This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e.g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.
RESOLVING
LEXICAL AMBIGUITY
IN A
DETERMINISTIC PARSER

Robert William Milne

Ph.D
University of Edinburgh
1983
List of Figures
Acknowledgements
Abstract

1. Introduction

1.1 The Human Sentence Parsing Mechanism
1.2 The Initial Framework
1.3 Summary of Results
1.4 Limitations of this Research
1.5 Outline of the Following Chapters

2. Marcus's Work

2.1 The Definition of Deterministic Parsing
2.2 Conscious Effort
2.3 How is Determinism Accomplished
2.4 Two Deterministic Parsers
2.5 What did Marcus do about Lexical Ambiguity
   2.5.1 Marcus and the Attention Shift
   2.5.2 The Three Buffers in Marcus's Parser

3. Using Determinism to Predict Garden Path Sentences

3.1 Garden Path Sentences
   3.1.1 The Garden Path Prediction of PARSIFAL
   3.1.2 Why is it Wrong
3.2 The New Theory
3.3 The First Experiment - Testing Garden Path Sentences
   3.3.1 The Pre-Test
   3.3.2 Testing the Predictions
   3.3.3 Subjects
   3.3.4 Procedure
   3.3.5 Analysis of the Data
   3.3.6 Results
   3.3.7 Discussion
3.4 Summary So Far

4. Non-Syntactic Interaction

4.1 Each Rule Makes a Semantic Contribution
4.2 Which Ambiguities need Non-Syntactic Information to Resolve Them
4.3 Which Ambiguities are Resolved with Non-Syntactic Information
   4.3.1 PP Attachment
   4.3.2 Reduced Relative Garden Paths
   4.3.3 THAT Garden Paths
4.4 The Theory of When and Why
4.4.1 When is Non-Syntactic Information Used
4.4.2 Why Must the Decision use Non-Syntactic Information
4.5 The New Garden Path Explanation
4.6 How to Recover from a Garden Path
4.7 Summary
4.8 The Second Experiment
   4.8.1 Purpose
   4.8.2 Task
   4.8.3 Subjects
   4.8.4 Examples
   4.8.5 Results
   4.8.6 Summary

5. The Role of Syntactic Information in Handling Part of Speech Ambiguity

5.1 Syntactic Context
   5.1.1 Word Data Structures
   5.1.2 Morphology
   5.1.3 Disambiguation
   5.1.4 An Example
   5.1.5 The Word "TO"
   5.1.6 Adjective/Noun and Noun/Noun Ambiguity
   5.1.7 Why Does This Work?

5.2 The Role of Agreement in Handling Ambiguity
   5.2.1 Marcus's Diagnostics
   5.2.2 Handling the Word "TO"
   5.2.3 Handling the Word "FOR"
   5.2.4 Ungrammatical Sentences
   5.2.5 Subject/Verb Agreement
   5.2.6 Plural Head Nouns
   5.2.7 Handling "WHAT" and "WHICH".
   5.2.8 Verb Agreement
   5.2.9 What About "HER"
   5.2.10 Handling "THAT"
   5.2.11 Handling the Word "HAVE"
   5.2.12 The Word "A"

5.3 Possible Uses for Agreement in English

6. The Psychological Status of Deterministic Parsing

6.1 Psychological Criteria
6.2 Kimball's Principles
6.3 The ATN as a Psychological Model
6.4 The Sausage Machine
6.5 Accounting for RA and MA in ROBIE
   6.5.1 Production System Rule Order
   6.5.2 Right Association
   6.5.3 Minimal Attachment
6.6 Some Predictions
7. Related Work

7.1 Deterministic Parsers
- 7.1.1 Resolving Noun/Verb Ambiguity
- 7.1.2 Church's YAP

7.2 The ATN

7.3 Chart Parsers

7.4 A General Syntactic Parser

7.5 Steedman and Ades

7.6 Other Approaches to Parsing

7.7 The Timing of Semantic Interaction
- 7.7.1 At the End of the Sentence
- 7.7.2 At the End of the Clause
- 7.7.3 Incremental Evaluation
- 7.7.4 The On-Line Theory

7.8 Linguistic Analysis
- 7.8.1 Extended Standard Theory
- 7.8.2 Phrase Structure Grammar

8. The Parser

8.1 Overview of the Parser

8.2 The Grammar
- 8.2.1 Grammar Functions
- 8.2.2 Closing Nodes
- 8.2.3 Attachment
- 8.2.4 Implementing the Production System Grammar
- 8.2.5 Rule Order

8.3 The Dictionaries

8.4 The Packets
- 8.4.1 Dotted Rules

8.5 A Few Notes on the Grammar
- 8.5.1 Passive
- 8.5.2 Conjunction
- 8.5.3 Movement

8.6 The Agreement Checks

8.7 The Semantic Database

9. Other Ambiguities

9.1 Verb Particle Handling

9.2 HAVE Re-visited
9.3 Global Ambiguity
   9.3.1 Preferred Reading, then Other Readings
   9.3.2 All at Once
      9.3.2.1 Similar Trees
      9.3.2.2 Similar Trees - Peudo Attachment
      9.3.2.3 Produced in Parallel
   9.3.3 Only the Right One

9.4 Why are These not Garden Paths

10. Using Semantic Information for PP Attachment and VP Parsing
   10.1 Semantic Handling of PPs
   10.2 Parsing VPs with Semantic Information
   10.3 Preference Semantics
      10.3.1 Wilks
      10.3.2 Boguraev

11. Conclusion
   11.1 Summary
   11.2 The Parser vs. The Grammar
   11.3 Areas for Future Study

Appendix A: An Annotated Example
Appendix B: Some Example Sentences
Appendix C: Free Text Analysis
Appendix D: The Annotated Grammar

Bibliography
LIST OF FIGURES

Figure 1: Examples from MECHO

Figure 2: Examples from Marcus and Church
ABSTRACT

This work is an investigation into part of the human sentence parsing mechanism (HSPM), where parsing implies syntactic and non-syntactic analysis. It is hypothesised that the HSPM consists of at least two processors. We will call the first processor the syntactic processor, and the second will be known as the non-syntactic processor. For normal sentence processing, the two processors are controlled by a 'normal component', whilst when an error occurs, they are controlled by an 'error recovery component'. These divisions are based on the observation that human beings are able to bring at least two distinct types of information to bear on a text. The resolution of lexical ambiguity will be used as a vehicle to investigate this hypothesis.

Under control of the normal component, the syntactic processor is unconscious, deterministic and fast, but limited. It is hypothesised that the syntactic and non-syntactic processors work in parallel during the processing of a normal sentence. During processing of some sentences, the syntactic processor will, at key points, ask the non-syntactic processor to make a decision in order to resolve an ambiguity. These key points occur whenever a situation arises in which the syntactic processor can no longer guarantee a correct analysis. A major focus of this research is the identification of those situations in which people use the non-syntactic processor to assist with the resolution of ambiguity and the sentences in which this occurs.

When both the syntactic and non-syntactic processors fail, for example during the processing of a garden path sentence, it is hypothesised that an 'error recovery component' is used, which controls both processors and is slower, semiconscious and non-deterministic.

We are concerned with modeling only the normal use of the syntactic processor. The major test of the psychological validity of such a model is that it fail on precisely those sentences that humans find to be garden paths. We use, as a starting point, Marcus's work on deterministic parsing. The advances reported here are:

- Reaction time experiments are used to provide a non-subjective classification of sentences as garden paths or not. Using this classification, it is shown that Marcus's parser would succeed on some garden path sentences and fail on some non-garden path sentences.
- This deficiency can be corrected by the use of non-syntactic information for ambiguities which may lead to a garden path.
- Non-syntactic information is to be used to help resolve an ambiguity when the syntactic processor can no longer guarantee a correct analysis. All other ambiguities are to be resolved on the basis of syntactic information.
- An amended parser, ROBIE is presented which incorporates these conclusions. ROBIE is shown to be compatible with the psychological evidence currently available on human sentence comprehension.
- ROBIE is computationally and conceptually simpler than Marcus's parser.
Introduction

1.1 The Human Sentence Parsing Mechanism

People are able to understand most utterances quickly and without effort. Whilst this is obvious, how they understand them is not known. It is assumed [Fodor and Frazier 1978], [Marslen-Wilson 1976], [Kimball 1975] that each person has a sentence parsing mechanism to perform this task, where ‘parsing’ implies syntactic and non-syntactic analysis. The exact nature of this mechanism is an area of active investigation.

