from the WH-phrase *Who* is defined as an instance of an $\bar{A}$-antecedent government feature, and therefore inherits the feature-passing restrictions from the $\bar{A}$-antecedent government object in the grammatical hierarchy. The c-command restriction entails that a feature can be passed only on a downward link within an $\overline{X}$ phrase, and also prevents the original source node from passing the feature within its own phrase. These two conditions ensure that a node only c-commands the nodes in the subtree of its sister. The government restriction entails that a feature cannot be passed across an a-node whose sister-is-selected feature is FALSE. The c-command and government restrictions thus result in the binding feature from *Who* being passed only to nodes that are c-commanded by *Who*, with no barriers intervening. The $\bar{A}$-specifier relation imposes the appropriate Relativized Minimality constraint by disallowing a feature from being passed below an $\bar{A}$ specifier position. The binding feature from the WH-phrase cannot pass the location of a potential $\bar{A}$ binder, since that position is a closer potential binder for the trace.

These results demonstrate how feature-passing in the parser is constrained according to the grammatical hierarchy, by verifying local restrictions at each decision point along a feature-passing path. The analysis of each non-local structural configuration[26] into a set of local feature-passing primitives is an important component of the computational theory of parsing developed here, since it allows the parser to implement the structural constraints of the theory without the use of phrase structure rules or a global control mechanism. The constrained feature-passing method, in conjunction with the feature unification and constraint-checking mechanisms described in Section 6.2.2, forms the basis of a unique distributed parsing approach for constraint-based linguistic theories. These symbolic mechanisms determine the grammaticality of node configurations in the parsing network, and the numeric processing functions choose the preferred attachments from among those. Chapter 7 will

[26]That is, not a sister or projection relation.

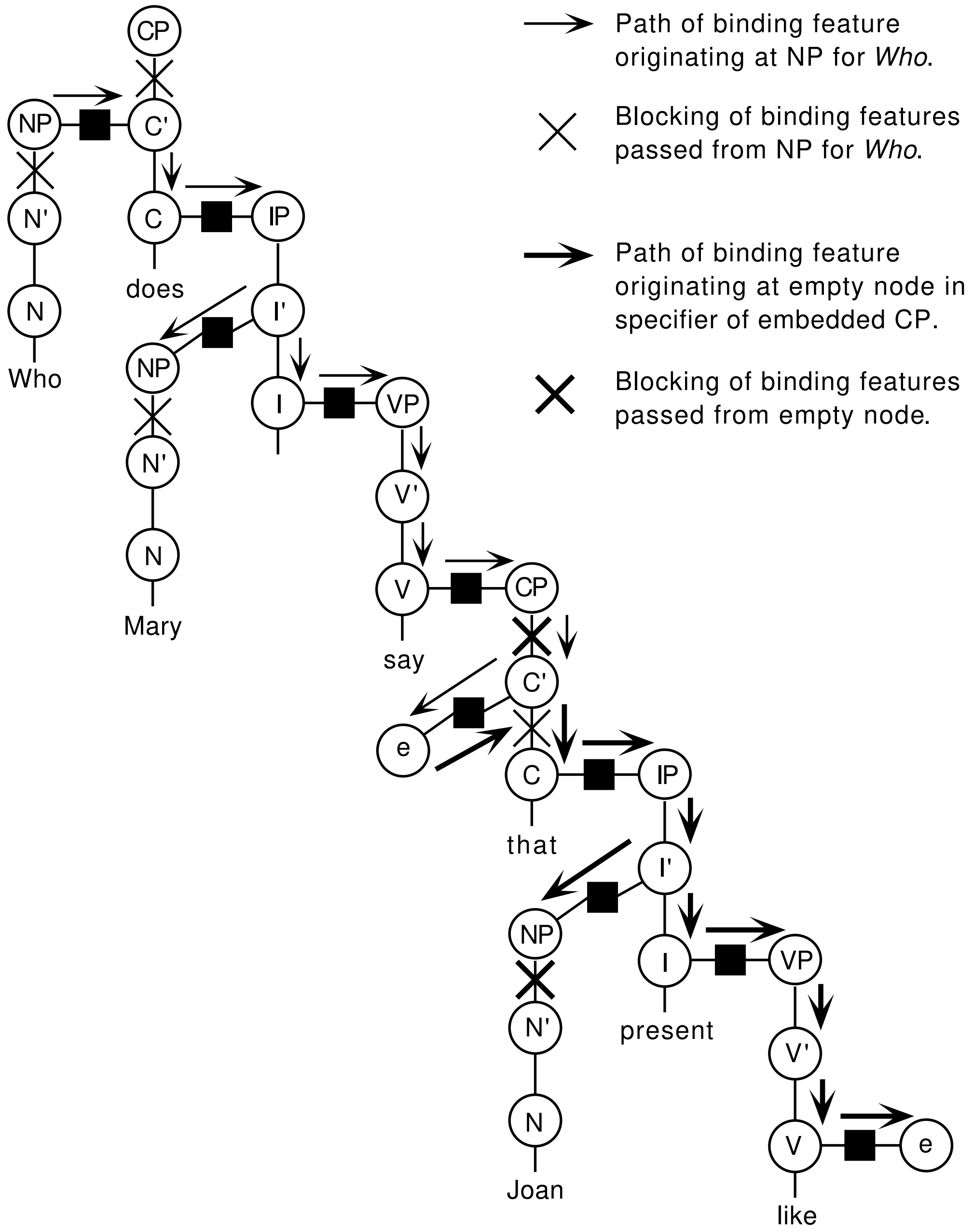


Figure 6.8: A binding feature passed from *Who* to its potential trace must conform to the feature-passing restrictions inherited from the Ā-antecedent government relation in the grammatical hierarchy.

demonstrate how the techniques apply to actual linguistic input in the processing of syntactic ambiguities.

## 6.4 Limitations of the Symbolic Processing

The obvious limitation of the symbolic processing of the parser is that only a fairly small subset of Government-Binding theory is implemented. However, the intent of the research was not to exhibit broad coverage of English syntax, but to demonstrate the feasibility of the proposed hybrid connectionist techniques for providing a direct implementation of a constraint-based grammatical theory. The important question then is whether the addition of more syntactic features, constraints, or grammatical relations will slow down the parser to the point of diminishing the usefulness of the approach.

There are several reasons to believe that the techniques will scale up to handle more comprehensive syntactic knowledge. One relevant factor is that the parser does not currently exploit a high degree of parallelism. Adding a large number of features to the model as it stands would slow things down considerably, because each node in the network would have to sequentially process many symbolic features. However, a straightforward extension of the model would replace each processing node with a set of nodes, one for each type of feature. All types of features could then be processed in parallel. The extra space would not be a problem, given that the parser now only uses a maximum of $8n + 3$ processing nodes during a parse of an input of length $n$.[27]

Similarly, additional syntactic constraints could be efficiently added to such a model: since most constraints would apply to only a small subset of features, verification of all of the constraints would proceed largely in parallel. Enhancing the ability of the message-passing algorithm to verify additional structural relations would also not have a great effect on the parser's efficiency, since very few of the primitive restrictions apply at any particular node. Furthermore, as discussed in Section 6.3, the parser already has the ability to capture most of the important structural relations referred to by GB. Although increasing the degree of parallelism exploited by the model is left for future research, it is clear that the current restriction of the parser to a small subset of the linguistic theory is not a limitation that is inherent to the approach.

[27] $4n$ a-nodes, $3n$ p-nodes, $n + 2$ empty nodes, and 1 stack node, for a total of $8n + 3$.

# Chapter 7

# Results of Parsing Syntactic Ambiguities

Since an adequate characterization of how people process and resolve syntactic ambiguities has not yet been achieved, designing a computational parser that processes these ambiguities in a way that matches human behavior has proven difficult. Recall the motivations for developing a principled model of parsing, in which human-like behavior results from independently motivated computational assumptions: to gain a better understanding of the computational basis of human behavior, and to achieve a better match with human performance. This chapter describes how the competitive attachment model realizes both of these goals. The model has been tested on a number of key syntactic ambiguities that have received considerable attention in the psycholinguistic and computational linguistic literature. The competitive attachment process will be shown to account for a number of well-known human structural preferences, without the use of construction-specific preference heuristics. Revisability of preferred analyses and associated acceptability judgments in the model also match human performance, without the use of explicit revision strategies. Furthermore, the competitive dynamics of the model mimic finer-grained on-line processing effects, explaining observations of both serial and parallel processing in human parsing. Thus, the underlying assumptions of the model and the resulting competitive attachment process will be shown to provide a principled account of human structural disambiguation that conforms with a broad range of psycholinguistic data.

Section 7.1 reviews the goals that were set for the model, and relates them to specific classes of psycholinguistic data that the competitive attachment process of the parser will be shown to account for. Section 7.2 demonstrates the ability of the model to achieve these goals, by comparing the detailed results of running the parser to the experimental observations. Section 7.3 concludes the chapter by summarizing the key results of the model.

## 7.1 Overview of the Results

Chapter 1 argued that a principled model of parsing must explain three aspects of the processing of an ambiguity in the human parser: (1) the structure or structures that are maintained, (2) the preference for one structure over another, and (3) the ability or inability to revise attachment decisions. In order to evaluate the parsing model with regard to these three issues, we can ask the following corresponding questions, comparing its behavior to that of the human parser:

1. **Serialism vs. Parallelism**: When presented with an ambiguity, does the parser build and maintain a single structure or multiple structures?

2. **Structural Preferences**: How does the parser determine the preference for one possible structure over another?

3. **Reanalysis**: If the continuation of the input is incompatible with the preferred structure, how easily, if at all, is the parser able to revise its initial hypothesis?

The performance of the model must be evaluated within the context of psycholinguistic data relevant to these three aspects of human parsing. Specifically, the model must account for the following empirical observations:

1. The "contradictory" experimental evidence for serialism and parallelism.

2. The wealth of preference data across a range of linguistic constructions.

3. The exhibited range of difficulty in reanalyzing erroneous attachments.

Furthermore, to stand as a general and well-motivated account, the behavior of the model must not be built-in in an *ad hoc* manner, but must be shown to emerge from its independently justified computational assumptions.

In fact, the behavior of the parser that is relevant to these issues arises directly from its basic competitive attachment operation. A syntactic ambiguity by definition has the potential to give rise to multiple grammatical attachment choices. When more than one attachment for a phrasal node (p-node) is valid, the multiple attachment nodes (a-nodes) must compete for the output activation from the p-node. In the model, the resolution of syntactic ambiguity is formulated as the competitive distribution of activation through the network of attachment possibilities that the ambiguity gives rise to. The parallel competitive process of ambiguity resolution underlies the human-like behavior of the model in the three areas noted above. First, the restricted network structure of the model constrains the competitive attachment process in a way that yields insight into the question of whether human parsing proceeds serially or in parallel. Second, the competitive attachment decisions made by the parser result in a unifying characterization of human structural preferences. Third, because the competition mechanism attaches the current input phrase and revises earlier attachments simultaneously, within a restricted parallel atomic operation, the model provides a precise formulation of the conditions under which reanalysis is possible. In each case, the desired behavior has not been built into the model, but rather follows from its principled design.

The next section describes in detail how the model's competitive attachment process underlies a principled account of the resolution of syntactic ambiguities. The discussion will focus on the processing of subcategorization ambiguities—that is, structural ambiguities that arise from the ability of a verb to take more than one kind of complement.[1] For example, the verb *know* has a subcategorization ambiguity because it can occur with either a noun phrase or sentential complement; the verb *race* can occur with a noun phrase complement or with no complement (that is, it can be used intransitively). Because of the key role played by verbal information in the syntactic structuring of a sentence,[2] a principled account of the processing of subcategorization ambiguities is an important step in an adequate characterization of human parsing behavior. Chapter 8 will discuss how resolving ambiguity through the competitive distribution of activation can extend to other types of ambiguity as well, such as lexical ambiguity and argument/adjunct ambiguity.

## 7.2 Evaluation of the Model

The following three subsections present results of the model that are relevant to each of the three areas of inquiry discussed above—that is, serialism vs. parallelism, structural preferences, and reanalysis. The parser will be run on a number of example sentences that demonstrate how it mimics the human behavior in question. The examples are drawn form a large body of empirical data on human processing of syntactic ambiguities. Furthermore, these sentences exemplify key structural configurations that have been a focus of psycholinguistic research. Note that it is the structural properties of the examples that are important, not the particular words being used.[3] Thus, each sentence actually represents a *class* of sentence types, for which the particular input items chosen are one instantiation; the relevant syntactic characteristics of each example will be pointed out.

Recall from previous chapters that, in parsing a sentence, a preprocessing routine sequentially processes the sequence of input tokens. The preprocessor looks up each input token in the lexicon, and appropriately initializes a new syntactic phrase.[4] The new phrase is then connected to the existing parsing network. The spreading activation/message-passing loop of the network nodes is then triggered, with the distributed network processing continuing until an acceptable state of the network is reached.[5] Each example sentence will have one

[1]Since, in the current implementation of the parser, a phrasal node can only activate a single attachment node, only verbs with single complements will be considered.

[2]For extended discussion of the importance of the so-called combinatory information associated with verbs, see Boland (1991) and references therein.

[3]The effects on the results of using different lexical items will be ignored, except where issues surrounding lexical preferences are directly addressed. Different lexical items can have varying lexical preferences, which lead to different weights on connections to attachments in the model. However, in all simulations except the lexical preference examples, weights are assumed to be 1.0, so that there is no effect of differing lexical items.

[4]The parser currently operates with a small lexicon of about 25 entries sufficient to support the range of syntactic constructions relevant to the ambiguities of interest. Scaling up to a reasonable sized lexicon would entail developing an efficient parallel indexing algorithm, but this issue was not addressed here.

[5]Recall that in an acceptable state of the network, each p-node sends all of its activation to exactly one of its a-nodes, and each a-node is either turned on (fully active) or off (inactive) by its p-nodes.

or more input words whose attachment within the network is particularly revealing of the parser's performance as it relates to human behavior. At each of these critical points in the parse, the state of the network will be examined in some detail. The properties of the parsing network that are important to evaluating its behavior are the following:

- the attachments that are possible, given the partial parse tree and the current input word;
- the set of a-nodes that are active when the network of possible attachments reaches an acceptable state;
- the number of iterations that it takes for the network to reach that state;
- the amount of activation that each active a-node has in that state.

The last two factors—the number of iterations required for the network to settle and the amount of activation of each a-node at that point—will be used as a measure of relative difficulty of the attachment of an input word. The number of iterations that it takes for an input word to be incorporated into the parse state is the amount of time that the network required to decide on a valid set of attachments that included that word. This measure of time is assumed to correspond to word-by-word reading times in human parsing; as in interpreting human reading times, longer network times indicate increased processing difficulty. The amount of activation of an a-node also indicates the level of difficulty in making an attachment, since attachment nodes with less activation are weaker hypotheses about the parse tree structure. These and the other properties listed above will be presented where relevant in the description of the network at each of the critical processing points in the example sentences.

### 7.2.1 Serialism vs. Parallelism

Consider the following sentence, in which the verb has a noun phrase/sentential complement ambiguity; the post-verbal NP may be attached directly to the verb phrase, or as the subject of the sentential complement of the verb:[6]

(7.1) Sara believes women…

| | |
|---|---|
| . [end of sentence] | {Preferred resolution.} |
| … to be successful. | {Non-preferred resolution.} |

[6]The verb *believe* subcategorizes for an NP, IP, or CP complement; for simplicity of presentation, the discussion here will focus on the choice between an NP and an IP. Although all of the reported results are comparing NP and IP attachments, CP attachments behave exactly the same as the IP attachments. That is, the fact that both an IP and a CP are projected for a CP complement has no effect on the number of iterations required to make the relevant attachments; the numbers obtained in simulations using a CP complement were exactly the same as for an IP complement.