However, researchers have agreed on several general observations about the Human Sentence Parsing Mechanism (HSPM).

a) It is very fast and efficient and rarely seems to make a mistake.
b) Humans are capable of understanding even very ambiguous sentences.
c) The listener is not aware of the majority of potential ambiguities in a sentence.
d) People are able to accept as syntactically well-formed, semantically anomalous sentences.
e) They are able to understand syntactically ill-formed but semantically meaningful sentences.
f) Some sentences, called garden path sentences, cause normal processing to fail and a semi-conscious process of error recovery takes place.

These observations suggest that the HSPM consists of at least two processors. We will call the first the syntactic processor, and the second will be known as the non-syntactic processor. In the syntactic processor, information about grammatical structure only is used. The non-syntactic processor can use information from the meaning of words, sentences and also information from intonation, discourse, and experience.

In this thesis, we will use the resolution of lexical ambiguity in written text as a vehicle to explore the nature of the HSPM and its interaction with the
non-syntactic processor. We will also attempt to present a model of the syntactic processor that is fast, deterministic and of the same power as the part of the human syntactic processor used by the normal component. To build such a model it was first necessary to explore the limitations of the HSPN; in particular, to determine when the syntactic processor might make a wrong decision and when the non-syntactic processor must be used.

It is postulated that when the syntactic processor can no longer guarantee a correct analysis, the non-syntactic processor is used. The non-syntactic processor appears to be able to bring together a large variety of information in a complex way. In this thesis, we will only investigate when this processor is called upon to make a decision and not its exact nature.

It is proposed that for normal sentence processing, these two processors are controlled by the 'normal component'. It is hypothesised that the two processors work in parallel during the processing of a sentence under control of the normal component. During processing of some sentences, the syntactic processor will, at key points, "ask" the non-syntactic processor to make a decision in order to resolve an ambiguity.

These key points would be whenever an ambiguity arose which the syntactic processor could not guarantee to resolve correctly, because of its limitations. Rather than using the non-syntactic processor only when an ambiguity has taken the syntactic processor astray, the non-syntactic processor is used when an ambiguity arises which might lead the syntactic processor astray.

A major focus of this research is the identification of those situations in which people use the non-syntactic processor to assist with the resolution of ambiguity and the sentences in which this occurs. This research is interested in the timing of this assistance, rather than its specific nature.

It appears that in the majority of cases, the non-syntactic processor will direct the syntactic processor to a correct analysis. We will however see cases in
which the non-syntactic processor leads the syntactic processor astray, where, left to itself, it might have found the correct analysis by guessing. We will see that this is a desirable feature for a psychologically plausible model of the HSPM.

When both the syntactic and non-syntactic processors fail, for example during the processing of a garden path sentence, it is hypothesised that an "error recovery component" is used. This component controls both processors and is slower, semi-conscious and non-deterministic. The nature of the error component will not be investigated in this thesis.

A model of normal sentence processing in the HSPM should fail on those sentences which people find difficult to understand and should not fail on those sentences which people have no difficulty in understanding. One type of sentences, that people have difficulty understanding, is the so-called garden path (GP) sentence. We have chosen to look at lexical ambiguity because it gives us many examples of ambiguities that can lead to a garden path.

A major test of the psychological validity of this model will be whether it fails on all garden path sentences, but does not fail on any non-garden path sentences.

1.2 The Initial Framework

As the initial framework, we will use Marcus's PARSIFAL [Marcus 1980]. Marcus presents a method of parsing that is "deterministic", i.e., that never backtracks or changes a decision. As well as being deterministic, PARSIFAL was primarily syntactic and was intended to be the first stage in a parsing system. Since Marcus's goal was the development of a deterministic parser, he did not investigate any psychological aspects. However, it is generally considered that his approach showed substantial promise as a basis for work in this area.
In this thesis, we will show that PARSIFAL would fail on some non-garden path sentences and not fail on other garden path sentences. We will explore ways of amending Marcus's parser so that it fails on all and only garden paths and also investigate part of speech ambiguity, both areas which Marcus did not consider. In addition, we want to build a computationally more simple parser which rejects the majority of ungrammatical sentences.

We will assume the linguistic analysis of Chomsky [1957,65,73,75b,76,77] and his Extended Standard Theory (EST) throughout this thesis. Marcus's parser used EST exclusively and much of his grammar has been copied into the current parser. However, although the parser uses EST, the majority of the examples in this thesis do not depend on the portion of EST which separates it from the older Standard Theory (i.e., the use of traces). As with Marcus's parser, ROBIE produces an annotated surface structure of the input sentence.

In several situations, the use of EST is incompatible with the limited parsing framework used in this thesis so we have deviated from it. This deviation is made to illustrate the implications of different grammatical theories in certain situations. These situations and places where deviations from Chomsky's analysis have been made will be noted. It is suggested that the parsing framework necessary to analyse a grammar is an effective criterion on which to judge the psychological validity of various linguistic theories. The evaluation of linguistic theories has not been a major focus of research in this thesis, although there are some very definite implications for linguistic theories. Some of these implications will be outlined where appropriate. In particular, a favourable comparison will be made between the newer Phrase Structure Grammar (PSG) of Gazdar, Pullum and Sag [Gazdar, Pullum and Sag 1980c], [Gazdar 1979,80,80b] and Chomsky's EST. PSG also helps to illustrate the interesting differences between a transformational and non-transformational approach to grammar. Although it now appears that PSG may have many benefits over EST for this deterministic parser, this potential was recognised too late to
permit a conversion of the grammar to PSG. (See Section 11.3).

In this thesis, we will discuss non-syntactic interaction rather than semantic interaction for the following reasons. In formal logic, semantic information is used to refer to that information which establishes the truth or falsity of a sentence. In other words, semantic information is all information other than syntactic information. In computational linguistics, semantic information is often used to refer to the meaning of words, the meaning of the sentence, information from the current discourse (often referred to as pragmatics), etc. In order to avoid confusion, we will refer to non-syntactic information as all types of information other than syntax. This definition will be explained in greater detail in Chapter 4.

1.3 Summary of Results

The advances reported here are:

- Reaction time experiments were used to provide a non-subjective classification of sentences into garden paths and non-garden paths. Using this classification, PARSIFAL is shown to succeed on some garden path sentences and fail on some non-garden path sentences.
- This deficiency can be corrected by the use of non-syntactic information for ambiguities which may lead to a garden path.
- Non-syntactic information is to be used to help resolve an ambiguity when the syntactic processor can no longer guarantee a correct analysis. All other ambiguities are to be resolved on the basis of syntactic information.
- An amended parser, ROBIE is presented which incorporates these conclusions. It is shown that the situations in which non-syntactic information is to be used can be accurately predicted by ROBIE's two buffer lookahead. ROBIE is shown to be compatible with the psychological evidence currently
available on human sentence comprehension.

- ROBIE is computationally and conceptually more simple than PARSIFAL [Marcus 1980].

1.4 Limitations of this Research

This research is based only on written English. It is hoped that a better specification of when other information is needed to assist processing will emerge through the discovery of the limitations of text analysis. No information from speech, such as intonation, has been considered.

As part of this research, a syntactic parser which covers a significant portion of English grammar has been written. The grammar has been designed to cover the mechanics problems of the MECHO system [Bundy et. al. 1979b]. Its coverage is illustrated in Appendix B. Nevertheless, WH movement and conjunction have not been thoroughly investigated. Marcus handled many examples of WH movement in PARSIFAL [Marcus 1980] and demonstrated that WH movement can be handled deterministically. [Church 1980] has investigated both of these areas and his work looks promising for a solution to these problems. The focus of this thesis is on the implications of lexical ambiguity on the nature of the HSPM, so WH movement and conjunction are not directly relevant to the discussion here, although a simple form of handling these has been implemented.

Because we are investigating only the syntactic processor of the HSPM, only that portion of the parser has been developed. The non-syntactic portion has been designed simply to handle the problems from the MECHO world. In order to handle problems such as conjunction, the parser maintains three buffers. The third buffer is only used by the few rules which have been identified in this thesis to be exceptions. (i.e., conjunction and the original 'have' rule.)
This research is not concerned with the exact nature of the non-syntactic tests, but rather, the timing of these tests. As a result, the situations in which a non-syntactic test should be made are identified, but the exact test is not always implemented. As will be seen, several of the tests have been implemented using semantic markers, while other tests are in the form of various heuristics. These heuristics are intended to model what a non-syntactic test should do, but they are not claimed to be theoretically significant in themselves.

Semantic problems of lexical ambiguity such as word sense ambiguity within a given part of speech and pronoun reference ambiguity have not been investigated. There are many proposed solutions to these problems, but all fall outside the scope of this research.

1.5 Outline of the Following Chapters

This thesis is divided into four main sections. In the first section, Chapter 2, "deterministic parsing" is defined and Marcus's deterministic parser, PARSIFAL, is explained. This section establishes the framework which will be used in the rest of the thesis.

The second section, Chapters 3-5, is an investigation into resolving lexical ambiguity, the first half of which concentrates on seeing how one might resolve lexical ambiguities that can lead to garden path sentences. This is done by exploring the consequences of the garden path prediction of Marcus's parser. It is seen what this prediction was and where it might be incorrect. A reaction time experiment is presented to show that this prediction was indeed incorrect.

To develop an improved prediction of garden path sentences, in Chapter 4 we explore an amended version of the parser, ROBIE, incorporating non-syntactic information. (In the form of semantic markers) This chapter also looks at the limits of the syntactic processor in an attempt to decide when non-syntactic information is
used to resolve lexical ambiguity and why. It is shown that people may use non-syntactic information to overcome the limitations of the syntactic processor.

The second half of our investigation into lexical ambiguity, Chapter 5, focuses on how to resolve lexical ambiguities that do not lead to garden paths. To provide data for this investigation, a second reaction time experiment is presented. In Section 5.1, we see which examples of lexical ambiguity can be resolved in our model by syntactic context alone. In Section 5.2, we will see what other information is needed to resolve part of speech ambiguity. It is shown that number and fixed constituent structure are sufficient to handle examples within the limitations of the syntactic processor.

The third section compares our model, as developed in section two, with other related work. Firstly, we investigate whether the model is psychologically plausible. The relevant literature is examined, in Chapter 6, and it is shown that the model can account for the data presented as well as explaining the principles of Minimal Attachment and Right Association. In Chapter 7, we look at other parsing proposals in relation to this work and we compare the proposals in this thesis with other theories regarding the timing of non-syntactic interaction.

The final section contains a description of the parser and suggestions for further work. Chapter 8 describes the workings of the parser in detail to provide the reader with a deeper understanding of the model. In Chapter 9, we return to the problems of "have" and global ambiguities and some of the linguistic implications of this work. Finally, in Chapter 10 we will see some details of how non-syntactic information is actually used in the current parser and how it could be added to an improved parser.
2.1 The Definition of Deterministic Parsing

Central to this thesis is the notion of determinism and deterministic parsing. In the following sections, what deterministic parsing is and the techniques that are used to implement it in a parser will be explained.

Marcus [Marcus 1975] first proposed that English could be parsed deterministically. He later stated:

"There is enough information in the structure of natural language in general, and in English in particular, to allow left-to-right deterministic parsing of those sentences which a native speaker can analyse without conscious effort." [Marcus 1980, p.204]

The terms "deterministic parsing" and "without conscious effort" must be explained. In relation to the former, Marcus said:

"Natural language can be parsed by a mechanism that operates "strictly deterministically" in that it does not simulate a non-deterministic machine." [Marcus 1980, p. 11]

Marcus did not propose that deterministic parsing implies that natural language could be parsed by a deterministic machine in the automata theoretic sense. He points out that any computational mechanism that physically exists is deterministic in this sense. The key point of the above statement is that the parser does not simulate a non-deterministic machine.

"Why would a natural language parser seem to need to simulate a non-deterministic machine? Kaplan suggests this answer:

'Because natural language is ambiguous, a
natural language grammar is essentially a
classification of a non-deterministic

The assertion of deterministic parsing is that a natural language grammar
can be essentially a characterisation of a deterministic machine. However, there are
two ways a grammar interpreter using a seemingly deterministic grammar can simu-
late non-determinism. These are backtracking and pseudo-parallelism.

Backtracking can be prohibited by insisting that all grammar substructures
are permanent. In a parsing context this means that, if one item is attached to
another, this attachment can never be broken. i.e., if a PP is attached to an NP, then
the parser cannot break the attachment and attach the PP to, say, the VP. If a word is
disambiguated to a certain part of speech, it can never be changed to a different part
of speech. i.e., if the parser disambiguates "block" as a noun, it cannot change it to a
verb. This prevents the grammar interpreter from pursuing a guess that turns out to
be incorrect.

It is possible to avoid backtracking, but simulate non-determinism, by tak-
ing all possible paths from a given state simultaneously. This is known as pseudo-
parallelism. This method, however, is still not permissible for a deterministic parser.
Using pseudo-parallelism, it is possible to follow each permissible transition simul-
taneously. If one of the paths fails, the parser does not return to a previous state,
but, instead, "throws away" any structure built and then terminates that path. In
deterministic parsing, building a constituent and then "throwing it away" is not per-
mitted. This technique is therefore also disallowed.

We have two points relating to a deterministic parser. It must neither
backtrack nor use pseudo-parallelism. In deterministic parsing, should a transition
be made from some state, we are guaranteed that the subsequent state will be on the
path to a successful parse, if such a path exists. We shall consider this to be the
definition of deterministic parsing.
Most current parsers are clearly non-deterministic. The ATN parser of Woods [Woods 1970] makes extensive use of backtracking. In a "Chart" parser, [Kay 1973] no backtracking occurs, but pseudo-parallelism causing wasted structure is very common. The Chart parser creates all possible connections (edges) between parse nodes. Of all these possible edges, only some are used and the unused nodes violate the definition of determinism. Definite Clause Grammars as proposed by Pereira and Warren [Pereira 1980, Pereira and Warren 1980], are generally parsed using the backtracking of PROLOG and, hence, are also not deterministic.

2.2 Conscious Effort

This thesis makes some claims about the Human Sentence Parsing Mechanism (HSPM). However, it does not claim that the entire HSPM is deterministic. For the reasons explained in the introduction, I feel that there is a "syntactic processor" for normal sentence processing which is deterministic. Too little is known about the rest of the HSPM to make any statement about it; nor do I claim that all sentences are parsed without conscious effort. I do claim that those sentences which can be parsed by people with no conscious effort, can be parsed deterministically by the "syntactic processor", even though the non-syntactic processor may assist. As we are only concerned with the function of the syntactic processor for normal sentence processing no claim is made as to whether the non-syntactic processor is deterministic.

It is assumed that the syntactic and non-syntactic processors of the HSPM rarely fail. When they do, conscious effort is used to recover from the error and the error component takes control. It may use the syntactic and non-syntactic processors non-deterministically. If we assume that every failure of these processors causes conscious effort, we know exactly when a failure occurs. If the model is unable to proceed at the point where people exert conscious effort, then it will be failing at the same point as the syntactic and non-syntactic processors of the normal com-
ponent of the HSPM.

What does "no conscious effort" mean and how do we decide which sentences fall into this category? Unfortunately, the answer to this question is not simple.

It is assumed that the processing of a normal sentence does not require conscious effort and it is generally agreed that to understand a garden path sentence requires conscious effort. The reader notices a mental "jump" or "block" when reading of the sentence stops and the garden path is consciously realised. Experimentally, conscious effort can be detected by an increase in reaction time to a given task. As an armchair definition; any grammatical sentence that seems abnormal to read, requires conscious effort.

Since the line between conscious effort and no conscious effort is unclear, we will concentrate on those examples that clearly require conscious effort. Without a clear definition and understanding of conscious effort, it is impossible to evaluate deterministic parsing for all sentences in a language. More experimental data must be collected in many areas before we can conclude what does and what does not require conscious effort. Throughout this thesis, I will point out when we are assuming that a sentence or fragment requires conscious effort, when we know that it does and when the deterministic parser predicts that it does.

2.3 How is Determinism Accomplished?

The simple technique that makes determinism possible is limited lookahead.

"Lookahead" means looking ahead in the input stream before deciding which grammar rule to execute and hence, which will be the next state. This lookahead is always to the next K constituents after the item the parser is currently constructing. (Where K may vary from parser to parser). The current grammar is written such that each rule can examine the features of two "buffers" containing consti-
tuents, i.e., $K=2$. (see Section 2.5 for a full explanation of this.)