People strongly prefer to attach the NP directly to the verb phrase as the complement of the verb; this preference for the "simpler" alternative is commonly known as Minimal Attachment (Frazier, 1978).

Various sources of psycholinguistic evidence concerning the processing of these types of ambiguities appear to support contradictory sentence processing models. People exhibit consistent strong preferences for one continuation of the sentence over the other—a continuation compatible with the noun phrase complement analysis—supporting the view that the human parser creates and maintains a single structure for the ambiguous initial string (Frazier, 1978; Frazier & Rayner, 1982). The serial model hypothesis gains additional support from experimental evidence indicating that people require time to revise the preferred hypothesis when the sentence continues in the non-preferred way (Frazier & Rayner, 1982). However, proponents of a parallel parsing model point to the fact that people have no *conscious* difficulty in parsing a non-preferred continuation—that is, a sentential complement. Furthermore, syntactic priming experiments contribute evidence that the human parser has access to the non-preferred structural alternative prior to having seen explicit evidence for that alternative in the input (Gorrell, 1987). The priming data have been interpreted as support for the parallel construction and maintenance of the alternative complement possibilities. This section will demonstrate how the competitive attachment model accounts in a natural way for each of these results, providing a unifying account of serial and parallel effects in processing syntactic ambiguities.

### Preference for a Single Reading

In the proposed model, given a sentence with a noun phrase/sentential complement ambiguity as in example (7.1), the so-called Minimal Attachment preference is a direct result of the properties of the competitive attachment process and the lack of top-down precomputation. The parsing network at the point of processing the NP *women* is shown in Figure 7.1.[7] The NP has valid attachments to the stack (a-node $a_0$) and to the V (a-node $a_1$). The lack of top-down precomputation prevents the network from creating an inflection phrase corresponding to the sentential complement possibility. Thus, at this point in the parse, the network has no representation of a potential attachment of the NP as the subject of an embedded clause. Since the default stack attachment is *of necessity* less competitive than an attachment to the developing parse tree,[8] a-node $a_1$ (the NP-to-V attachment) becomes highly activated, with the network settling in only 17 iterations. The parsing network after the losing a-nodes are deallocated is shown in Figure 7.2. The basic assumptions of the model force it to settle on a single analysis of the input, as in a serial model, thus accounting for the observed preference.

[7] Note that although a tensed verb such as *believes* projects a full sentential structure (that is, CP/IP/VP), the figures here are simplified by omitting display of the CP of root clauses. The figures are further simplified by omitting grammatically invalid a-nodes and irrelevant empty nodes.

[8] See the discussion on "Reasonableness," beginning on page 82 in Section 5.3.4, for a discussion of the competitive properties of the stack node in comparison to other phrasal nodes.

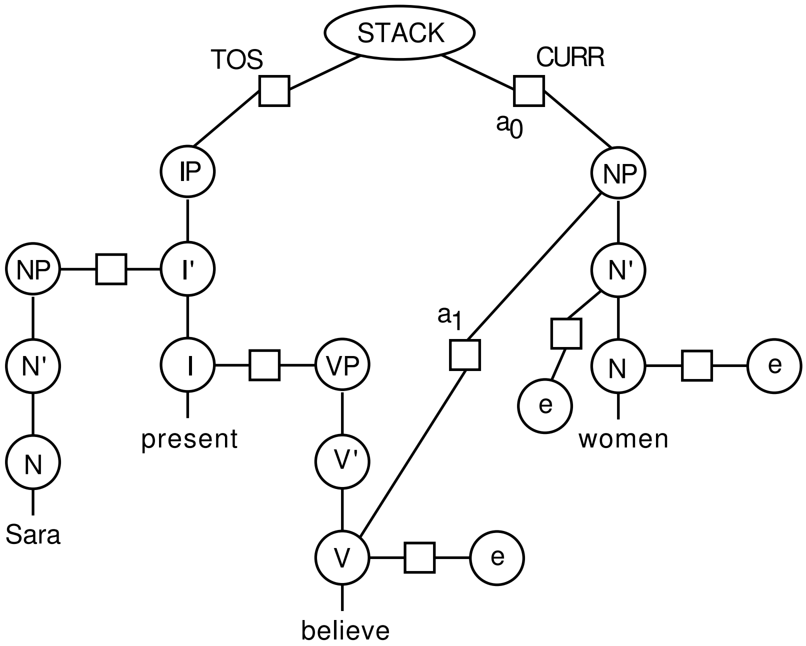


Figure 7.1: The network after projecting the NP *women*, in the sentence beginning *Sara believes women.*

In fact, a closer examination of the data regarding this preference reveals that the competitive attachment model is more compatible with human processing than are most other serial *or* parallel models. Race-based serial models (for example, Frazier, 1987; Frazier & Rayner, 1982; McRoy & Hirst, 1990), as well as parallel models that build multiple structures in response to an ambiguity (for example, Gibson, 1991; Gorrell, 1987), predict an increased processing load at the point of processing a verb with a subcategorization ambiguity.[9] Psycholinguistic experiments have found no evidence of such a processing load (see the discussion in Frazier, 1987). For example, in an on-line judgment experiment by Gorrell (1987), reaction times to sentences with unambiguous NP complement verbs were the same as reaction times to sentences with ambiguous NP/sentential complement verbs that were used with an NP complement. Both of these reaction times were significantly faster than those to sentences with ambiguous NP/sentential complement verbs where the verb was used with a sentential complement. Parsing models that build multiple structures in response to a subcategorization ambiguity are most compatible with an increase in response time for the ambiguous verb in *either* of its usages. In the competitive attachment parser, the inherent restrictions on the model constrain it to consider only a single analysis, and so there is no

[9]Serial models are normally believed to not lead to an increased processing load at an ambiguity. However, serial models in which the choice of which structure to maintain is based on a race (or other ranking procedure, as in Inoue & Fodor, in press) necessarily depend on the preliminary exploration of the multiple choices in parallel.

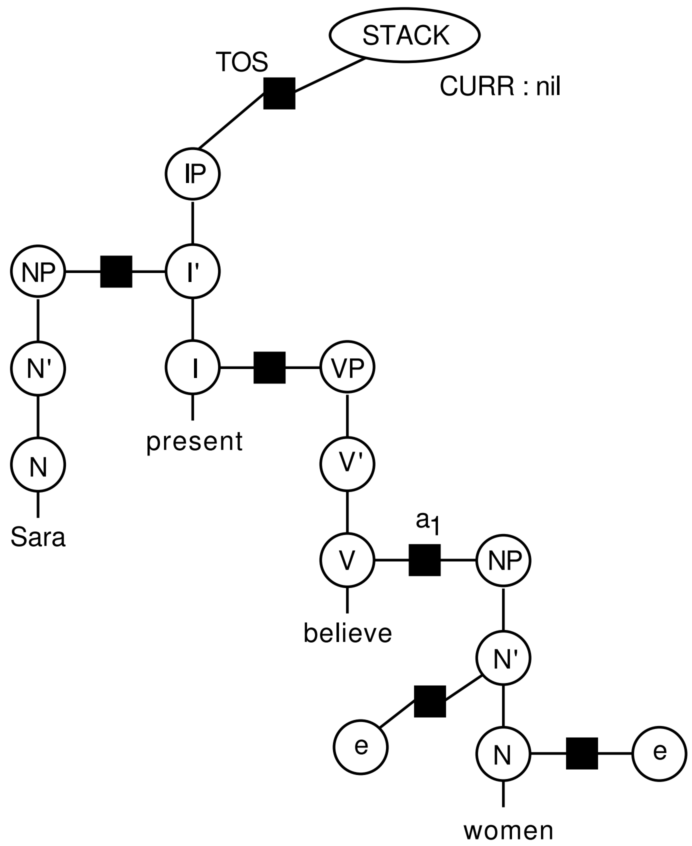


Figure 7.2: The network after attaching the NP *women* to the verb.

increase in processing load. The model thus conforms better with the experimental data.

**Priming of the Non-Preferred Alternative**

One of the pieces of evidence for a parallel model that builds and maintains multiple structures is the data from Gorrell (1987) that showed that the non-preferred alternative of these subcategorization ambiguities could prime a lexical decision task that immediately follows the post-verbal NP. Subjects were shown (word-by-word) sentences containing a verb with an NP/sentential complement ambiguity, and at the end of the post-verbal NP were presented with a target for a lexical decision task.[10] In this experimental paradigm, subjects appear to try to integrate the lexical decision target into the syntactic structure of the displayed sentence. Targets were either pronouns or modal verbs. A pronoun target could not be a grammatical continuation of the sentence, regardless of the resolution of the subcategorization ambiguity of the prior verb. However, a modal verb target could be a grammatical

[10]The target word was displayed above the words of the sentence being presented; subjects were asked to press one of two buttons indicating whether the target was a word or not.

continuation if the ambiguous verb was interpreted as taking a sentential complement. Gorrell found that the lexical decision task for modal verbs was significantly faster than for pronouns following an ambiguous verb and the post-verbal NP; that is, a target compatible with a sentential continuation is primed relative to the pronoun target. Furthermore, there was no difference in speed of decision between a modal verb following an ambiguous verb, and a modal verb following an unambiguous sentential complement verb. Thus, processing a sentential continuation appears to be equally easy for an ambiguous NP/sentential complement verb and an unambiguous sentential complement verb. Gorrell argued that the timing of the lexical decision task entailed that any model that performed reanalysis given the non-preferred continuation would be unable to account for this data, since reanalysis in the ambiguous verb case would entail a slow-down in processing the target modal verb. This is a potential problem for the competitive attachment model. Because only the so-called Minimal Attachment structure is initially created, the parser must revise its analysis when presented with the evidence for a sentential complement. However, the parallel competitive attachment operation of the parser in fact leads to a natural account of priming of the non-preferred alternative.

Mimicking the format of the syntactic priming task, assume the parser is given the lexical decision target *to* after processing the initial sentence fragment of example (7.1), and tries to integrate that word into the current parse tree:[11]

(7.2) Sara believes women [to]

At the word *to*, the parser projects an IP; its initial connections to the network are shown in Figure 7.3.[12] Note that the multiple complement possibilities of the verb are active simultaneously—that is, the two complement attachments to the verb, $a_1$ and $a_2$, are competing for activation from the V node. The subcategorization of *believe* for an IP raises the state value of a-node $a_2$ between the V node and the IP node. Thus, the attachment of *to* to *believe* is activated ("primed") by the expectation of *believe* for an IP complement, accounting for the priming effect found by Gorrell. It is important to note that it is the active *expectation* for the IP, not the pre-computation of sentential structure, that is responsible for the priming effect. Also, since the parallel attachment operation of the parser immediately integrates the IP into the parse tree, the model can account for the timing of the lexical decision task and the priming effect, unlike a serial model with reanalysis. The result depends on the automatic postulation of all possible attachments between the IP and the existing parse tree. It should therefore be emphasized that the a-nodes as explicit representations of possible attachments are an integral and necessary part of the parser. That is, the active expectations embodied in the a-nodes are not an *ad hoc* mechanism added onto the parser merely to achieve this kind of priming effect.

[11]Here I am using the infinitive marker *to* as the "lexical target" to be consistent with the other examples in this chapter; using a modal verb as in Gorrell's experiment would lead to equivalent behavior in the model.

[12]Since the parser currently does not handle lexical ambiguity, the word *to* is projected only as an inflection phrase.

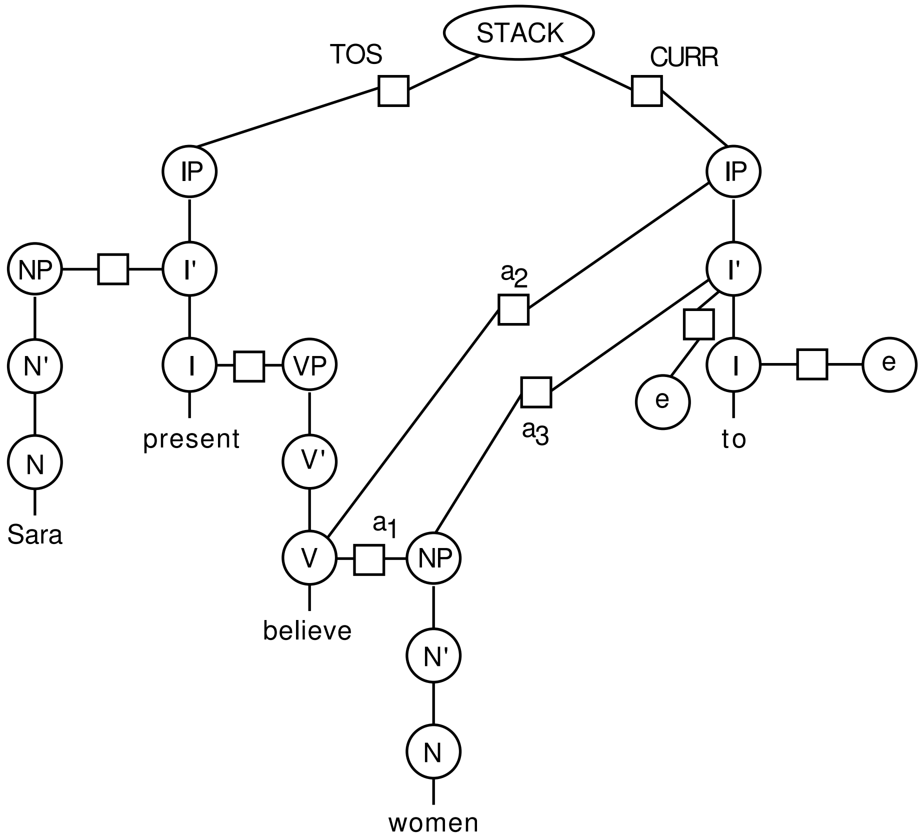


Figure 7.3: The network at the point of processing the word *to* after processing *Sara believes women.*

Since priming is a process of relative facilitation of processing, the complete picture of the priming effect requires an examination of how the network behaves given a target that would constitute an ungrammatical continuation of the sentence. The assumption is that, while the expectation for a target that would indicate an IP continuation eases the lexical decision task, the increased difficulty of processing a target that would indicate an ungrammatical continuation interferes with the lexical decision task; see Gorrell (1987). When processing a target compatible with the IP continuation, as in Figure 7.3, the network reaches an acceptable state in 24 iterations. By contrast, given an ungrammatical continuation (using a pronoun as the target word, as in the Gorrell experiment), the network takes 33 iterations to settle. Thus there is a clear facilitation of the processing of a target compatible with a sentential complement continuation in comparison to one constituting an ungrammatical continuation.

The network accounts for the priming of the non-preferred alternative as the consequence of active expectations in the form of multiple potential attachment nodes that simultaneously compete for activation. The approach is quite different from parallel models that maintain and rank multiple alternatives, such as in Gibson (1991) or Gorrell (1987). A ranked-parallel model must incorporate additional assumptions to explain the following two facts: first, that NP complement structures are judged grammatical faster than sentential complement structures, and second, that equivalent priming of a sentential complement is obtained for ambiguous NP/sentential complement verbs and unambiguous sentential complement verbs. To account for the judgment data, a ranked-parallel model must specify that the judgment task is sensitive to the ranking of the structures. On the other hand, to account for the priming data, it must be further specified that the lexical decision task is *not* sensitive to ranking. The competitive attachment model avoids such stipulations in the sensitivity of particular tasks, because the preference shown by the parser does not rely on a ranking of two existing structures. Its explanation of the preference data and the priming data is therefore more parsimonious than these previous approaches.