If it were possible to look arbitrarily far ahead, then it would be possible to have a buffer cell containing each word in the sentence. A single grammar rule which constructed the correct parse tree could then be written for each possible sentence. Whilst a parser that did this would be deterministic, it would not be psychologically plausible. The reasons for this will be explained in Chapter 4 and in Section 7.3. Unrestricted lookahead would also enable the grammar interpreter to simulate a non-deterministic machine by allowing it to perform 'closet backtracking'.

Limited lookahead is supported by "wait and see". If it is unclear how a certain word or constituent should be used during the parse, the parser "waits to see" what should be done. Because the parser is not allowed to ignore or throw away any structures which it has built, this should not be construed as 'closet backtracking'. As [Marcus 1980, p. 24] points out, it is not possible for the parser to simulate what it might do with the input (and hence simulate backtracking) because it cannot build and then discard these structures.

"Wait and see" means that, if the parser is unsure of a situation, it does not make a random guess. Instead it waits until it has enough information to make the decision correctly. Rather than making an arbitrary decision or pursuing several options in parallel, it suspends working on that constituent until it has sufficient information to make the decision correctly. Marslen-Wilson's comment on "securely" indicates that there may be a psychological basis to this approach:

"The general claim I want to make is that the human speech understanding system is organised in such a way that it can assign an analysis to the speech input at the theoretically earliest point at which the type of analysis in question can be securely assigned. What is meant by the term "securely" here is that the system does not, within limits, make guesses about the correct analysis of the input" [Marslen-Wilson 1980b, p.16]
My claim is that this applies to the syntactic processor as well as the speech recognition system. Not making guesses is central to the theory of deterministic parsing. By using limited lookahead and wait and see, it is possible to analyse many sentence forms correctly, without the need for backtracking.

There is one more important strategy in implementing the deterministic parser. A non-deterministic parser is typically driven top-down. It often tries to start a new constituent before it has seen whether any of the following input words can start that constituent. Whenever the parser discovers that there are no lexical items for that constituent, it must backtrack and attempt to find an alternative constituent. Creating a new node before one sees if there are any lexical daughters for it can cause much backtracking. Creating a PP node at the end of a NP, before the parser has checked to see if the next word is a preposition, is an example of this guessing. By using lookahead this is not necessary. It is possible to check that the appropriate lexical items are present before a constituent is initiated. ROBIE does not begin the construction of a new constituent unless it has a lexical daughter for that constituent. We shall return to this later in Section 6.5. One can see that creating new constituents without regard for the following items is the same as the guessing to which Marslen-Wilson alluded above.

One main difference in principle between a non-deterministic parser and a deterministic parser occurs when decisions are made during the parse. In a non-deterministic parser, a path is usually followed and the structure is built first. After this, checks are made to see that both path and structure were correct, if not, the parser can backtrack and try again. In a deterministic parser, one asks: 'Does it make sense to build the item?' before it is built. If it will make sense, then it is built, otherwise it is not. This is one of the key processing principles that distinguishes deterministic parsing from non-deterministic parsing.
2.4 Two Deterministic Parsers

Given this definition of deterministic parsing, we will now turn to Marcus's parser, PARSIFAL, and see how this definition was used by him in parsing. In this section, PARSIFAL, its motivation and structure will be described and contrasted to the Milne parser, ROBIE. The reasons for the modifications of PARSIFAL incorporated in ROBIE will be explained in the subsequent chapters.

Marcus first became interested in deterministic parsing when he watched an ATN parser needlessly backtrack, making the same error over and over again, in a situation where the correct solution was obvious to him. (personal communication) He then designed a deterministic parser. To be deterministic, it seemed that the parser would need at least the following three properties:

1) It must be, at least partially, data-driven.
2) It must reflect expectations.
3) It must have some sort of limited lookahead.

The Marcus parser, PARSIFAL, has two main data structures and a grammar interpreter. The first of these is a system of buffers. These are a number of cells containing words or items that have been constructed, but whose grammatical role is unknown. Whilst the buffers can contain any constituent under a single node, for reasons that will be explained below, when a NP is being parsed, the buffers only contain words.

The other is a 'push-up' stack which contains incomplete constituents. This is called the Active Node Stack. The Active Node Stack is constrained such that the constituents it contains must be dominated by a non-terminal node, i.e., a partially built NP, VP, PP, S, etc. If it could contain a terminal (word) that was not dominated by a non-terminal node, then it could be used as an extension of the buffers. This would provide arbitrarily long lookahead, which, as previously explained, is forbidden. PARSIFAL placed words on the Active Node Stack in order to implement the Attention Shift. In ROBIE, all node movements are performed explicitly by the gram-
mar rules and these rules are forbidden from placing words on the Active Node Stack.

The Active Node Stack in ROBIE is identical to that used in PARSIFAL with this one exception.

Two Main Data Structures:
1) The Active Node Stack
   - contains incomplete constituents
2) The Buffers
   - contain words or constituents (complete or incomplete)

Marcus allowed from three to five buffer cells in his parser and these provided the lookahead capacity. For reasons which will be explained in Section 2.5, ROBIE uses two static buffers. (It actually maintains three to handle conjunction, but the remaining grammar rules can only use the first two buffers.) These two buffers are always kept filled and this fact constitutes one of the major differences between these two parsers.

The parsers move from left to right over the input string, building structure as they proceed. They may suspend construction of a constituent by pushing another item up onto the Active Node Stack. The buffers are always the rightmost nodes under consideration by the parsers. They can be considered to be "below" the Active Node Stack, with possibly completed constituents being "dropped" from the Active Node Stack into the buffers.

In ROBIE, these buffers are always the next two cells after the bottom of the Active Node Stack. The buffers move right and left as items are pushed up onto and popped down off the stack. The grammar and the rule matcher can only look at the two buffers. (The one exception is conjunction. No one has yet researched a satisfactory conjunction method for a deterministic parser. There appears to be no solution that can handle a wide range of conjunctions without a special mechanism.)

Marcus's parser was written in LISP, whilst ROBIE is written in PROLOG [Pereira, Pereira, and Warren '1978] and is the natural language front-end to the MECHO project [Bundy et. al. '1979b]. Both parsers take, as their input, an English sentence. They then produce a syntax tree of that sentence as output. For example,
the output from ROBIE for the sentence:

[1] The shy boy has kissed Mary.

will be:

S-1 [s,major,decl]
   NP-1 [np,def,ns,n3p]
       DET THE [det,def,ns,n3p]
       ADJ SHY [adj]
       NOUN BOY [noun,ns,n3p]
   AUX-1 [aux,past,v3s]
       AUXVERB HAS [auxverb,past,v3s,verb]
   VP-1
       VERB KISSED [verb,en,past,v3pl]
   NP-2 [np,name,ns,n3p]
       NAME MARY [name,propnoun,ns,n3p]

This tree is very similar to the output of PARSIFAL.

In parsing the sentence "The shy boy has kissed Mary.", the state of ROBIE, which is very similar to the state of PARSIFAL, would be as follows:

Packet: CPOOL
Rule about to run: PROPNAME  pattern: [name]

Active Node Stack:
2: <open> S  NP det-the  [SS-FINAL,CPOOL]
       adj-shy
       noun-boy
       AUX auxverb-has
1: <open> VP  verb-kissed  [SS-VP,CPOOL]

B1: Mary
B2: 

Words are first considered by ROBIE, as they arrive into the second buffer. For PARSIFAL words arrive into whichever buffer is the rightmost. In both parsers, the grammar may indicate the start of a new constituent, based on the syntactic features of the next word. This constituent will be placed on the bottom of the Active Node Stack. The Active Node Stack grows upward and only the bottom item of the stack is currently Active, i.e., the only item being actively constructed at the
time. In this example, the VP node is the **Current Active Node** and work on the S node has been suspended until the VP node is finished.

In the diagram, the first buffer contains the word "Mary" and the second buffer contains the word ".".

In both parsers, the grammar consists of a set of production system rules ordered into packets. Each rule consists of a pattern for the head, that serves for the production system pattern, and a body that is executed by the interpreter once the rule has been selected. Every rule has a name, a priority and is a member of a packet. For example, here is the rule from ROBIE for parsing a determiner. Programming details are left out for convenience. (PARSIFAL's rule did not have the semantic interpreter step.)

**Rule DETERMINER in packet PARSE-DET:**
To analyse a determiner, if you have the feature "det"
in the first buffer then:-
1) attach the first Buffer to the bottom of the Active Node Stack as a determiner.
2) tell the semantic interpreter you have a determiner.
3) deactivate the packet containing this rule, PARSE-DET
4) activate the packet PARSE-QP-2
Recursively call the rule matcher.

In both parsers, only rules in active packets can be tested by the interpreter. The parsing mechanism puts no constraint on the number of packets that can be active at any one time. In practice there are rarely more than three packets active. A rule body can activate and deactivate packets. This provides the top-down or "reflect expectation" component of PARSIFAL and ROBIE. The other functions of a rule body in ROBIE are explained in Chapter 8. When a pattern has been matched and a rule body is about to be run, the packet containing that rule is known as the 'Current Active Packet'.

In PARSIFAL, the patterns of the grammar rules could match any combination of the contents of the three buffers and the bottom of the Active Node Stack. (The item currently being built). PARSIFAL could also check features on the lowest S node of the stack. This means that each pattern could inspect up to five nodes before
matching. In ROBIE, the patterns are constrained to match only two buffers and it is not possible to access any node in the Active Node Stack except the bottom one. The reasons for this will be described in the next section.

The grammar rules in ROBIE are very similar to Marcus's original rules. In fact, most of the rules have the same names. They have, however, been modified slightly, as will become apparent later.

The production system grammar rules were structured by combining groups of rules into packets, a packet being a collection of rules. These packets can be made active or inactive by the parser. Only active rules can match against the state of the parser. Each node on the Active Node Stack has a list of active packets associated with it. In the above diagram, the packets SS-VP and CPOOL are associated with the VP node, and the packets CPOOL and SS-FINAL are associated with the S node. The only packets which are active are those associated with the Current Active Node.

The following diagram illustrates the structure of the grammar. This is similar to [Marcus 1980, p. 19].

The grammar:

<table>
<thead>
<tr>
<th>Matched against the Buffers</th>
<th>Priority</th>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKET1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>[ ] [ ]</td>
<td></td>
<td>ACTION1</td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION2</td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION3</td>
</tr>
<tr>
<td>PACKET2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION4</td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION5</td>
</tr>
<tr>
<td>15:</td>
<td>[ ]</td>
<td></td>
<td>ACTION6</td>
</tr>
<tr>
<td>PACKET3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:</td>
<td>[ ] [ ]</td>
<td></td>
<td>ACTION7</td>
</tr>
<tr>
<td>5:</td>
<td>[ ] [ ]</td>
<td></td>
<td>ACTION8</td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION9</td>
</tr>
<tr>
<td>10:</td>
<td>[ ]</td>
<td></td>
<td>ACTION10</td>
</tr>
</tbody>
</table>

THE BUFFERS: [1st][2nd]
We can now look at the three desirable properties mentioned above and see how both parsers incorporate them.

1) The parsers are data-driven in that the patterns of the rules will match the words as they arrive in the buffers, or other items in the buffers.

2) Both parsers reflect expectations, since only rules in active packets can match. By activating the packets for the expectations we have, the parser provides a top-down component.

3) The buffers give a constrained lookahead such that the contents of a limited number of buffers may be examined before a rule is matched.

More on ROBIE and how it differs from PARSIFAL will be found in Chapter 8.

2.5 What did Marcus do about Lexical Ambiguity?

In the previous section we saw the structure of Marcus's parser. In the next sections Marcus's method of handling lexical ambiguity [Marcus 1980] and several problems with his approach will be explained.

In Marcus's parser, almost all words were defined as only one part of speech. For example, "block" was defined only as a noun, whilst "schedule" was defined only as a verb. This is clearly an over-simplification of the English language. As a result, the following sentences could not be parsed by his parser: (example sentences will marked with a number in square brackets. (e.g. [4])).

[2] I lost my schedule
[3] The car will block the road.

It is easier to parse sentences if one has to deal only with structural ambiguity and not part of speech ambiguity as well. Whilst Marcus's thesis clearly demonstrated that it is possible to parse a wide range of syntactic phenomena, he showed this on a relatively simple, non-ambiguous set of sentences.
One might then think that his work is not really a proof of how easy it is
to parse deterministically, since there was very little ambiguity to confuse the
parser. If words had only one part of speech, then part of speech ambiguities could
not occur and the following sentences would either be readily comprehensible or
total nonsense depending on which part of speech the ambiguous words (i.e., 'block',
'will', "can" and 'her') were given.

[4] The block will be made of wood.
[5] The plug will block the pipe.
[6] He wrote the will.
[7] The trash can is red.
[8] Mary patted her dog.

If deterministic parsing is to be able to handle a wide range of sentences,
then, clearly it must resolve part of speech ambiguity. A non-deterministic parser
solves this problem by trying one part of speech, and if it is wrong, backtracking, to
try another part of speech. A deterministic parser, on the other hand, is not allowed
to experiment, but must find the correct part of speech at first try. As the work by
[Milne 1978] showed, handling noun/verb ambiguity in Marcus's parser was possi-
ble, but other types of part of speech ambiguity were not investigated. Nor did
Marcus investigate this problem.

We are interested in a psychologically plausible method of parsing. It seems
that people have no trouble with most occurrences of part of speech ambiguity, so it
should be stipulated that it must be easy for the parser as well. In general people do
not notice all occurrences of lexical ambiguities. If handling part of speech ambi-
guity in the parser requires much extra effort and machinery, then this would be
considered a serious blow to deterministic parsing as a psychological model. Con-
versely, the ability to handle part of speech ambiguity quite easily would support
deterministic parsing as a viable approach to natural language parsing. It would be
important if the constraints of the parser influenced the way we handle ambiguity
and even more important if the parser design enabled us to predict successfully the
effect of ambiguous situations on people.
As has been just explained, Marcus did not investigate part of speech ambiguity. There are two other problems arising from Marcus's research. These are the "Attention Shift" and his use of "three buffers". We will look at each of these in turn in the next two sections.

2.5.1 Marcus and the Attention Shift

When Marcus [Marcus 1980] presented his parser, he assumed that NPs arrived in the buffers in a method which was transparent to the parser. Throughout his work, it is assumed that NPs appear fully parsed. With this assumption, the parser uses three buffer lookahead to do all the rule matching.

In his chapter "Parsing Noun Phrases", [Marcus 1980, Chapter 8], Marcus describes how NPs are parsed. He extends the parser by adding a new class of grammar rules called "Attention Shifting" rules. These rules work, as the name implies, by shifting the attention of the parser. Normally the parser has its attention on the first buffer, but an Attention Shift can cause the parser to "shift its attention" to any of the three buffers. The buffer on which attention is focused is the new first buffer. This Attention Shift (AS) can occur whenever a word which may start a noun phrase arrives into one of the buffers. For example, if a determiner arrives into the third buffer, then the parser will "Attention Shift" to the third buffer. This will make it the virtual first buffer and have two more buffers available. The parser will then build the NP as if the preceding buffers were not there. When the NP is finished, the parser returns to the original first buffer, leaving the newly built NP in the third buffer. When the Attention Shift is used, PARSIFAL is using 5 buffers.

This method provides the mechanism for NPs arriving in the buffers in a way which is transparent to the rest of the parse. It seems that some similar method is needed if the parser is to have rule patterns of the form: 
Each of these has an NP in the second buffer. If the parser cannot shift past the first word, it is very difficult to build this NP. For example, if the pattern to start an NP was:

[start\[np\]
[to][np]
[for][np][to]

and the rule could match only one of the buffers, then all three of the following patterns would be needed:

[start\[np\]
[] [start\[np\]
[] [] [start\[np\]

To avoid the need for all three patterns, the Attention Shift rule could match its pattern to any of the three buffers.

The Attention Shift is a very powerful tool, but it has a few undesirable side effects. First, to handle the parsing of NPs, an extra mechanism was added. This seems to be necessary because of the above patterns. Secondly, there seems to be no principle governing the situation in which the Attention Shift should be used.