**Evidence for Reanalysis**

In addition to the syntactic priming data, proponents of a parallel model point to the fact that the non-preferred resolution of this type of ambiguity is easy for people to process. However, detailed eye-movement studies (Frazier & Rayner, 1982) have shown that people exhibit longer per-letter reading times in the disambiguating region when these types of sentences are resolved in the non-preferred way.[13] The competitive attachment model provides a unifying account of these disparate observations. Consider the following sentence, containing a non-preferred resolution of the ambiguity:

(7.3) Sara believes women to be successful.

At the point of processing the disambiguating word, *to*, the network has the structure that was shown in Figure 7.3. The same set of a-nodes that define the initial attachment possibilities for the current IP phrase, $a_2$ and $a_3$, simultaneously define the revised attachment necessary for the NP *women*. The NP-to-V attachment, a-node $a_1$, competes both with $a_2$ for the activation from the V node, and with $a_3$ for the activation from the NP node. These two competitions draw activation away from $a_1$. The network reaches an acceptable state in 24 iterations; when it does so, $a_2$ and $a_3$ are highly active and $a_1$ has become inactive, resulting in the network of Figure 7.4. In a single atomic operation, the network has revised its earlier attachment hypothesis for the NP and incorporated the new IP phrase into the parse tree. Changing the attachment of the NP in this way is possible only because that revision involves one of the sets of logical attachment possibilities allowed by the competitive attachment process.[14]

Because the necessary revision occurs within the normal attachment operation of the parser, the model accounts for the fact that people are not consciously aware that they are

[13] A related experiment in Rayner & Frazier (1987) provides evidence that the increased reading times are not just a complexity effect that arises due to the processing of a subordinate clause structure.

[14] See Figure 4.32 on page 57, and the accompanying discussion in Section 4.3.

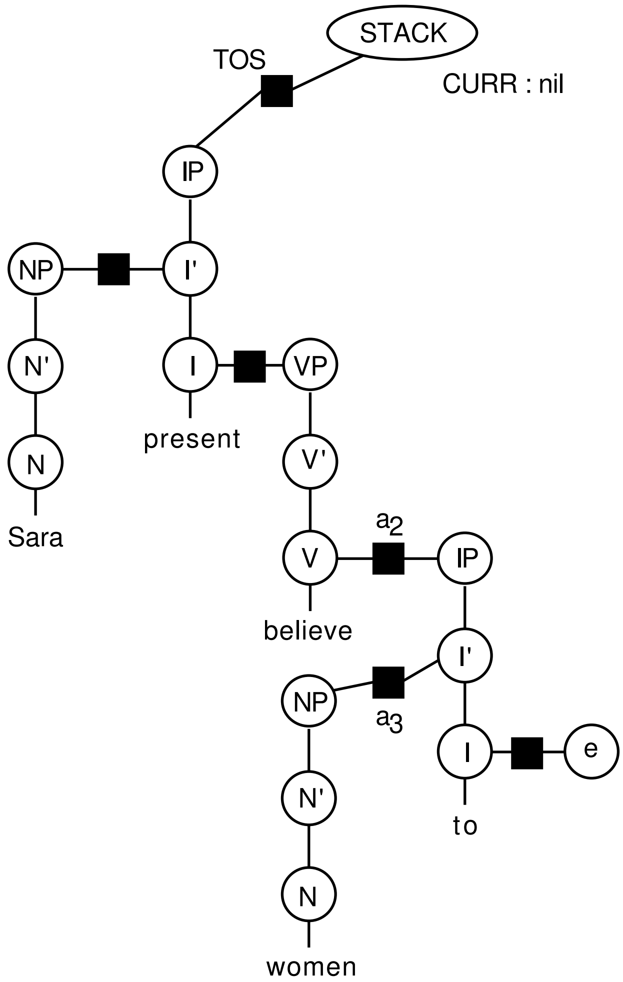


Figure 7.4: The network after re-attaching the NP to the I′, and attaching the IP to the verb.

revising an earlier structure. However, the operation of the model contrasts with traditional parallel approaches, in which a less preferred alternative analysis may be immediately selected from multiple structures that are already computed. Here the revision does require a reanalysis process, consisting of a competition for activation between the old and new attachment nodes. Because of the increased competitive activity, the network takes 24 iterations to settle after *to*, as compared to 17 iterations after *women*. The model thus accounts for the longer reading times exhibited at the disambiguation point in the Frazier & Rayner experiments. The increase in reading time in their data consisted of more fixation durations in the disambiguating region (here, the word *to*, which indicates an IP complement) and regressive eye movements to the post-verbal NP. These are the key phrases involved in the competitive activation process of the model. It seems plausible that eye movements focusing on the relevant input could correspond to the competitive activation of nodes in the parse tree.[15] Parallel models that maintain multiple structures cannot account naturally for the observed pattern of eye movements, given the immediate adoption of an available alternative in those approaches. A serial model, on the other hand, must rely on explicit strategies within the parser for directing its attention appropriately, while the competitive attachment model captures this focusing of attention automatically.

#### Summary

The restricted parallelism of the competitive attachment model has been shown to lead to both serial and parallel processing behaviors in parsing syntactic ambiguities. The lack of top-down precomputation and the necessity of focusing on a single structural analysis lead to strong initial preferences; the competitive spread of activation entails that changes to those preferred structures requires time. These serial aspects of the model's performance are complemented by its parallel behaviors, arising from its active expectations and parallel attachment operation. Together, these properties of the model provide a concise and unifying account of the observed serial and parallel aspects of human parsing.

### 7.2.2 Structural Preferences

People exhibit a number of structural preferences when processing subcategorization ambiguities. Although individual preference heuristics may be able to determine the preferred attachment in a range of configurations, such strategies constitute neither a concise nor explanatory approach to ambiguity resolution. Furthermore, formulating structural preferences as explicit strategies leads to complications when the strategies conflict, as they inevitably do (for further discussion of this point, see McRoy & Hirst (1990)). This section demonstrates that the competitive attachment mechanism is a plausible model of the underlying causes of several observed attachment preferences: Minimal Attachment and Late Closure, as well as recency and lexical strength effects. In each case, it will be shown that the exhibited preferences are a result of the interaction of fundamental, independently motivated properties of the model. The competitive attachment model will also be shown to provide a natural integration of preference factors.

#### Minimal Attachment and Late Closure

The account of the Minimal Attachment preference was given in Section 7.2.1, which focused on the processing of verbs with NP/sentential complement subcategorization ambiguities. It

[15] Given this explanation of the eye-movement data, one might wonder why the data do not show significant regressive eye movements to the verb, since the V node is also actively engaged in the reanalysis competition. I would propose that the focus of attention in reading corresponds to the re-activation of XP nodes (in this case, the NP and the IP), since those nodes are determining the attachment of their $\overline{\mathrm{X}}$ phrases into the parse tree.

was shown that the competitive attachment model arrives at a "Minimal Attachment" analysis of the initial input string of example (7.1) due to the competitive attachment mechanism and the lack of top-down precomputation. These very same properties apply in so-called Late Closure sentences, exemplified by the following:

(7.4) When Kiva eats food...

| | |
|---|---|
| ...it disappears. | {Preferred resolution.} |
| ...disappears. | {Non-preferred resolution.} |

In these sentences, the subordinate verb is optionally transitive—that is, it can occur with an NP complement or without a complement. People show a strong preference for the Late Closure reading in which the post-verbal NP attaches as the complement of the subordinate verb (the transitive reading of *eat*) rather than as the subject of the main clause (in which case *eat* is intransitive).[16]

Figure 7.5 shows the configuration of the network at the point of processing the NP *food*; compare the network to Figure 7.1 on page 118. Again, because of the lack of top-down precomputation, the NP has only the options of attaching to the verb or to the stack. Since the phrasal structure for the main clause has not yet been allocated, the possibility of attaching as the subject of the main clause (as would be required by the non-preferred continuation of the sentence) does not exist. As in example (7.1), the post-verbal NP makes the best attachment available to it, as the complement of the verb. Since the initial attachment in these cases of Late Closure is determined in exactly the same manner as in the Minimal Attachment cases illustrated by sentence (7.1), these two classic preferences receive a uniform account in the proposed model. In Section 7.2.3 we will return to the behavior of the parser given the non-preferred continuation of example (7.4), which is quite different from the behavior given the non-preferred continuation of example (7.1).

### Recency

The human parser shows a strong tendency to attach the current input phrase to more recent syntactic structure. In a number of parsing models, this preference has been stated as an explicit processing strategy (for example, Kimball's "Right Association" (1973), Frazier's "Late Closure" (1978), and Gibson's "Recency Preference" (1991)). By contrast, in the competitive attachment model, the active memory management techniques required to maintain the pool of network processing nodes *indirectly* give rise to recency effects. Since the number of processing nodes in the parser must be finite, a scheme has to allow for their reuse. Earlier chapters have explained that, in order to accomplish this, the activation of p-nodes decays

[16]It has been proposed that a sentence such as *When Kiva eats food disappears* is ungrammatical because of the lack of appropriate punctuation—that is, a comma after *eats*. See Gibson (1991) for arguments that these types of examples cannot be consistently ruled out in the grammar, given the grammaticality of sentences such as *When Kiva eats food it disappears*, which also lacks a comma following the subordinate clause.

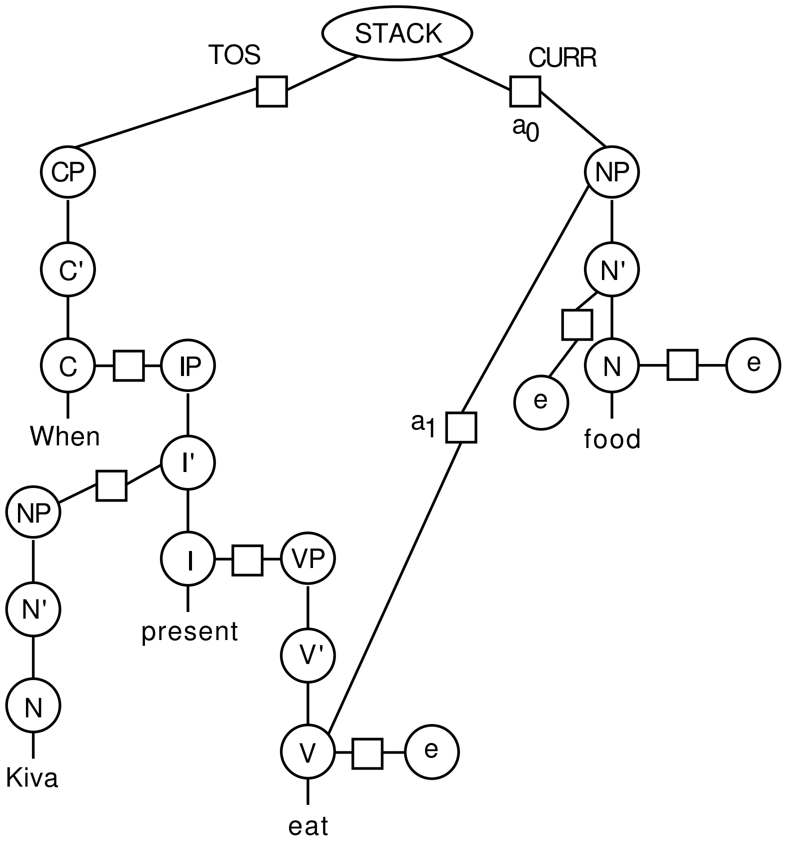


Figure 7.5: The network at the point of processing the NP *food* in the sentence beginning *When Kiva eats food.*

over time. This memory management technique—the decay of phrasal activation—leads to a principled account of recency effects.

Consider the following sentence, in which each verb—*believe* and *suspect*—can have either an NP or IP complement:

(7.5) Sara believes the report that women suspect Laika to...

| | |
|---|---|
| ...have committed the crime. | {Preferred resolution.} |
| ...have been published. | {Non-preferred resolution.} |

In the preferred continuation, *suspect* has an IP complement, [*Laika to have committed the crime*], and *believe* has a factive NP complement, [*the report* [*that women suspect Laika to have committed the crime*]]. In the less preferred continuation, *suspect* has an NP complement, *Laika*, and *believe* has an IP complement, [*the report* [*that women suspect Laika*] *to have been published*]. The ambiguity arises at the word *to*, which projects an IP that can attach either to the embedded verb *suspect* or to the main verb *believe*.

Figure 7.6 shows the state of the network at this point in the parse. The IP headed by *to* can activate a-node $a_3$, replacing the NP headed by *report* as the complement of *believe*, or it can activate a-node $a_4$, replacing the NP headed by *Laika* as the complement of *suspect*. The lower verb has more activation to output than the higher verb, because it was allocated more recently and its activation has decayed less. This means that a-node $a_4$, the IP attachment to the lower verb, gains more activation from its verb than a-node $a_3$ does from the higher verb. A-node $a_4$ therefore competes more effectively for activation from the IP, and the IP makes the attachment as the complement of the lower verb *suspect*. Although the IP makes the lower attachment, the existence of the higher attachment is not irrelevant. In this configuration, the increased competition of choosing between two attachments causes the network to require 31 iterations to reach an acceptable state, compared to 24 iterations when there is only a single valid attachment for the IP (as in example (7.3)).

The issue of recency effects will be addressed again in Section 7.2.3, since the recency of nodes involved in a necessary revision of attachments also affects how easily the reanalysis proceeds.

### Lexical Preferences

A number of sentence processing theories have incorporated a model of the effect of lexical expectations on the ease of analyzing certain inputs (for example, Ford, Bresnan, & Kaplan (1982); MacDonald (1994); Tanenhaus, Stowe, & Carlson (1985)). Connectionist processing techniques are able to naturally integrate this type of preference information through the use of weighted connections. In the competitive attachment model, lexical strength is represented by the weights on the links between a p-node and its potential attachments, and these weights reflect the frequency of a p-node licensing a certain category of XP attachment.[17]

Consider again the sentence of example (7.3), repeated here:

(7.6) Sara believes women to be successful.

To test the effects of lexical expectations in the parsing network, a comparison was made of the ease of reanalysis required at the word *to* given different strengths for the NP/IP complement expectations of the verb *believe*.[18] In the baseline test, the verb's expectation values for an NP and an IP are the same high value; this is the "normal" condition, which was used for all reported tests of the parser except for those explicitly involving comparisons of lexical strength. Under these conditions, the NP requires 17 iterations to make its attachment to the verb, and the IP requires 24 iterations to make its attachment, which simultaneously

[17] Since the model does not currently include a theory of learning, these weights are hard-coded into the lexical entries. Adapting a connectionist learning algorithm to adjust these weights based on the experience of the parsing network would be straightforward. The important point is that the computational framework of the model can naturally encode this information and apply it in determining attachments.

[18] Recall that although the results reported here focus on the NP/IP choices, the figures obtained are the same for a CP as for an IP, and are also the same whether the subcategorization choice is between an NP and IP, an NP and CP, or an NP, IP and CP.