How many buffers are now needed? Marcus introduced his parser with three buffers. If the parser is shifted to the third buffer, does it still have the three buffer lookahead? The answer to this question is 'yes'. When the parser is in an Attention Shift, the old third buffer, in the worst case, is now the first buffer and there are two more buffers after it. This means that at the worst, five buffers are needed, the three buffer lookahead, and the two buffers that have been Attention Shifted past, not three as before.

What happens if we are in an Attention Shift and another determiner arrives in the third buffer? If we could re-attention shift then the number of buffers would be five plus two, i.e., seven. Clearly this is against our goal of a limited lookahead.
Marcus prevented this by allowing only two Attention Shifts at any one time. The second attention shift was restricted to parsing numbers and other semantic items inside the NP. For example; in the fragment, 'a two hundred pound rock', the words 'two hundred pound' would be assembled by a second Attention Shift.

The Attention Shift is undesirable for several processing reasons. Firstly, it increases the number of buffer cells. If the parser was using three cells with an Attention Shift it could be increased to five cells. This made the claim of limited lookahead weaker. Secondly, it adds a special mechanism to the parser. Finally, note that it is possible to Attention Shift past individual words. We have said that we do not allow the Active Node Stack to contain individual words unless they are dominated by a non-terminal. This was necessary to prevent the Active Node Stack from being an extension of the buffers, and providing an unlimited number of lookahead buffers. By Attention Shifting past individual words, we are violating the spirit of this principle. It seems that it would be desirable to get rid of the Attention Shift.

2.5.2 The Three Buffers in Marcus's Parser

Many people have raised objections to the use of three buffers. Marcus gave only empirical rather than experimental evidence of the need for three buffers [Marcus 1980]. He also varied the number up to five in some cases and proposed that the number may actually vary from person to person. If three buffers are needed for English, the question also arises, are three buffers needed for all languages? Do some languages need four, and others require only two?

As we will see, looking at the next word only (one buffer) is not sufficient to prevent backtracking. If it was sufficient, the non-determinism of most current parsers would have been eliminated. We will see examples in Chapter 5 where one buffer lookahead is not sufficient. Therefore, two is the minimum number of buffers to prevent backtracking. We shall also see that no matter how
many buffers we have, there will still not be enough information to resolve some ambiguities.

The two primary motivations for three buffers were to handle some of the diagnostics and embedded sentence parsing. We will look at the diagnostics in Chapter 5 and show how to re-formulate them without three buffers. The three buffers were needed to start embedded sentences such as: [Marcus 1980]

<table>
<thead>
<tr>
<th>Marcus's Pattern:</th>
<th>Marcus's rule name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[for][np][to]</td>
<td>INF-S-START</td>
</tr>
<tr>
<td>[np][to][tenseless]</td>
<td>INF-S-START1</td>
</tr>
<tr>
<td>[that][np][verb]</td>
<td>THAT-S-START</td>
</tr>
<tr>
<td>[verb][np][verb]</td>
<td>SUBJ-QUEST</td>
</tr>
<tr>
<td>[have][np][verb]</td>
<td>HAVE-DIAG</td>
</tr>
</tbody>
</table>

Rules of the form X-S-START, started embedded sentences with a complementiser of the type X. These are Marcus's only rules (excluding those used to parse time and number phrases) which used the three buffers in their patterns. In order to get the subject of the embedded sentence in the proper place, Marcus's parser waited until the above patterns were filled before it started the embedded sentence. To avoid problems with ambiguity and get the correct semantic analysis, this seems necessary. In fact, if one uses Chomsky's [Chomsky 1973,75,76,77] analysis, three buffers are needed, but if a different linguistic analysis is used, such as Gazdar's [Gazdar 1979,80a,80b,80c] Phrase Structure Grammar, only two buffers seem to be needed for parsing VPs. The Chomsky analysis is used with MECHO to remain compatible with its semantics.

The embedded sentence can be started before all its daughters are present, provided there is no ambiguity about its beginning. It will be seen in the following chapters how each of these can be eliminated and re-formulated with only two buffers. (The rule SUBJ-QUEST above will not be discussed, since it is not relevant to the current grammar. It is not anticipated that this rule will cause further problems.)
This claim, that only two buffers are needed, can be used to investigate linguistic theories. Some theories (Chomsky) seem to need the three buffers to start an embedded sentence. In others, (Gazdar) it seems possible to use only two buffers. For example, consider the INF-S-START1 rule above.

In Chomsky's analysis, the NP is attached as the subject of the S-. In order to be certain that "to" is part of the VP, it is necessary to check that it is followed by a tenseless verb. If this is true, then the S- can be started and the NP attached as the subject of the new S. It would be wrong to attach the NP to the VP of the current S, as this would produce the wrong semantic meaning. Therefore, the three buffers seem to be necessary.

In Gazdar's analysis, the NP will be attached to the upper VP, not as the subject of the S-. In fact, he does not have an S-, only a VP. Whether "to" starts a PP or a VP, the NP is still attached in the same place. Hence, we can attach the NP first and then a later rule can decide what to do with 'to'. This approach can be implemented with two buffers.

The resulting structures are:

Chomsky: (NP V (NP to V NP))
Gazdar: (NP V:NP (to V NP))

Either of these could be correct, but the latter requires only two buffers. In ROBIE we use only two buffers for the pattern matching. It is intuitively obvious that deterministic parsing with one buffer is not possible. We will see several examples of ambiguity in the next two chapters that demonstrate this. I will demonstrate that it is possible to resolve ambiguities with two buffers. This number is proposed to be invariant throughout the parse and possibly across languages.

It should be noted that to handle the phenomenon of conjunction three buffer lookahead is still needed, in order to jump over the "and". As I have not investigated conjunction, all that is said here excludes sentences using conjunctions.
It may be noticed that almost all of the rules with three buffers need the Attention Shift as well. Removal of the three buffers and the AS go together. The rules which needed three buffers were either the "diagnostics", which are the subject of Chapter 5, or the VP rules, which I reformulated above. Once they were changed, there was no need for three buffers and no need for the AS, except to build prepositional phrases.

To build a prepositional phrase in Marcus's parser, the pattern

\[\text{[prep]}[\text{np}] \rightarrow \text{PP}\]

was used. This pattern required that the Attention Shift build the NP before this rule could run. This approach built PPs bottom-up, ignoring Kimball's [Kimball 1973] principle of "function words start new nodes". Kimball's principle stated that the preposition initiate the construction of a PP. But Marcus's approach did not build the PP node until the NP had been built.

To rectify this, we can build PPs top-down by using the pattern

\[\text{[prep]}[\text{ngstart}]\]

to start the PP. This follows Kimball's principle and guarantees that the node with the feature "ngstart" will turn into a NP. The parser can start the PP, attach the preposition and then build the NP. When the NP is finished it can be attached to the PP. This has the same effect as Marcus's rule, but does not require the Attention Shift.

The PP phrase is then on the Active Node Stack (ANS) whilst the NP is being built. As a result there are now more items on the ANS, but fewer buffer cells. It is clear that the NP will be attached to the PP when finished, so this could be done at the time the NP is started and hence reduce the number of ANS cells. The current parser is not concerned with limitations on the size of the Active Node Stack, so this is not an issue. The total number of cells in the parser is the same in both systems. It is basically as efficient for the parser to have separate unlinked cells, as it is to have lots of cells, so the memory problem is not really relevant here.
Thus, we have removed the need for the Attention Shift and the third buffer. From now on, we will use neither the third buffer for pattern matching, nor the Attention Shift.

There is one problem Marcus mentioned that we will not look at in this thesis. This is ambiguities involving the word "as". Some of the problems which can arise are illustrated in the following examples:

[9] As many as ten different explanations were proffered
[10] No one could ever be as big as big bad John.
[11] He left as quickly as he could
[12] Who could believe he caught as big a fish as that?
[13] Bill offered his advice as an expert in such matters.

[Marcus 1980] used these examples to suggest that perhaps a three buffer lookahead and the Attention Shift are necessary to diagnose the proper use of the leading "as" in each example. The following buffer patterns indicate the type of constituent which should be built, based upon the three buffers. The pattern [np] has been changed to [ngstart], as this is all that is necessary to start an NP.

\[
\begin{align*}
[as][quant][as] & \rightarrow \text{quant} \\
[as][adj][as] & \rightarrow \text{adj} \\
[as][adv][as] & \rightarrow \text{adv} \\
[as][adj][a] & \rightarrow \text{np} \\
[as][ngstart] & \rightarrow \text{pp}
\end{align*}
\]

These above patterns demonstrate that the Attention Shift is not actually needed in order to diagnose the different uses of 'as', but only three buffers. Further more, based on the above data, there is only one ambiguous case. This is exemplified by [10] and [12], when an adjective occurs in the second buffer. But in these examples, the verb subcategorisation provides enough restriction to remove the ambiguity. That is, one cannot say:

[14] *Who could believe he caught as big as big bad John.