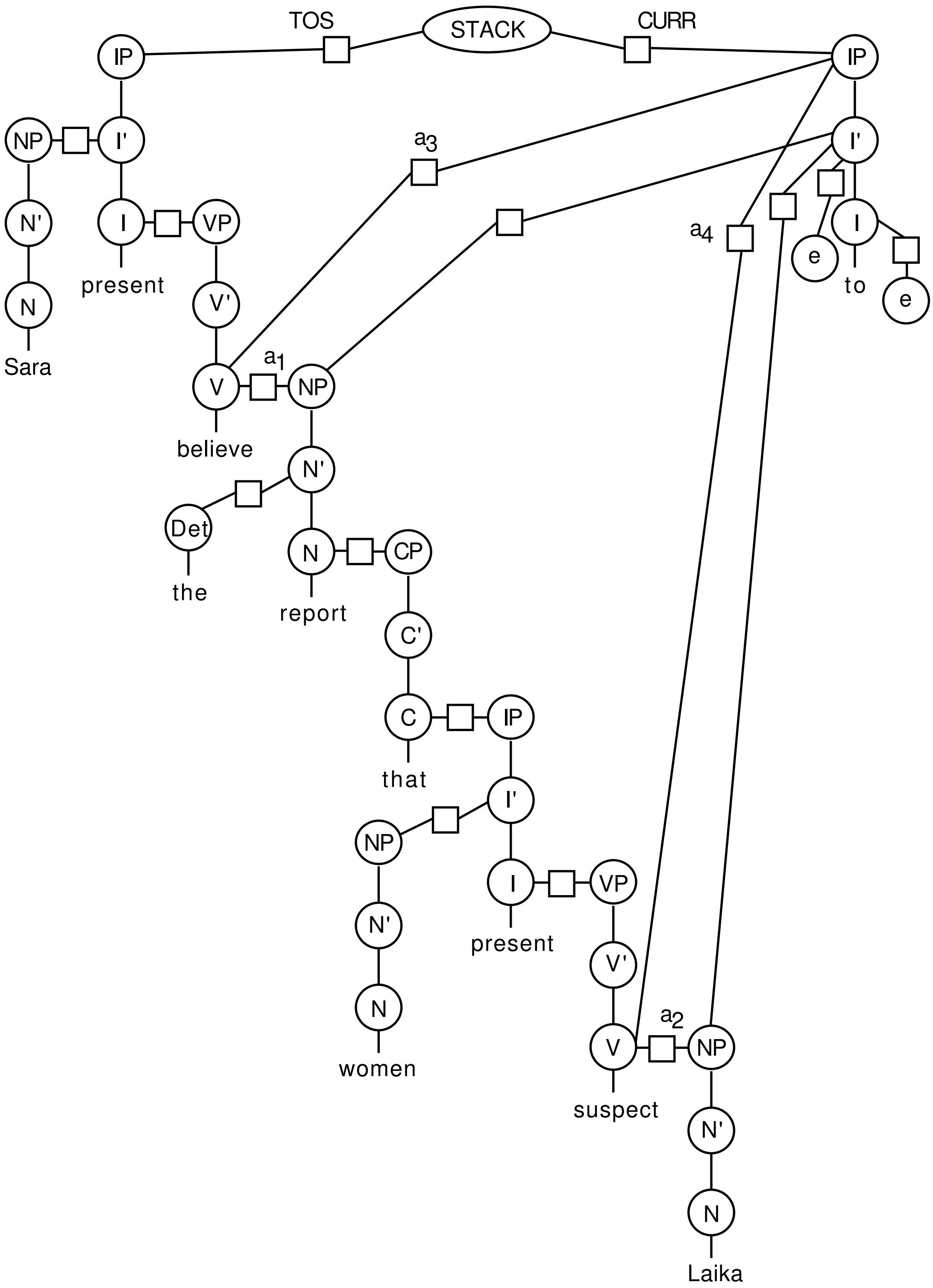


Figure 7.6: The network at the point of processing *to*.

| Strength of Expectation for Each Complement | | Number of Iterations to Make Attachment | |
|---|---|---|---|
| NP | IP | NP | IP |
| 1.0 | 1.0 | 17 | 24 |
| 1.0 | .9 | 17 | 25 |
| 1.0 | .8 | 17 | 26 |
| 1.0 | .7 | 17 | 27 |
| 1.0 | .6 | 17 | 28 |
| 1.0 | .5 | 17 | 30 |

Table 7.1: The effect of varying strength of expectation on the number of iterations required to make the post-verbal NP and IP attachments in the sentence *Sara believes women to be successful.* The strength of expectation is varied from high to low for an IP complement of the verb, which has an NP/IP subcategorization. The number of iterations required to initially attach the post-verbal NP is unaffected, but the number of iterations required to attach the IP increases with decreasing strength.

involves reattaching the NP as the specifier of the I′.[19] The following tests examine the effect of varying lexical preferences on these baseline figures.

The first test was to hold constant the strength of expectation of an NP complement at the highest possible value, and to vary the strength of the IP expectation from high to low; the results are summarized in Table 7.1. Under these conditions, the NP always took 17 iterations to make the initial attachment to the verb. The speed of the NP attachment is unaffected by the degree of expectation for the IP, since the IP attachment is not yet being considered when the NP is first attached. By contrast, the number of iterations for the IP to make its attachment increased steadily from 24 to 30. As the lexical expectation for the IP decreased, the parser had greater and greater difficulty in settling on the necessary attachments for the reanalysis. Hence, the decrease in lexical strength had an inhibitory effect on the IP attachment, by decreasing its ability to compete strongly with an attachment alternative.

The second test was to hold constant the strength of expectation of an IP complement at the highest possible value, and vary the strength of the NP expectation from high to low. The number of iterations to make the initial NP-to-V attachment increased from 17 to 20, while the number of iterations to make the IP attachment decreased from 24 to 22; see Table 7.2. Again, a decrease in lexical strength, this time of the NP expectation, had an inhibitory effect on making the corresponding attachment, even though the NP-to-V attachment is competing only with the default attachment of the NP to the stack. Additionally, although the expectation for the IP was held constant, its attachment was made easier as the NP expectation was decreased. Thus, a decrease in strength of expectation for a phrase not only inhibits its own attachment, but can facilitate attachments that compete with it by making

[19]These are the same values as were presented in earlier discussion of this example.

| Strength of Expectation for Each Complement | | Number of Iterations to Make Attachment | |
|---|---|---|---|
| NP | IP | NP | IP |
| 1.0 | 1.0 | 17 | 24 |
| .9 | 1.0 | 18 | 23 |
| .8 | 1.0 | 18 | 23 |
| .7 | 1.0 | 18 | 22 |
| .6 | 1.0 | 19 | 22 |
| .5 | 1.0 | 20 | 22 |

Table 7.2: The effect of varying strength of expectation on the number of iterations required to make the post-verbal NP and IP attachments in the sentence *Sara believes women to be successful*. The strength of expectation is varied from high to low for an NP complement of the verb, which has an NP/IP subcategorization. The number of iterations required to initially attach the NP increases, and the number of iterations required to attach the IP decreases, with decreasing strength of the NP expectation.

it a weaker competitor. The competitive model therefore not only accounts for direct lexical preference effects, but predicts indirect effects on the competing attachments.

### Interaction of Preferences

It is interesting to note that in the competitive attachment model, the Minimal Attachment and Late Closure preferences are quite different in quality from the recency and lexical strength effects. Parsing models use a variety of strategies to account for the fact that the human parser prefers structures that are compatible with what we have been describing as Minimal Attachment and Late Closure; what many of these approaches have in common is that the parser is faced with a choice of structures that it somehow ranks (for example, Frazier, 1978; Gibson, 1991; Gorrell, 1987; McRoy & Hirst, 1990). In these models, for example, at the point of processing the post-verbal NP in sentences (7.1) and (7.4), the parser chooses between attaching the NP as the object of the verb phrase, or as the subject of a sentential phrase. By contrast, in the competitive attachment model, these "preferences" are in fact not preferences at all from the point of view of the parser, because there is no relevant choice. What we externally observe as a preference between two possibilities is caused by an absolute condition in which only one of the possibilities exists at the processing point in question.

In contrast, the properties of recency and lexical strength lead to true relative preferences, and thus we can investigate how their interaction is resolved in the model. This will be demonstrated by examining how the tendency for the most recent attachment to win is affected by varying the lexical strengths on more and less recent attachment possibilities. Consider again the recency example repeated here and in Figure 7.7:

(7.7) Sara believes the report that women suspect Laika to...

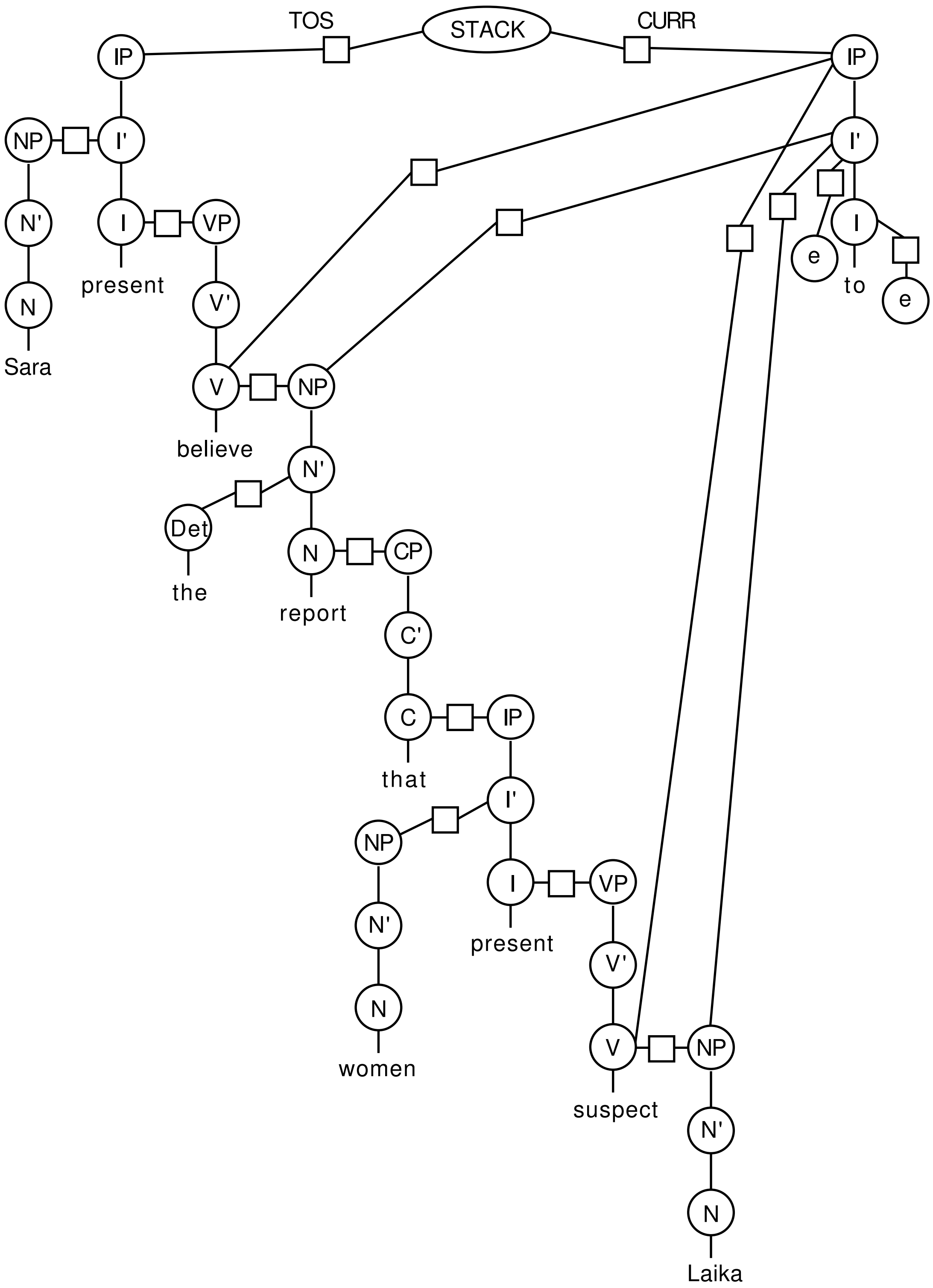


Figure 7.7: The network after projecting the post-verbal IP.

| Strength of Expectation for Each Complement | | | | Number of Iterations to Make IP Attachment | |
|---|---|---|---|---|---|
| of *believe* | | of *suspect* | | | |
| NP | IP | NP | IP | to *believe* | to *suspect* |
| 1.0 | 1.0 | 1.0 | 1.0 | | 31 |
| 1.0 | 1.0 | 1.0 | .8 | | 32 |
| 1.0 | 1.0 | 1.0 | .6 | | 34 |
| 1.0 | 1.0 | 1.0 | .4 | | 37 |
| 1.0 | 1.0 | 1.0 | .2 | 35 | |
| 1.0 | 1.0 | 1.0 | 1.0 | | 31 |
| .9 | 1.0 | 1.0 | .9 | | 31 |
| .8 | 1.0 | 1.0 | .8 | | 33 |
| .7 | 1.0 | 1.0 | .7 | | 34 |
| .6 | 1.0 | 1.0 | .6 | 37 | |

Table 7.3: The effect of varying strength of expectation on the number of iterations required to make the IP attachment at the word *to* in the sentence beginning *Sara believes the report that women suspect Laika to*. The strength of expectation is varied from high to low for NP and IP complements of verbs with an NP/IP subcategorization. The number of iterations required to attach the IP to the more recent verb *suspect* increases as its expectation for an IP decreases. When the expectation is very low, the attachment to the higher verb is made. When the expectation of the higher verb *believe* for an NP complement is simultaneously decreased, the number of iterations required to attach the IP to the more recent verb increases more rapidly, and the shift to the higher attachment is made sooner.

Table 7.3 summarizes the results of varying the strengths of expectations of the two verbs in a number of tests. Given that *believe* and *suspect* have equal strengths of expectation for NP and IP complements, the IP will attach to the most recent verb, as demonstrated earlier. If *suspect*'s strength of expectation for an IP is decreased, however, the number of iterations required for this attachment to be made is steadily increased, from 31 to 37 iterations. This again demonstrates the inhibitory effect of decreased lexical strength, which makes the preferred attachment a less strong competitor. Moreover, if the lexical strength is decreased substantially, then the IP will attach instead to *believe*, in 35 iterations. Thus, the inhibitory effect of a decreased expectation not only increases the time it takes for the network to settle on the "preferred" (more recent) attachment, it can in fact *change* the preference to the less recent attachment.

The less recent attachment to *believe* also wins (in 37 iterations) if *suspect*'s strength of expectation for an IP is decreased only moderately, but at the same time *believe*'s strength of expectation for an NP is equivalently decreased. Decreasing the strength of the NP attachment to *believe* makes that option a weaker competitor, thus increasing the ability of the IP attachment to *believe* to compete effectively. In the earlier test, when *suspect*'s strength of expectation for an IP was only .6 and *believe*'s strength of expectation for an

NP was 1.0, the IP still attached to the more recent verb, *suspect*. Here, when *suspect*'s strength of expectation for an IP was .6 and *believe*'s strength of expectation for an NP had been decreased to .6 as well, the IP attaches instead to the less recent verb, *believe*. Again, we see the indirect effects of lexical strength on the competitive attachment process.

It is worth noting that in all cases in which the less recent attachment wins, the activation of that a-node is much lower than the activation of a winning attachment to the more recent phrase. Thus, although a less recent attachment may become strong enough to win over a more recent one, there is still an element of difficulty in making the attachment, in terms of both the increased time to reach an acceptable state and the decreased strength of the resulting attachment hypothesis.

In conclusion, the competitive attachment process of the parsing model serves to smoothly integrate diverse sources of preference information. Whereas other models that rely on explicit preference heuristics must employ some means of resolving disagreements between them, the model here captures the effects in a way that obviates the need for explicit conflict resolution strategies.

### 7.2.3 Reanalysis

The final goal that was set for the parsing model was to account for the range of difficulty that the human parser exhibits in reanalyzing erroneous attachments. In this area, too, the explanations fall out from basic properties of the model. Other sentence processing models account only for the discrete division of sentences into two classes: those allowing for necessary revisions within the normal operation of the parser, and those for which a necessary revision would require a special recovery mechanism (that is, garden path examples) (for example, Gibson, 1991; Gorrell, in press; Fodor & Inoue, 1994; Pritchett, 1992; Weinberg, 1991). Fundamental properties of the competitive attachment mechanism determine the general structure of its attachment and revision operation, providing a principled explanation of these possible and impossible reanalyses. Furthermore, the competitive activation approach yields finer-grained predictions of relative difficulty within the class of possible reanalyses. In fact, recency and lexical strength, which affect relative preferences, are instrumental in determining the relative ease of reanalysis. Thus, the model provides a unifying account of the mechanisms involved in relative preferences and relative ability to reanalyze.[20]

#### Possible Reanalyses

The possible reanalyses are precisely defined in the competitive attachment model as those which involve competing attachments along the right edge of the partial parse tree on the

[20]Edward Gibson has correctly pointed out to me that it is not a necessary property of the human sentence processor that the same factors that are involved in determining the ease or difficulty of an initial preference are also involved in determining the ease or difficulty of reanalysis. Although not a necessary property, I will assume that it is a desirable one for a sentence processing model, since it entails a more uniform and parsimonious account of the data. Thus, I consider this an advantage of the competitive attachment model over other sentence processing theories.

top of the stack. Only revisions of this nature can be processed within the normal attachment operation of the parser, without recourse to special recovery strategies. This type of reanalysis has been exemplified in sentences (7.3) and (7.5). For completeness and ease of reference, Figure 7.8 shows the network state for the similar reanalysis at the word *to* in the following sentence:

(7.8) Sara believes the fact to...

As mentioned above, the network is able to revise the NP-to-V attachment represented by a-node $a_1$ because the competitions of a-node $a_2$ and $a_3$ draw sufficient activation away from it. In effect, the new attachments between the IP and the V, and the I′ and the NP, conspire to break the old attachment between the V and the NP. The number of iterations required for this network configuration to settle is 24, which is longer than the 17 iterations required for the simpler initial NP attachment to the verb, but not indicative of great processing difficulty.[21] Also, the activation of the IP complement a-node is .637 and the activation of the NP subject a-node is .534, which are both high levels for newly activated attachments. Thus, a reanalysis of this type, which involves attachments to a recent portion of the right edge of the parse tree, are not only possible but are fairly easy for the parser.

Recall that the restriction of making attachments only to the right edge of the tree on the top of the stack is independently motivated by properties of the competitive activation mechanism of the parser.[22] Thus, the ability to make the necessary revisions in this type of structure follows from fundamental properties of the competitive attachment architecture. Interestingly, the structural configurations in which the competitive relations allow reanalysis are very similar to those captured by explicit restructuring strategies in other approaches. For example, the revisions within the model conform to the restrictions imposed by Pritchett's (1992) "On-Line Locality Constraint," Fodor & Inoue's (1994) "Steal" operation, and Abney's (1989) right-edge continuation heuristic. However, in the competitive attachment model, these restrictions are an emergent property of the general attachment mechanism of the parser, enabling the model to avoid explicit heuristics defining allowable reanalyses. The next section discusses how the model in turn prohibits reanalysis of garden path examples, again as the result of inherent restrictions of the competitive activation mechanism.

### Garden Path Sentences

There are two types of impossible reanalyses in the competitive attachment model. The first occurs when the attachments that would be required for the necessary revision are not available within the current set of competing attachment nodes, because they involve nodes that are not along the right edge of the top of the stack. Under these conditions, the parser is completely unable to proceed, since there are no available alternative hypotheses

[21] Recall from Chapter 5 that the number of iterations for the network to settle ranged from 10 to 70 in the numeric simulations.

[22] See the discussion in Section 4.1.3.

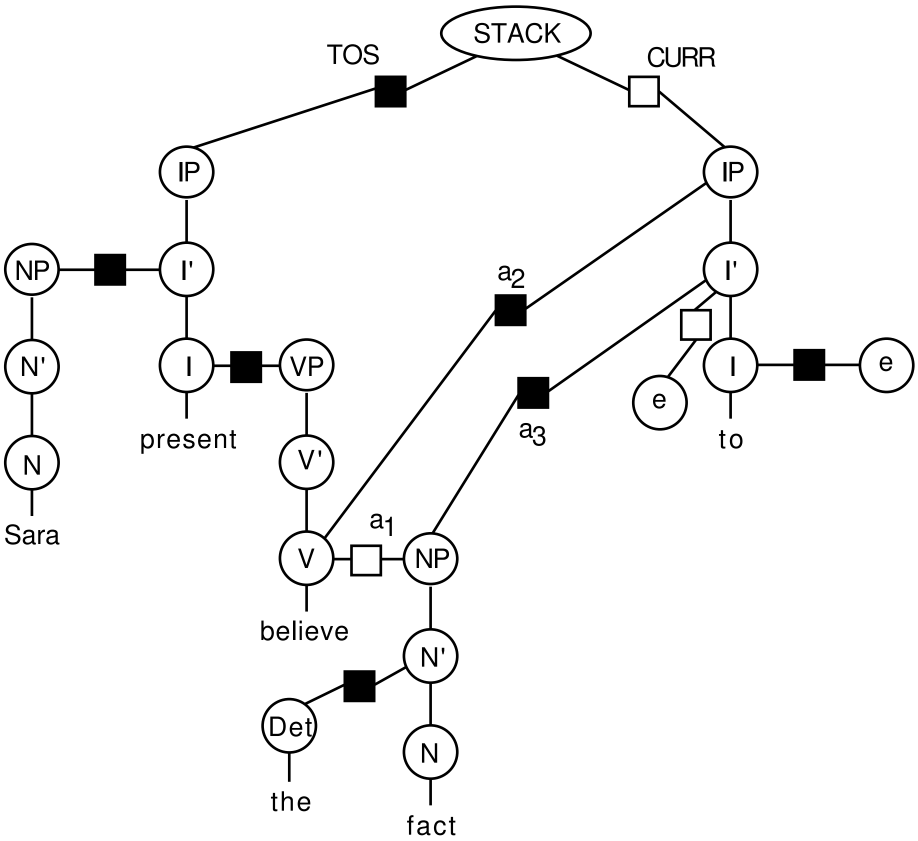


Figure 7.8: The network after reanalyzing the initial attachment for the post-verbal NP in response to the word *to* in the sentence beginning *Sara believes the fact to*.

for it to consider. Because the restriction on the allowable attachments is independently motivated, the unavailability of the correct analysis in these types of garden paths follows from fundamental properties of the competitive attachment architecture. This situation arises in the following classic garden path sentence:

(7.9) The horse raced past the barn fell.

Figure 7.9 shows the state of the parser after projecting the input phrases corresponding to *raced*; note that the tense features of the verb project a CP/IP pair of phrases. The parser has a choice between attaching the NP headed by *horse* as the subject of the IP, or attaching the CP to the NP as a reduced relative clause.[23] The attachment of the NP to

[23] A real solution to this example relies on an implementation of lexical ambiguity, since *raced* projects two possible phrases, as a main verb and as a passive participle. Chapter 8 discusses how the model can be extended to handle competing lexical alternatives. Given those extensions, the main verb alternative wins the competition, and the reduced relative reading becomes inactive and therefore inaccessible for later revision. The analysis given here can be considered a simplification of the more complete approach.

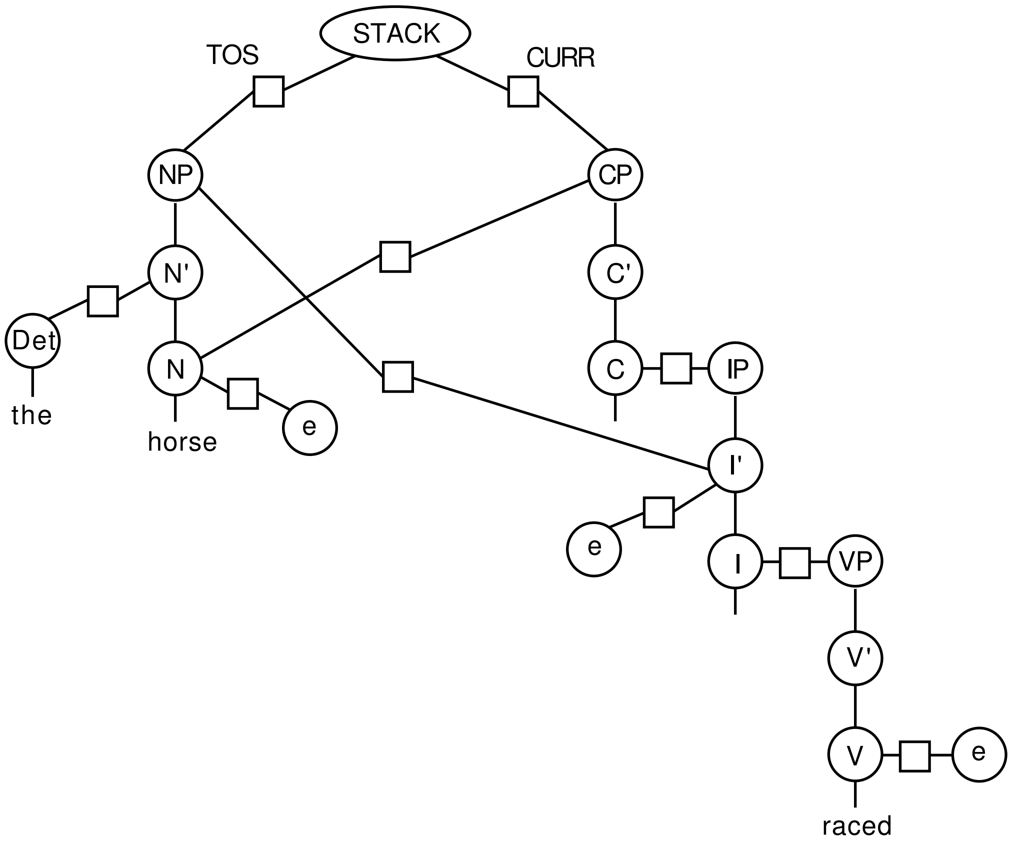


Figure 7.9: The network after projecting *raced* in the sentence beginning *The horse raced.*

the I′ is strongly preferred over the attachment of an empty node to the I′ because an empty node would currently be unbound in that position. The attachment of the CP to the N is not strong enough to win its competition because it does not receive a theta role.[24] Thus, the attachments corresponding to the main verb reading—that is, the attachment of the NP as the subject of the IP and the attachment of the CP to the stack—win the competition. When the verb *fell* is input, the parsing network is as shown in Figure 7.10. To successfully incorporate the input phrase would require having access to a-node $a_1$ between the previously attached NP and I′, so that the NP can be correctly reattached as the subject of the main verb *fell*. However, the a-nodes representing the attachments that need to be revised are not visible to the current input phrase. Because there is no grammatical analysis available to it, the parser is severely garden-pathed at this point.[25]

[24]The NP does not assign a theta role to a relative clause; the CP is instead licensed through the process of predication.

[25]Stevenson (1993b) discusses additional severe garden-path examples that depend on extensions to the parser to handle double-object verbs.

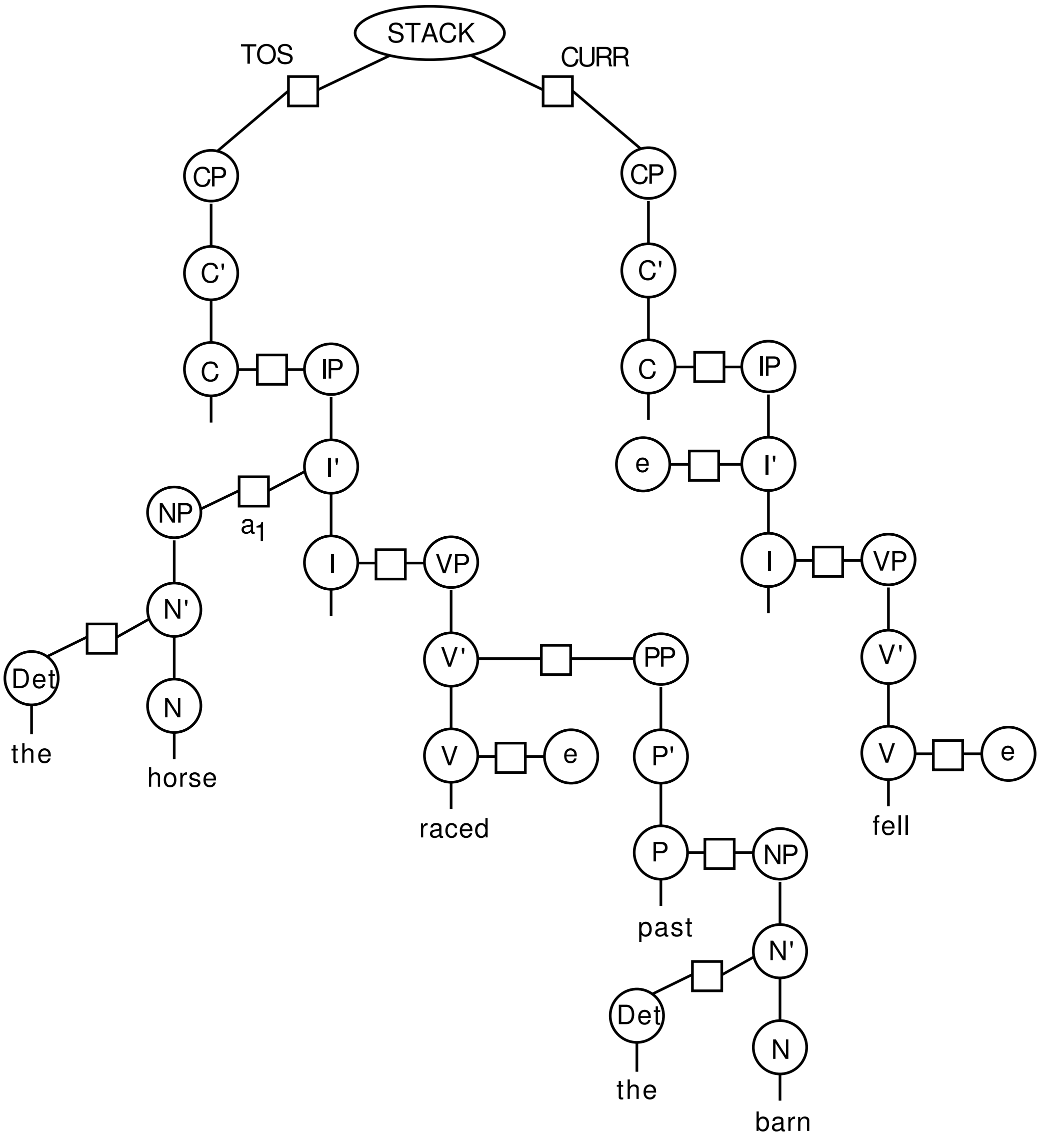


Figure 7.10: The network after projecting *fell* in the sentence *The horse raced past the barn fell*.

In the second type of impossible reanalyses, the necessary revision *does* involve a-nodes that are along the right edge of the top of the stack, but the competitive behavior of the parser prevents the a-nodes representing the correct analysis from becoming activated. This situation occurs in the non-preferred continuation of sentence (7.4):[26]

(7.10) When Kiva eats food disappears.

[26]See footnote 16 on page 125 for discussion regarding the grammaticality of example (7.10).