It should be noted that Marcus did not implement any grammar rules to handle the word "as". "As" can play many grammatical roles and there is more than one theory of its possible roles. For a discussion of these difficulties, see [Bresnan 1973]. I feel that the current linguistic and psychological evidence is not
sufficiently compelling to reject the use of two buffers and no Attention Shift on the basis of these "as" examples alone.

PARSIFAL could handle a wide range of syntactic constructions as illustrated in Appendix B. Marcus also demonstrated that many linguistic "Universal" constraints could be accounted for by the parser structure. In this section, we have seen that Marcus did not handle part of speech ambiguity. Other limitations of PARSIFAL will be discussed in subsequent chapters. Firstly, though, we will consider its use in the prediction of garden path sentences.
Using Determinism to Predict Garden Path Sentences

We will begin our investigation into resolving lexical ambiguity by looking at those examples which can lead to a garden path.

A model of normal sentence processing for the HSPM should fail on those sentences which people find difficult to understand and should not fail on those sentences which people have no difficulty understanding. One type of sentences, which people find difficult to understand, are the so-called garden path (GP) sentences. Lexical ambiguity gives us many examples of ambiguities leading to a garden path. In this chapter, we will look at words which can be either a plural noun or a singular verb. We will investigate the decision as to whether the word is being used as a plural noun or a singular verb and how this can lead to garden path sentences. In the next chapter we will look at other examples of ambiguity which can lead to a garden path.

3.1 Garden Path Sentences

A garden path sentence is one which seems to lead people "down the garden path", i.e., a person seems to analyse incorrectly a portion of the sentence and then, because of later evidence, must go back, reanalyse and correct the mis-analysis. In this section we will explore several definitions of a "garden path". Until we redefine a garden path later in this thesis, we will use Marcus's definition.

Marcus [Marcus 1980, p. 202] says garden path sentences are those:

"which have perfectly acceptable syntactic structures, yet which many readers initially attempt to analyse as some other sort of construction, i.e., sentences which lead the reader 'down the garden path'.

38
The following is a classic garden path:

\{15\} The horse raced past the barn fell.

In each sentence of this type, there is a point where two possible analyses are possible, i.e., at "raced". The need to backtrack is a result of selecting an analysis differing from that demanded by the rest of the sentence. For each garden path sentence there is a corresponding sentence which does not require backtracking, e.g.

\[16\] The horse raced past the barn.

This non-garden path partner has the same two possible readings at the same point, but the analysis selected is that demanded by the rest of the sentence. Such a pair of sentences will be called a pair of potential garden path sentences. Whenever a person encounters a particular potential garden path sentence, that sentence may, or may not, cause a garden path. In that situation, of the pair of potential garden path sentences, one is a garden path and the other is not, although which is the garden path is variable over subjects and from situation to situation. Examples of potential garden path sentences will be marked with curly brackets (e.g. \{1\}).

Why do these sentences cause problems for people? The proposed answers to this question are in the form of other definitions for a garden path. Crain and Coker [Crain and Coker 1979] note: "Bever as well as Chomsky and Lasnik have argued convincingly that unacceptability of GPs is due to processing difficulty." Chomsky and Lasnik say "garden path sentences result from the omission of all syntactic markers which signal that one is parsing a Complex NP". This explanation suggests that all garden path sentences should be a problem of un-marked relative clauses.

For example, in \{15\}, the relative clause marker has been omitted. Other explanations for the difficulty come from Fodor, Bever and Garret.

Bever says "the first N..V..(N) clause.. is the main clause, unless the verb is marked sub-ordinate" [Bever 1970, Strategy B, p.294]. Fodor, Bever and Garret [Fodor, Bever and Garret 1974, p. 356] have the Canonical Sentoid Strategy to
account for the unacceptability of GPs. This is a structure independent mapping from the surface syntactic structure to the semantics. This strategy always "takes the verb which immediately follows the initial NP of a sentence as the main verb, unless there is a surface structure mark of an embedding".

These explanations account for the difficulty in [15], but, again, suggest that all garden paths are due to the difficulty of an un-marked relative clause.

Woods' presents the following definition of a garden path:

"In human parsing, there are clearly cases where, on the basis of local context and the history of the sentence up to a point, a decision is made to follow a particular alternative and all other alternatives are left to be processed later. This type of processing gives rise to the so-called "garden path" sentence in which the listener is fooled into a false choice among syntactic alternatives and must consciously undo this choice after detecting an inconsistency." [Woods 1973, p. 133]

This definition, like Marcus's, is more general and allows for garden path sentences which do not involve relative clause conflicts. We will use Marcus's definition, until a new definition is presented later in this thesis. Whilst these definitions seem to account for the examples presented here, do they truly account for all garden path sentences?

In order to answer this question, let us look at a simple case of ambiguity and how it can be resolved with a two buffer lookahead. We will then turn to a more difficult example. The sentence fragment:

[17] The toy rocks....

could be completed as:

[18] The toy rocks are red.

In sentence [18] the subject NP is "the toy rocks" and "rocks" is a noun, while in [19] the subject NP is only "the toy" and "rocks" is a verb. In order to find the end of the noun phrase correctly, the parser must detect these possibilities and
decide which is applicable. A non-deterministic parser with a backtracking capability, could always try one analysis first, and if this fails, try the other analysis by backtracking. The only difficulty is which alternative to try first as a matter of efficiency.

A deterministic parser however, is not able to backtrack and hence cannot follow this strategy. Once the deterministic parser 'decides' that a word is a noun, it is committed and cannot change its mind. Hence, the deterministic parser must be able to decide what part of speech the ambiguous word is without making an error.

ROBIE uses two buffer lookahead to see the following word and decides which part of speech the word is being used as. For example, when parsing [18], the parser would have the word "rocks" in the first buffer and the word "are" in the second buffer, i.e:

[rocks][are]

This pattern indicates that "rocks" was being used as a noun in this sentence. For [19], the buffers would be:

[rocks][easily]

showing the verb usage. By looking at the following word, the parser can handle these examples without the necessity to backtrack.

Using two buffers then, it is often easy to decide which part of speech a word is being used as without needing to backtrack. This is a much cleaner and simpler result than the non-deterministic approach explained above.

As was explained in Section 2.4, the Active Node Stack cannot contain words unless they are dominated by a non-terminal node, i.e., a partially built constituent. Because of this, the lookahead buffers will always contain words when the headnoun of a NP is being constructed. The detailed reasons for why this must be true are not relevant to our discussion here, but will become clear later in the text.

Let us now look at a more difficult example. In this case, the two buffer
lookahead is not sufficient to resolve the ambiguity.

\[ \text{[20]} \text{ The building blocks the sun.} \\
\text{[21]} \text{ The building blocks the sun faded are red.} \\
\text{[22]} \text{ The building blocks the sun shining on the house.} \\
\text{[23]} \text{ The building blocks the sun shining on the house faded are red.} \]

In these examples, the second buffer is not sufficient to disambiguate the ambiguous word properly. To distinguish between [20] and [21], it seems we would need four buffers and to distinguish between [22] and [23] we would need eight buffers since we need to see the word "faded" before we could resolve the ambiguity. If the word "blocks" is currently in the first buffer, we need six buffers containing "the sun shining on the house", plus a buffer containing "faded". This is eight buffers in total.

In fact, there can be any arbitrary number of words between "blocks" and the word which indicates whether "blocks" is a noun or a verb, see [Milne 1978]. Therefore, no fixed amount of lookahead will be able to disambiguate the word "blocks" in this situation. In no way, using only syntactic information, could a two buffer deterministic parser handle these examples. As was explained in the previous chapter, the use of an arbitrary number of buffers would make our claim of psychological plausibility vacuous.