We saw above that the initial Late Closure preference for the post-verbal NP to attach directly to the verb occurred for the same reasons as in the Minimal Attachment example of (7.1). However, while this initial preference is revisable in the Minimal Attachment case, it is not in sentence (7.10). The initial attachment of the post-verbal NP to the verb is shown in Figure 7.5 on page 126; at *disappears*, the NP needs to break its attachment to the verb and reattach as the specifier of the current IP. The network could do this by deactivating a-node $a_1$ and activating a-node $a_3$ of Figure 7.11. However, $a_3$ is not able to win the competition with $a_1$ for the output activation from the NP. The difference from the case in Figure 7.8 (page 135) where reanalysis is possible is that there, the a-node between the verb and the NP (a-node $a_1$) was competing with two new attachments (a-nodes $a_2$ and $a_3$), which were together able to draw activation from the NP-to-V attachment. Here there is no corresponding a-node $a_2$ for the V node to redirect its output to, and so it continues to activate the NP attachment. The attachment of the NP to the I′ is not strong enough by itself to win the competition with the attachment of the NP to the V. The current I′ thus activates the default empty node attachment, leading to a clause with an empty (and unbound) subject. Since the network settles on an irrecoverably ungrammatical analysis, the model correctly predicts a garden path.[27]

However, this garden path situation differs in an important respect from the one above. In example (7.9), the necessary attachments are simply unavailable to the parser, leading directly to its inability to make the revision. Because there are no alternative hypotheses to return to, adding a simple recovery mechanism to the parser to give it a "second chance" to find a better set of attachments would not help it to recover from its failure—the parser has no recourse but to re-parse the sentence. In example (7.10), though, the necessary attachments *are* in the current competing group of attachments; they just are not strong enough to win the competition. Thus we would expect that the parser could recover more easily from its misanalysis; and in fact, there is consensus in the psycholinguistic community that this is an easier sentence than the quite difficult example (7.9).

### Reanalyses of Intermediate Difficulty

Within the class of possible reanalyses, there is also a range of difficulty. Experiments have supported the hypothesis that if the length of an ambiguous region is increased, reanalysis becomes more difficult. For example, increasing the length of the post-verbal NP in a sentence like example (7.8) leads to longer per-letter reading times in the disambiguating region than in the version of the sentence with a short post-verbal NP (Frazier & Rayner, 1982). The straightforward interpretation is that longer reading times correspond to more difficult reanalysis. Other models of human parsing have not provided a principled explanation of

[27]Frazier & Rayner (1982) provide experimental evidence for increased reading times at the disambiguating point in these types of sentences. In the competitive attachment model, the increased reading time would result from the detection of the ungrammaticality and the triggering of processing routines to recover from the garden path situation. However, the necessary recovery mechanism has not been built into the current parser.

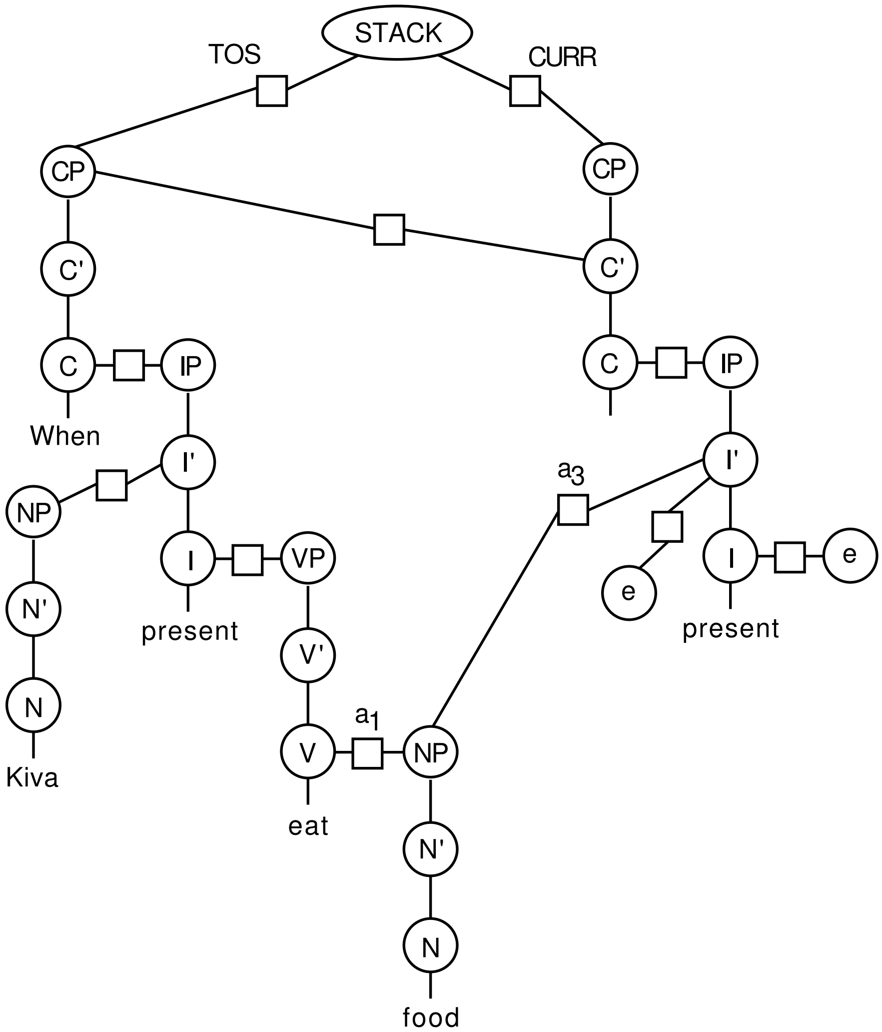


Figure 7.11: The network after projecting the main clause CP and IP in the sentence *When Kiva eats food disappears.*

this phenomenon of the duration of an ambiguity affecting the difficulty of revision (Gibson, 1991; Gorrell, in press; Fodor & Inoue, 1994; Pritchett, 1992; Weinberg, 1991).

Here again we can observe the recency effects that fall out from the competitive attachment model. Consider the following sentence, which is the same as example (7.8) but with a longer post-verbal NP:

(7.11) Sara believes the fact that women raced to...

Since the embedded verb *raced* is unable to license a sentential complement, the IP projected from *to* must attach to the main verb *believe*. Thus, the parser is faced with exactly the same attachment choices as in example (7.8). The length of the NP [*the fact that women* *raced*] means that *to* must attach higher in the tree than after a short NP like [*the fact*]; compare Figure 7.12 to Figure 7.8 on page 135. The attachment of the IP here takes 31 iterations, compared to 24 iterations in the "short NP" case of example (7.8). Furthermore, the attachments get less activation than the attachments to the more recent phrases—here the complement attachment has an activation level of .568 and the specifier attachment has an activation level of .509, compared to the corresponding values .637 and .534 for example (7.8).

Thus, the memory management techniques of the model that lead to a general recency preference also provide an explanation for these reanalysis cases of intermediate difficulty. The effects of lexical strength can also contribute to making a possible reanalysis more or less difficult; these effects were demonstrated on example (7.6) of Section 7.2.2 above. In conclusion then, the competitive behavior of the model provides an account of the range of difficulty of reanalysis, incorporating precisely the same factors as affect preferences (that is, recency and lexical strength).

## 7.3 Summary of Results

The competitive attachment model has been shown to provide an explanatory account of a range of psycholinguistic observations relevant to the human processing of syntactic ambiguities. The model explains the interesting mix of data that supports conflicting conceptualizations of the human parser as a serial or parallel processor. The constraints on the network structure lead to the single-reading preferences cited as evidence for a serial model. The competitive spread of activation through the restricted set of attachment possibilities conforms to the observed eye-movement patterns that have also been used to bolster the serial hypothesis. The active expectations in the form of simultaneously competing a-nodes underlie the syntactic priming phenomenon that strengthens the claim of parallelism. Thus, the distributed parallel approach which relies on the competitive focusing of activation provides a more complete and parsimonious account of the set of on-line processing data concerning serialism and parallelism in parsing.

Choosing a single attachment structure to maintain relies only on the underlying properties of the competitive attachment model. Fundamental properties of the competition mechanism and the lack of top-down precomputation yield a preferred reading for an ambiguous input that conforms to Minimal Attachment and Late Closure, without the use of explicit preference heuristics. Moreover, the very same properties are responsible for both of these observed preferences, giving them a unifying account for the first time. The memory management techniques of the model have the side effect of producing another preference, that of attachment to more recent structures, again obviating the need for an explicit structuring strategy. The use of competition-based spreading activation with weighted connections accounts for lexical preference effects in a natural way. The use of lexical strengths within the competitive attachment model leads to indirect effects of lexical expectations, an issue that has been previously unaddressed. The spreading activation approach also provides for a natural integration of relative preference effects such as recency and lexical strength, with

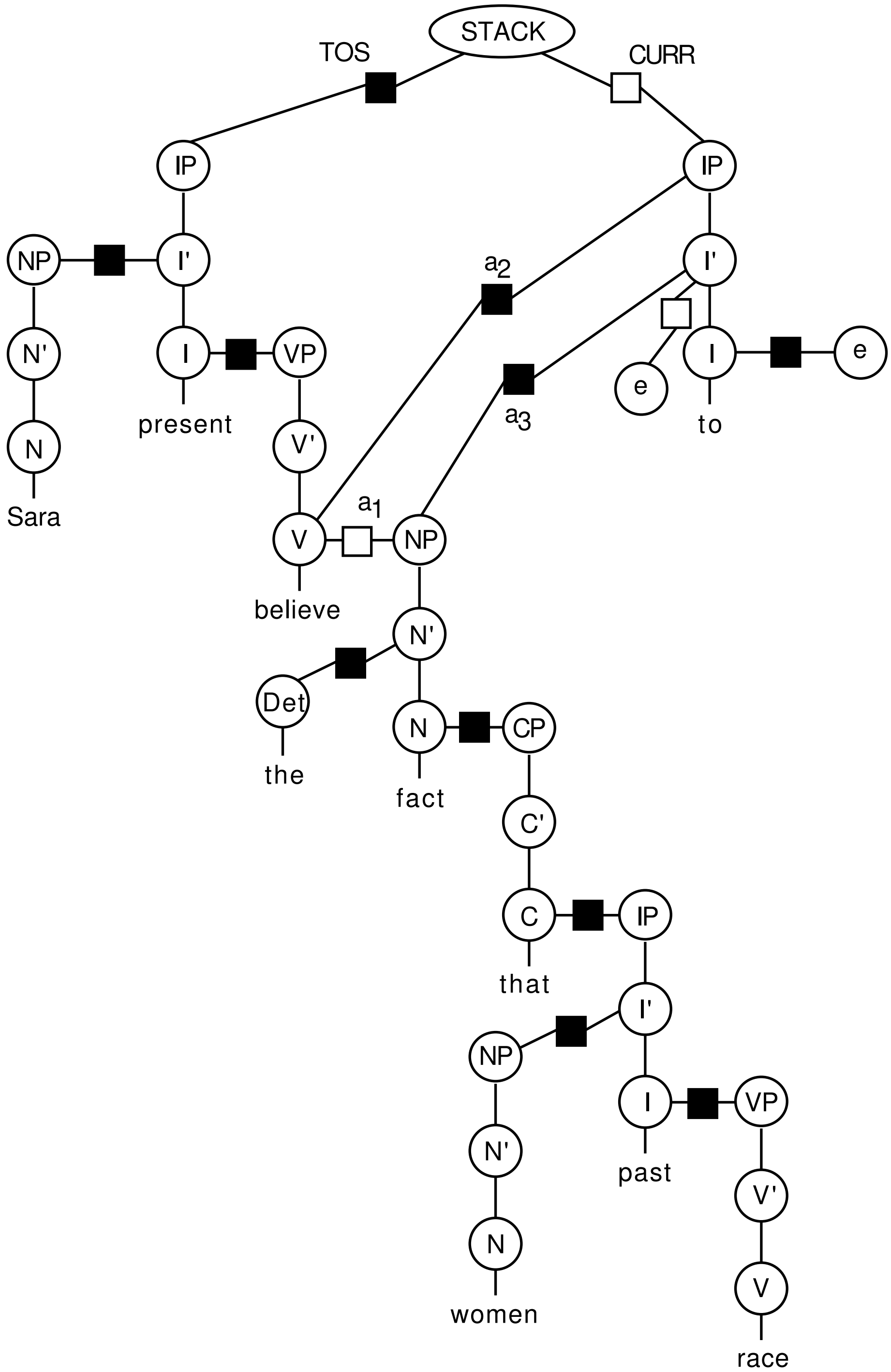


Figure 7.12: Reanalyzing the initial attachment for the post-verbal NP takes longer when the NP is longer, as in *Sara believes the fact that women raced to....*

the results demonstrating that variations in lexical strength can affect the preference for recent attachments.

These same relative preference factors affect how easily the model can revise its original attachments, and contribute to an account of reanalysis that explains fine-grained acceptability. The model defines easy reanalyses to be those that fall within the normal competitive attachment operation of the parser, which can revise pre-existing attachments under strictly set conditions. These reanalyses become harder as they involve less recent phrases or weaker lexical expectations, leading to an account of reanalyses that are of intermediate difficulty. The model also explains both severe and "milder" garden path sentences as ones that would require recourse to special recovery procedures in the parser. In the milder garden paths, the alternative analysis is available but is unable to win its competition. In the severe garden paths, the alternative is unavailable, due to independently motivated restrictions on the network structure, and the parser must reprocess the sentence.

This set of results taken together provides a comprehensive picture of the computational processes that underlie the processing of syntactic ambiguity in the human parser. The chapter began with a statement of the following open questions that must be answered in order to adequately characterize these computational processes:

1. **Serialism vs. Parallelism**: When presented with an ambiguity, does the parser build and maintain a single structure or multiple structures?

2. **Structural Preferences**: How does the parser determine the preference for one possible structure over another?

3. **Reanalysis**: If the continuation of the input is incompatible with the preferred structure, how easily is the parser able to revise its initial hypothesis?

In answer to these questions, the results of the competitive attachment model support the following hypotheses concerning the human process of syntactic ambiguity resolution. When faced with a syntactic ambiguity, the parser considers a set of multiple alternative analyses in parallel, but immediately chooses a single preferred analysis to maintain. The choice is made by a restricted competition mechanism that indirectly constrains the set of attachments that are considered by the parser, as well as determining the preferences among those attachments. The process of choosing the preferred attachments involves the competitive distribution of activation through a restricted network structure. The parser can change an attachment only if the revision comprises one of the subsets of attachments allowed by the competition mechanism. Reanalysis involves re-directing activation away from previously preferred attachments and activating new attachments. Because reanalysis is performed within the normal competitive attachment process of the parser, the difficulty of allowable revisions is determined by the same factors as affect the difficulty of initial attachments, such as recency and lexical strength.

## Chapter 8

# Conclusions

This dissertation has described a natural language parsing architecture whose emergent behavior mimics that of the human parser in the processing of syntactic ambiguities. The model is a new type of hybrid connectionist architecture that relies on an integration of a novel approach to marker-passing with extended techniques for controlling the distribution of numeric activation. The marker-passing method enables the establishment of syntactic relations based solely on local communication in a network, without the use of phrase structure rules. This technique supports distributed parsing using simultaneous declarative constraints, as specified by recent linguistic theories. The model also relies on extended techniques for achieving effective competitive activation in a complex domain. These two hybrid connectionist techniques together yield a distributed natural language parsing mechanism that uses only simple and uniform processing nodes.

The competitive attachment model incorporates a general and parsimonious account of a wide range of human behavior in syntactic ambiguity resolution. The behavior of interest emerges from the interaction of fundamental properties of the model, avoiding the need for construction-specific processing strategies. The competition mechanism applies uniformly at all processing nodes to explain a number of human structural preferences in parsing. In addition, the model's unique parallel attachment operation applies across general structural configurations to account for the ease or difficulty of revisability in parsing. The results of the model underscore the importance of determining computational mechanisms that can explain an extensive range of human behavior, rather than simply specifying heuristics that capture individual surface observations.

The approach to parsing developed here has interest within the field of Artificial Intelligence for two interdependent reasons: as a theoretical model exploring plausible mechanisms of human syntactic processing, and as a step toward mimicking human performance in NLU systems. More broadly, the research makes a number of contributions to the development of computational techniques for achieving intelligent behavior in a massively parallel network architecture. Furthermore, the model provides a solid framework for the investigation of additional open problems, both in the control of competitive dynamics within complex network applications, and in the automatic processing of syntactic structure and linguistic ambiguities.

## 8.1 Contributions

Recall the three fundamental computational assumptions of the model: (1) The basic architecture is that of a hybrid connectionist network integrating symbolic and numeric computation. (2) Numeric decision-making is focused through competition-based spreading activation (CBSA), and no inhibitory links are used. (3) The network is established through dynamic instantiation of generic template nodes, and top-down hypothesizing of structure is prohibited. Each of the three fundamental assumptions of the model relies on computational advances in connectionist modeling, which interact to yield interesting linguistic behavior.

### 8.1.1 Hybrid Architecture

In contrast to previous parsers implemented within a connectionist framework, the grammatical knowledge of the competitive attachment model is a subset of a well-founded linguistic theory. Since linguistic theories describe the knowledge underlying human language abilities, it is important to develop computational structures and mechanisms that are sufficient to encode and process these descriptions. A recent advance in linguistics is the description of human knowledge of language in terms of a small number of simple, interacting constraints on syntactic structure, replacing the use of large numbers of phrase structure rules. While other connectionist parsers have captured rule-based grammatical knowledge by *a priori* structuring of the parsing network, with a constraint-based syntactic theory such an approach is not an option. In order to achieve a faithful implementation of a constraint-based theory, it was necessary to develop a hybrid approach to network processing. The goal was to support a sophisticated level of syntactic processing, while retaining the computational advantages of a massively parallel architecture.

The symbolic capabilities of the competitive attachment approach were limited to the creation and comparison of simple symbolic features with atomic values. Because decision-making in the network is distributed among a set of highly restricted processing units, syntactic attachments must be determined solely through the local communication among the nodes of these simple features. Thus, to establish valid syntactic relations among the syntactic processing nodes, a novel form of feature-passing was devised that enforces the structural restrictions imposed by the linguistic theory. Communication of symbolic features is restricted by the grammatical properties of the syntactic nodes. The structural constraints of the grammar were analyzed into computational primitives that could be verified entirely between neighbors along a feature-passing path in the parsing network. A node's decision to output a feature packet that it has received depends solely on the values of features that are local to that node. Using this communication method, the parsing network is able to enforce even long-distance grammatical constraints entirely through local, distributed interactions among the syntactic nodes.

### 8.1.2 Competitive Network Dynamics

One of the potential advantages of competition-based spreading activation (CBSA) is its suitability for use within computational models of higher-level cognitive abilities (Reggia, Marsland, & Berndt, 1988). However, to date the use of CBSA within these types of applications has been limited. Because its restrictiveness suited the goals of the parsing model developed here, CBSA was chosen to serve as the sole mechanism for focusing activation within the parsing network onto a consistent syntactic structure. The restrictiveness of the approach, in contrast to the use of inhibitory links for the same purpose, entails that CBSA plays a vital role in constraining the computational architecture in critical ways.

Competition is crucial in the model for choosing between structural alternatives. In order to use CBSA alone to make decisions in the parser, the connections among the processing nodes must be limited to enable the competitive relations to propagate effectively through the resulting network structure. The model employs a stack to structure the syntactic input and limit the connectivity of the network. The stack can be represented by a single processing node that has uniform capabilities with the other syntactic processors. With this simple addition to the parsing network, the use of CBSA alone is sufficient to determine the attachment decisions. The competitive attachment behavior of the network was highly successful in the nearly 1400 tests that were run: the network converged in over 98% of the simulations, achieving correct and consistent attachment behavior in all cases. Furthermore, the restrictions on the parser that are motivated by CBSA have additional computational advantages for the parsing model, by reducing the parser's processing load, as well as preventing a combinatorial explosion of attachment possibilities.

### 8.1.3 Dynamic Network Creation

Because the network structure cannot be determined *a priori* on the basis of a set of syntactic phrase structure rules, the network must be built in response to each specific sentence that is input to the parser. The parsing network is constructed as words are input sequentially, by dynamically instantiating fixed phrasal templates in direct response to features of each input word. The phrasal templates are *not* phrase structure rules in the traditional sense; they encode only the connections between the phrasal nodes corresponding to a single input word (such as a V, V$'$, and VP node), and not the connections between different phrases (such as the connection from a verb phrase to its NP object). The dynamic instantiation of these fixed phrasal templates is enabled by the limited symbolic capabilities of the parser, and consists simply of determining the atomic feature values for the attributes of each node based on knowledge in the lexicon. The dynamic creation of the network solves a number of problems arising from the fixed network approach adopted in other connectionist parsers. With its dynamic allocation and reuse of phrasal nodes, the model here can parse sentences of indefinite length, while previous connectionist approaches impose an unrealistic fixed upper bound. Furthermore, the method here is more space-efficient; it can instantiate phrases from a small pool of generic templates, instead of having to resort to massive duplication of a number of dedicated node types.

Because the network is built dynamically, there must be some principled limitation on the precomputation of syntactic structure. The model takes the most restrictive approach by disallowing top-down hypothesizing of phrase structure. Phrasal templates can be instantiated only by direct evidence in the input; a reasonable expectation for a phrase is not enough. Constraining parallelism to nodes with overt evidence leads to better scale-up potential in the model. The number of syntactic nodes is kept to a minimum, and therefore the number of attachments which must be considered at any particular point in the parse is also reduced. Thus, both the use of generic phrasal templates and the prohibition on top-down precomputation contribute to the computational feasibility of the model.

### 8.1.4 Competitive Attachment Behavior

Human-like behavior in the processing of syntactic ambiguities emerges from the interaction of the computational developments discussed above. The results of the competitive attachment parser in modeling human performance fall into three major areas. First, the model incorporates a principled mix of serial and parallel processing behaviors. These enable it to match human expectations in its determination of syntactic preferences, as well as in the accessibility of alternative structural analyses to pursue. Second, the attachment preferences that it exhibits arise from the underlying properties of the competitive attachment process, which successfully integrates interacting preferences in a uniform manner. The attachment decisions in the model conform to Minimal Attachment, Late Closure, recency, and lexical preferences, without those preferences being explicitly built in. Third, the performance of the model mimics that of people in revising erroneous attachments, while avoiding the use of explicit revision strategies as well. The degree of difficulty of reanalysis in the model in fact results from the same properties that determine structural preferences. The competitive attachment approach thus provides a unifying and parsimonious model of human behaviors across the entire process of syntactic ambiguity resolution.

## 8.2 Future Work

Many issues must be explored in future research in order to extend the competitive attachment framework to a more complete approach to natural language understanding. This section will discuss a number of promising directions for extensions to the model. First, the approach must be evaluated with respect to additional types of syntactic ambiguity, including lexical ambiguity and argument/adjunct ambiguity. In addition, the effect of semantics on syntactic decision-making should be addressed. Second, the basic competitive parsing techniques must be generalized to provide a uniform syntactic processing mechanism that can subsume a broad range of parsing functions. Third, the model must be subjected to a cross-linguistic investigation of both the linguistic adequacy and the processing behavior of the model.

### 8.2.1 Other Types of Ambiguity

#### Lexical Ambiguity

The research here focused on subcategorization ambiguities; that is, ambiguities arising from a verb's ability to take different kinds of objects. Another common type of syntactic ambiguity is lexical ambiguity, in which a word has more than one potential syntactic category. For example, the word *have* may be an auxiliary verb or a main verb, as in *Have the children taken the exam?* and *Have the children take the exam.* Resolving lexical ambiguity requires extending the competitive attachment process from only operating on competing attachments to encompass competing phrases. In the current model, an input word activates a single phrase by sending it a fixed activation. In the extended model, an ambiguous input word would need to activate all the possible structural choices that arise from it, dividing its output in a competitive way that forces a structural decision. Thus, an input word would activate multiple syntactic phrases competitively, in exactly the same way that a phrasal node currently activates attachments competitively. In the example above, the word *have* would create phrasal structures corresponding to its auxiliary and main verb readings, and these structures would compete for its activation.

Because of its inherent competitive dynamics, the model has the potential to explain human behavior in processing a lexical ambiguity over the course of time—behavior that other sentence processing models cannot currently account for. Lexical ambiguity presents the parser with a choice of structures; in order to minimize the amount of structure that they must maintain, all parsing models, serial or parallel, must use some method for pruning out the least likely choices. In other models, the decision to maintain or discard particular structures is made too early—right at the ambiguous word (for example, Gibson, 1991; Weinberg, 1991). This approach often models human behavior incorrectly, in one of two ways: by maintaining multiple structures when people appear not to, or by prematurely discarding the correct choice when people in fact maintain it. The competitive attachment process is qualitatively quite different, because it inherently incorporates an aspect of processing over time that is missing in other models; it thereby avoids the problem of committing too early to a decision to maintain or discard structures. The model thus has the potential to match a fuller range of human behavior in the processing of lexical ambiguities.

However, extending the model in this way poses a clear computational challenge, due to the necessary incorporation of additional competitive effects in the parsing network. The model will have to involve simultaneous and interacting competitive processes: the competition among the phrasal choices for an ambiguous word and the competition among attachments to those structural alternatives. Controlling these types of interacting competitive processes with competitive activation has not been attempted before. Thus, work in this area will provide a stringent test of whether competitive activation can live up to its promise of being a cognitively plausible technique for focusing activation within a network.

### Argument/Adjunct Preferences

In the sentence beginning *Ann put the candy on the table*, the attachment of the prepositional phrase *on the table* is ambiguous: it can attach to the verb phrase as the location argument of the verb *put*, or to the noun phrase as an adjunct modifier of the noun *candy*. In choosing between an argument and adjunct attachment of this type, the human parser shows a strong preference for making the argument attachment. This preference is so strong, in fact, that if, in this example, the prepositional phrase turns out to be a modifier of the noun instead, as in *Ann put the candy on the table into her mouth*, people experience difficulty processing the remainder of the sentence. Accounting for the argument attachment preference shown by the human parser has been one of the goals of every sentence processing model proposed, and yet to date, an adequate explanation of this phenomenon has not been achieved.

The competitive attachment model currently is able to parse only argument attachments; in all cases, a phrasal node must activate a fixed number of attachment nodes. Currently this "fixed number" is always one; extending the technique to accommodate other values is straightforward, as long as the number is fixed for each node. However, in extending the model to parse adjuncts, their attachment sites cannot be allocated in the same fashion. Adjuncts are always optional, and there may be a highly variable number of them modifying any given word or phrase. Thus, adjunct attachments in the competitive attachment parser would rely on an attachment site having competitive properties that allow zero or more attachments. The challenge for the approach is to find a set of well-defined and stable adjustments to the competitive activation functions that allow the desired behavior of activating zero or more attachments. An advantage of the competitive activation function is that it has well-defined parameters for experimentally varying the degree of competition induced. Experiments with the model will provide an interesting application for determining the flexibility and adequacy of the competitive activation process.

### Semantic Effects

Many of the example ambiguities that previous NLU research has focused on have been cases of ambiguous prepositional phrase attachments that are resolved by the semantic context. The current restriction of the competitive attachment model to syntactic knowledge prevents it from making testable predictions concerning the effect of context on the resolution of attachment ambiguities. However, previous work in semantic effects on disambiguation indicates that the massively parallel network style of the parser lends itself well to extension in this area (see, for example, Cottrell, 1989; Hirst, 1987). In fact, the competitive attachment approach has the potential to allow lexical, semantic, and discourse preferences to come into play without additional provisions or changes to the basic mechanism. The ability of the competitive activation mechanism to integrate diverse sources of preference information was already demonstrated. The model would serve as a testbed for an "interactive" approach, in which semantic information is able to affect syntactic attachment decisions.

### 8.2.2 Representational Adequacy

The linguistic theory of Government-Binding (GB) marks a radical departure from prior theories of human syntactic knowledge in its move from rule-based to constraint-based linguistic descriptions. However, most GB parsers have assumed fairly traditional parsing mechanisms, replacing the process of rule reduction with that of licensing attachments according to GB constraints (Abney, 1989; Fong, 1991; Gibson, 1991). One of the goals of this dissertation was to build a computational model of parsing that mirrored the move within GB from rule-based to constraint-based linguistic descriptions of syntactic knowledge with a corresponding shift from rule-based to connectionist computational processing techniques. This was accomplished with the use of a restricted message-passing procedure and the use of competitive attachment nodes. In the current implementation of the model, these methods are used solely to establish attachment relations in the parser. If the techniques can be extended to handle *all* syntactic relations, the model would have clear conceptual advantages over previous GB parsers, in which a system of numerous rules has been replaced by a system of numerous licensing mechanisms. The competitive attachment model has the potential to offer a uniform processing mechanism for all of these parsing responsibilities, truly exploiting the statement of GB as a simple system of constraints.

The competitive attachment mechanism must be extended to the other major syntactic relations of GB theory, including the process of coindexation, and the assignment of theta roles and Case.[1] This means not only passing the features appropriately through the network, but establishing competitive "binding" nodes corresponding to *all* syntactic relations. The resulting model would be quite elegant, using only two distinct node types—one type for syntactic phrases and one type for binding those phrases in some syntactic relation. These two types of nodes are required in pure connectionist parsers as well. One of the potential problems in maintaining this uniform competitive binding approach is an increase in network complexity. Since every syntactic relation would involve a competitive binding node, the network would consist of a number of competitive relations which interact and affect each other. One conceptually attractive solution is to layer the network, so that each type of relation has its own layer, and the layers are mediated by links to the layer that encodes the attachment structure (the parse tree). There could be limited, controlled interaction between the layers. Such a multi-layered approach would tie in well with the addition of a layer of interacting semantic information.

### 8.2.3 Cross-Linguistic Representation and Behavior

A longer-term goal is to extend the competitive attachment model to apply to languages other than English. Cross-linguistic investigation of the model will provide a harsh test of the computational theory of parsing in terms of both the adequacy of its syntactic knowledge representation and its ability to achieve human-like behavior. Recently, debates on the

[1]Stevenson (1993a) discusses extensions to the model that incorporate the competitive establishment of coindexation relations.

best approaches to parsing—bottom-up, top-down, head-driven, left corner—have recognized the need to consider the structural properties of vastly different languages. The hybrid architecture developed here is most compatible with an approach that is *evidence-driven*; that is, the parser, within its domain, will be smart and efficient, making use of the information it has at any given point in the parse, to build as much structure as it can.

This type of model appears quite promising for head-final languages like Dutch and Japanese, where the word that determines the propositional content of a phrase occurs after its arguments. Similar approaches have been put forward by others (Crocker, 1992; Inoue & Fodor, in press), but the restricted parallelism of the model developed here has great advantages, by avoiding both the problem of overgeneration within a parallel approach and the limitations of strict serialism. Other models either must devise pruning mechanisms for controlling the number of alternatives maintained in parallel, or must add on explicit revision strategies to an essentially serial parser. Tuning these extra parsing mechanisms to different languages can be problematic.

In addition to basic structural differences between languages, it appears that some parsing preferences may remain constant across languages, while others vary (Gibson et al., 1993). It is an open question whether the observed differences are due to frequency effects in the languages, or result from some more fundamental structural properties. An advantage of the competitive attachment model is that its hybrid symbolic/numeric nature lends itself well to an investigation of both qualitative statements of processing differences between languages, as well as statistical or frequency-based effects.

# References


Abney, S. (1989). "A Computational Model of Human Parsing." *Journal of Psycholinguistic Research* **18:1**, 129–144.

Abney, S. (1986). "Functional Elements and Licensing." GLOW Conference, Gerona, Spain.

Abney, S., and J. Cole (1985). "A Government-Binding Parser." *Proceedings of NELS 16*, 1–17.

Altmann, G. (1988). "Ambiguity, Parsing Strategies, and Computational Models." *Language and Cognitive Processes* **3:2**, 73–97.

Altmann, G. and M. Steedman (1988). "Interaction with Context During Human Sentence Processing." *Cognition* **30**, 191–238.

Anderson, J. (1983). "Cognitive and Psychological Computation with Neural Models." *IEEE Transactions on Systems, Man, and Cybernetics* **5**, 799–815.

Berwick, R. (1982). "Locality Principles and the Acquisition of Syntactic Knowledge." Doctoral dissertation, MIT.

Boland, J. (1991). "The Use of Lexical Knowledge in Sentence Processing." Doctoral dissertation, University of Rochester.

Cardie, C., and W. Lehnert (1991). "A Cognitively Plausible Approach to Understanding Complex Syntax." *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 117–124.

Carlson, G., and M. Tanenhaus (1988). "Thematic Roles and Language Comprehension." *Syntax and Semantics* **21**, 263–288.

Chalmers, D. (1992). "Transformations on Distributed Representations." In N. Sharkey (Ed.), *Connectionist Natural Language Processing: Readings from Connection Science*. Dordrecht: Kluwer.

Charniak, E. (1986). "A 'Neat' Theory of Marker Passing." *Proceedings of the National Conference on Artificial Intelligence*, 584–588.

Charniak, E., and E. Santos (1987). "A Connectionist Context-free Parser Which is not Context-Free, But Then It is not Really Connectionist, Either." Manuscript, Department of Computer Science, Brown University, Providence, Rhode Island.

Chomsky, N. (1986a). *Knowledge of Language: Its Nature, Origin, and Use*. New York: Praeger.

Chomsky, N. (1986b). *Barriers*. Cambridge: MIT Press.

Chomsky, N. (1981). *Lectures on Government and Binding: The Pisa Lectures*. Dordrecht: Foris Publications.

Cinque, G. (1990). *Types of A′ Dependencies*. Cambridge: MIT Press.

Clark, R. (1988). "Parallel Processing and Local Optimization." Talk given at the University of Maryland Processing Workshop, December 9, 1988.

Cottrell, G. W. (1989). *A Connectionist Approach to Word Sense Disambiguation*. Los Altos, CA: Morgan Kaufmann.

Crain, S. and M. Steedman (1988). "On Not Being Led Up the Garden Path: The Use of Context by the Psychological Parser." In D. R. Dowty, L. Kartunnen, and A. M. Zwicky (Eds.), *Natural Language Processing: Psychological, Computational, and Theoretical Perspectives*. Cambridge: Cambridge University Press.

Crocker, M. (1992). "A Logical Model of Competence and Performance in the Human Sentence Processor." Doctoral dissertation, University of Edinburgh.

Dorr, B. (1993). *Machine Translation: A View from the Lexicon*. Cambridge: MIT Press.

Fahlman, S. (1981). "Representing Implicit Knowledge." In G. Hinton and J. Anderson (Eds.), *Parallel Models of Associative Memory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Fanty, M. (1985). "Context-free parsing in connectionist networks." Technical Report TR174, University of Rochester.

Feldman, J., and D. Ballard (1982). "Connectionist Models and Their Properties." *Cognitive Science* **6**, 205–254.

Fodor, J. A., T. Bever, and M. Garrett (1974). *The Psychology of Language*. New York: McGraw-Hill.

Fodor, J. D., and L. Frazier (1978). "The Sausage Machine: A New Two-Stage Parsing Model." *Cognition* **6**, 291–325.

Fodor, J. and A. Inoue (1994). "The Diagnosis and Cure of Garden Paths." *Journal of Psycholinguistic Research* **23**:**5**, 407–434.

Fong, S. (1991) "Computational Properties of Principle-Based Grammatical Theories." Doctoral dissertation, MIT.

Ford, M., J. Bresnan, and R. Kaplan (1982). "A competence-based theory of syntactic closure." In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations*. Cambridge: MIT Press.

Frazier, L. (1987). "Sentence Processing: Evidence from Dutch." *Natural Language and Linguistic Theory* **5**, 519–559.

Frazier, L. (1978). *On Comprehending Sentences: Syntactic Parsing Strategies*. Doctoral dissertation, University of Connecticut. Bloomington, IN: Indiana University Linguistics Club.

Frazier, L., C. Clifton, and J. Randall (1983). "Filling Gaps: Decision principles and structure in sentence comprehension." *Cognition* **13**, 187–222.

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

Gibson, E. (1991). "A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown." Doctoral dissertation, Carnegie-Mellon University.

Gibson, E. (1987). "Garden-Path Effects in a Parser with Parallel Architecture." Presented at the Eastern States Conference on Linguistics, Columbus.

Gibson, E., N. Pearlmutter, E. Canseco-Gonzalez, and G. Hickok (1993). "Cross-linguistic Attachment Preferences: Evidence from English and Spanish." Manuscript, MIT.

Gorrell, P. (in press). *Syntax and Parsing.* Cambridge: Cambridge University Press.

Gorrell, P. (1987). "Studies of Human Syntactic Processing: Ranked-Parallel versus Serial Models." Doctoral dissertation, University of Connecticut.

Hanson, S. and J. Kegl (1987). "PARSNIP: A Connectionist Network that Learns Natural Language Grammar from Exposure to Natural Language Sentences." *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 106–119.

Henderson, J. (1994). "Connectionist Syntactic Parsing Using Temporal Variable Binding." *Journal of Psycholinguistic Research* **23**:**5**, 353–379.

Hendler, J. (1987). "Marker-passing and Microfeatures." *Proceedings of the Tenth International Joint Conferences on Artificial Intelligence*, 151–154.

Hirst, G. (1987). *Semantic Interpretation and the Resolution of Ambiguity.* Cambridge: Cambridge University Press.

Howells, T. (1988). "VITAL: A Connectionist Parser." *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, 18–25.

Inoue, A. and J. Fodor (in press). "Information-paced Parsing of Japanese." In R. Mazuka and N. Nagai (Eds.), *Japanese Sentence Processing.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Jones, M. (1987). "Feedback as a Coindexing Mechanism in Connectionist Architectures." *Proceedings of the Tenth International Joint Conferences on Artificial Intelligence*, 602–610.

Jurafsky, D. (1991). "An On-Line Model of Human Sentence Interpretation." *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 449–454.

Kashket, M. (1991). "A Parameterized Parser for English and Warlpiri." Doctoral dissertation, MIT.

Kempen, G. and T. Vosse (1989). "Incremental Syntactic Tree Formation in Human Sentence Processing: A Cognitive Architecture Based on Activation Decay and Simulated Annealing." *Connection Science* **1**:**3**, 273–290.

Kimball, J. (1973). "Seven Principles of Surface Structure Parsing in Natural Language." *Cognition* **2**:**1**, 15–47.

Kimura, K., T. Suzuoka, and S. Amano (1992). "Association-Based Natural Language Processing with Neural Networks." *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, 224–231.

Kwasny, S. and K. Faisal (1992). "Connectionism and Determinism in a Syntactic Parser." In N. Sharkey (Ed.), *Connectionist Natural Language Processing: Readings from Connection Science*. Dordrecht: Kluwer.

Lin, D. (1993). "Principle-Based Parsing without Overgeneration." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 112–120.

MacDonald, M. (1994). "Probabilistic Constraints and Syntactic Ambiguity Resolution." *Language and Cognitive Processes* **9**, 157–201.

MacDonald, M., N. Pearlmutter, and M. Seidenberg (1993). "The Lexical Nature of Syntactic Ambiguity Resolution." The Beckman Institute, TR UIUC-BI-CS-93-13.

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MIT Press.

Marcus, M., D. Hindle, and M. Fleck (1983). "D-theory: Talking about Talking about Trees." *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, 129–136.

McClelland, J. and A. Kawamoto (1986). "Mechanisms of Sentence Processing: Assigning Roles to Constituents." In McClelland, J., D. Rumelhart, and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2*. Cambridge: MIT Press.

McClelland, J., D. Rumelhart, and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge: MIT Press.

McRoy, S. and G. Hirst (1990). "Race-Based Parsing and Syntactic Disambiguation." *Cognitive Science* **14**, 313–353.

Merlo, P. (1992). "On Modularity and Compilation in a Government-Binding Parser." Doctoral dissertation, University of Maryland. Available as Technical Report CS-TR-2982, Department of Computer Science, University of Maryland.

Milne, R. (1986). "Resolving Lexical Ambiguity in a Deterministic Parser." *Computational Linguistics* **12**:**1**, 1–12.

Milne, R. (1982). "Predicting Garden Path Sentences." *Cognitive Science* **6**, 347–373.

Norvig, P. (1989). "Marker Passing as a Weak Method for Text Inferencing." em Cognitive Science **13**:**4**, 569–620.

Paolucci, M. (1993). "Implementation of a Principle-Based Parser Based on On Line Locality Principle." Manuscript, LRDC, University of Pittsburgh.

Peng, Y. and J. Reggia (1989). "A Connectionist Model for Diagnostic Problem Solving." *IEEE Transactions on Systems, Man, and Cybernetics* **19**:**2**, 285–298.

Pinker, S. and A. Prince (1988). "On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition." *Cognition* **28**, 73–193.

Pollack, J. (1985). "On Connectionist Models of Natural Language Processing." Technical Report MCCS-87-100, Computing Research Laboratory, New Mexico State University, New Mexico.

Prince, A. and P. Smolensky (1993). "Optimality Theory: Constraint Interaction in Generative Grammar." Technical Report RuCCS TR-2, Rutgers University Center for Cognitive Science. Rutgers, the State University of New Jersey.

Pritchett, B. (1992). *Grammatical Competence and Parsing Performance*. Chicago: University of Chicago Press.

Rager, J. and G. Berg (1992). "A Connectionist Model of Motion and Government in Chomsky's Government-binding Theory." In N. Sharkey (Ed.), *Connectionist Natural*

*Language Processing: Readings from Connection Science.* Dordrecht: Kluwer.

Rayner, K. and L. Frazier (1987). "Parsing Temporarily Ambiguous Complements." *The Quarterly Journal of Experimental Psychology* **39A**, 657–673.

Reggia, J. (1987). "Properties of a Competition-Based Activation Mechanism in Neuromimetic Network Models." *Proceedings of the First International Conference on Neural Networks*, San Diego, II-131–II-138.

Reggia, J., P. Marsland, and R. Berndt (1988). "Competitive Dynamics in a Dual-Route Connectionist Model of Print-to-Sound Transformation." *Complex Systems.*

Reggia, J., Y. Peng and P. Bourret (1991). "Recent Applications of Competitive Activation Mechanisms." *Neural Networks: Advances and Applications*, 33–62.

Reggia, J., and G. Sutton (1988). "Self-Processing Networks and Their Biomedical Implications." *Proceedings of the IEEE* **76**:**6**, 680–692.

Reilly, R. (1992). "A Connectionist Technique for On-Line Parsing." *Network* **3**:**1**.

Rizzi, L. (1990). *Relativized Minimality.* Cambridge: MIT Press.

Rumelhart, D., J. McClelland, and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge: MIT Press.

Schubert, L. (1984). "On Parsing Preferences." *Proceedings of the Tenth International Conference on Computational Linguistics (COLING-84)*, 247–250.

Schubert, L. (1986). "Are There Preference Trade-Offs in Attachment Decisions?" *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, 601–605.

Sells, P. (1985). *Lectures on Contemporary Syntactic Theories: An Introduction to Government-Binding Theory, Generalized Phrase Structure Grammar, and Lexical-Functional Grammar*. Stanford: CSLI.

Selman, B., and G. Hirst (1985). "A Rule-Based Connectionist Parsing System." *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 212–219.

Shieber, S. (1983). "Sentence Disambiguation by a Shift-Reduce Parsing Technique." *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, 113–118.

Slack, J. (1991). "Hybrid Encoding: The Addressing Problem." *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 431–436.

Small, S. (1981). "Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding." Doctoral dissertation, University of Maryland, College Park.

Smolensky, P. (1988). "On the Proper Treatment of Connectionism." *Behavioral and Brain Sciences* **11**:**1**, 1–23.

Sopena, J. (1992). "ERSP: A Distributed Connectionist Parser That Uses Embedded Sequences to Represent Structure." Manuscript, University of Barcelona, Spain.

Speas, M. (1990). *Phrase Structure in Natural Language.* Dordrecht: Kluwer.

Spivey-Knowlton, M., J. Trueswell, and M. Tanenhaus (1993). "Context Effects in Syntactic Ambiguity Resolution: Discourse and Semantic Influences in Parsing Reduced Relative Clauses." *Canadian Journal of Experimental Psychology* **47**:**2**, 276–309.

Stevenson, S. (1993a). "Establishing Long-Distance Dependencies in a Hybrid Network Model of Human Parsing." *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, 982–987.

Stevenson, S. (1993b). "A Competition-Based Explanation of Syntactic Attachment Preferences and Garden Path Phenomena." *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 266–273.

Stowe, L. (1986). "Parsing WH-constructions: evidence for on-line gap location." *Language and Cognitive Processes* **1**:**3**, 227–245.

Stowell, T. (1981) "Origins of Phrase Structure." Doctoral dissertation, MIT.

Sutton, G. (1992). "Competitive Learning and Map Formation in Artificial Neural Networks using Competitive Activation Mechanisms." Doctoral dissertation, University of Maryland, College Park.

Tanenhaus, M., L. Stowe, and G. Carlson (1985). "The interaction of lexical expectation and pragmatics in parsing filler-gap constructions." *Program of the Seventh Annual Conference of the Cognitive Science Society*, 361–365.

Taraban, R. and J. McClelland (1990). "Parsing and Comprehension: A Multiple-Constraint View." In D. Balota, G. Flores d'Arcais, & K. Rayner, Eds. *Comprehension Processes in Reading*, Hillsdale, NJ: Lawrence Erlbaum.

Vosse, T. and G. Kempen (1991). "A Hybrid Model of Human Sentence Processing: Parsing Right-Branching, Center-Embedded and Cross-Serial Dependencies." *Proceedings of the Second International Workshop on Parsing Technologies*, Cancun.

Waltz, D., and J. Pollack (1985). "Massively parallel parsing: A strongly interactive model of natural language interpretation." *Cognitive Science* **9**, 51–74.

Weinberg, A. (1991). "A Parsing Theory for the Nineties: Minimal Commitment." Manuscript, University of Maryland.

Wermter, S. and W. Lehnert (1989). "A Hybrid Symbolic/Connectionist Model for Noun Phrase Understanding." *Connection Science* **1**:**3**, 255–272.

Yu, Y. and R. Simmons (1990). "Truly Parallel Understanding of Text." *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-86)*, 996–1001.