The above sentences are in two pairs of potential garden paths. Within each pair, the sentences are the same for most of the string, but differ at the point at which the function of the word in question (blocks) can be ascertained (at faded). For the parser this means that the buffers will contain the same items and the disambiguating word will be beyond the buffers (to the right). [20,21] and [22,23] form two such pairs. For all of these, the buffers contain [blocks][the] and the disambiguating word is too far to the right.

[Milne 1978] explored the handling of noun/verb ambiguity in a deterministic parser and showed that the "building blocks" sentences could lead to garden paths and hence each one is a potential garden path. In fact this paper showed that
all situations involving a word that could be either a noun or a verb followed by a word that could be a plural noun or a verb, could lead to a garden path. The above definitions do not explain why these could be garden path sentences. [Milne 1978] made no attempt to handle these situations.

Consider the following sentences:

{24} The toy rocks near the child quietly.
{25} The toy rocks near the child are pink.

These are a pair of potential garden path sentences. They help to demonstrate that, in the situation of a singular noun/verb word followed by a plural noun/verb word, it is always possible to finish the sentence so that it will be a garden path. We will refer to this type of a garden path as a plural garden path.

As previously stated, the lookahead may need to be arbitrarily long to handle these examples. In these cases, the deterministic parser cannot disambiguate all of them properly.

At this point, we must return to the motivation for the parser. It was constructed to parse in a psychologically plausible way. This means that we want the parser to perform in exactly the same way as people do. Should the parser parse sentences which people found incomprehensible, then it would not be fulfilling its role as a psychological model of normal human sentence processing.

We have noted that, by definition, people fail on garden path sentences, needing to employ conscious effort in their analysis. The parser, then, should also fail whenever it is presented with a garden path. This point was made by Marcus, who stated that any deterministic parser should fail on this type of sentence. He says:

"a deterministic parser ideally should take the garden path and become "stuck" at exactly the point at which people become conscious that they have been misled." [Marcus 1980, p. 204]

PARSIFAL was not built to handle garden path sentences, therefore it failed on some, but not on all.
In order to limit the model to parsing successfully only non-garden path sentences, those sentences on which we wish the parser to fail must be identified. Since garden paths are defined as sentences on which people fail, this should be possible by experimentation.

Assuming that the parser is limited in such a way, it can then be used to predict which sentences will cause problems in humans and, hence, give a definition of a garden path sentence as one which cannot be parsed deterministically by the model. The experiments which follow were designed to test the model against human performance, in its ability to distinguish and fail on garden path sentences.

3.1.1 The Garden Path Prediction of PARSIFAL

Let us look at the garden path prediction of Marcus's parser. PARSIFAL consisted of an Active Node Stack, where partially built items resided, and three buffers. Each buffer contained a word or constituent that could be represented by a single node. A buffer can then hold a word, NP, PP, VP, etc. NP stands for Noun Phrase, VP for Verb Phrase, PP for Prepositional Phrase and S for a Sentence, S- is an embedded S, toVP is an embedded VP with the auxverb 'to'. In the best situation, an ambiguous word will be in the first buffer and the lookahead will be two items (Buffers 2 and 3). When an NP is being built, these two items of lookahead will be single words and never whole NPs or larger items.

At the time the word "rocks" is being analysed in the fragment "the toy rocks...", the parser's state would be as below: (The symbol [noun], will mean a buffer which contains an item with the syntactic feature 'noun', symbols such as [rock] will mean a buffer which contains the word rock.)

Active Node Stack: NP the toy
Buffers: [rocks] [the] [child]
A sentence is predicted to be a potential garden path if the lookahead is insufficient to disambiguate the word correctly. Marcus did not recognise the concept of a potential garden path sentence. Instead of the pair of sentences we have called potential garden path sentences, he considered one a garden path and the other an ordinary sentence.

For the examples in this chapter, the lookahead is not sufficient to resolve the ambiguity. We have said that each time a person encounters a pair of potential garden path sentences, one and only one is a garden path. Marcus's method says that one will be a garden path, but cannot tell which it will be.

Because of this, his parser would arbitrarily choose one case (i.e., the noun usage) as preferred. For example, in the case of a word that could be a plural noun or a singular verb, his parser would always choose the plural noun usage. This would be correct for:

{26} The granite rocks by the seashore are eroded.
{27} The granite rocks by the seashore with the waves.

In [26], 'rocks' is used as a noun, and Marcus's approach would correctly predict [27] to be a garden path. But in:

{28} The statue stands in the park.
{29} The statue stands in the park are rusty.

where 'stands' is used as a verb in [28], his approach would incorrectly predict [28] to be a garden path and not [29]. Therefore, this approach would predict some sentences to be garden paths which are not.

Marcus's prediction of garden path sentences also tells us how "short" a garden path must be. In sentence [22], the disambiguating word is outside the three buffers. What happens when the word is in one of the three buffers? Marcus's prediction would say that it was not a garden path. If the parser used all the information in the three buffers, then for a sentence to be a garden path, there would need to be at least three words between the ambiguous point and the disambiguating word. Are all garden path sentences this long? This uncertainty suggests some possible counter
examples to the garden path prediction of the deterministic parser.

3.1.2 Why is it Wrong?

In fact, the well-known garden path:

{30} The prime number few.

is a counter example to Marcus's prediction. When PARSIFAL is analysing the word "number" the state will be:

Active Node Stack: NP the prime
Buffers: [number] [few] [ .]

Since the entire sentence fits into the three buffers, all information to analyse the sentence is available, but people do garden path on this sentence. The following may also be considered counter examples:

{31} The granite rocks during the earthquake.
{32} The jeep rocks are large.

Even though the number of words read, before the error is realised, is very small, it seems that people are aware of some confusion whilst analysing these sentences. Again, all the information for proper analysis is contained in the three buffers and it is not predicted to be a garden path by PARSIFAL. We will later see experimental evidence that these sentences cause problems for a reader and, therefore, must be considered garden paths.

This shows that Marcus's garden path prediction is inadequate. While PARSIFAL will correctly tell us some sentences are garden paths, it will also predict some sentences are garden paths which are not and judge as acceptable some sentences which are garden paths.

So how can we predict whether a sentence will be a garden path? What is happening that causes the garden path? Can our model be extended to answer these questions?
I feel that the previous definitions are inadequate because they fail to incorporate non-syntactic information, where non-syntactic information includes information such as the meaning of words, intonation and pragmatics. (We will discuss this definition more fully in the next chapter.) To account for the difficulty of garden path sentences, the following hypothesis is proposed to describe what people do when they encounter a potential garden path.

Semantic Checking Hypothesis:

"When a person encounters a situation which syntactic context implies might lead to a garden path, they decide which alternative to pursue based on non-syntactic information, instead of using lookahead. They do this without regard to the following words in the sentence. If their preference for this leads to an analysis different from that demanded by the remainder of the sentence, they will garden path."

In our example of a noun/verb word followed by a noun/verb word which is plural, a person would attempt to make a complex item name of the two words. The prediction now depends on a preference for noun/noun combinations, rather than on lookahead, for garden path sentences. I will expand this theory more in the following chapter.

Does this new explanation fit our examples?

People seem to like "prime numbers", but don't like "jeep rocks". I believe people will garden path if "prime number" is not a complex headnoun because it is a common construction, as with "aluminium screws" and "granite rocks". People will use "rocks" as a verb in "jeep rocks" since it is very difficult to imagine the complex item (jeep rock). Finally for the case represented by [28] and [29] (toy rocks), both constructions are equally possible, so some people would garden path on [28] and some on [29]. It is also very easy to bias this last case with context, etc., altering the
Marcus's approach to predicting garden path sentences was often correct because it seemed that one member was the garden path more often than the other. Although the non-syntactic choice can go either way, in many cases it tends to go the same way each time the sentence is encountered and, hence, one of the pair causes a garden path more often than the other. This fact explains why the previous garden path definitions seemed correct.

The Semantic Checking hypothesis predicts that

{33} The sentry stands on guard.
will not be a garden path, but

{34} The sentry stands are red.
will be.

This new theory makes definite predictions of what will be a garden path sentence, based on a person's preference for complex headnouns. Lookahead is predicted to have no affect on these examples, as the decision is being made on non-syntactic basis alone. In the next sections, an experiment designed to test this will be presented.
The "Semantic Checking Hypothesis" makes definite testable predictions. According to this hypothesis whether a person choses to combine a pair of nouns is dependent upon their preference to pair these words. We can model this preference for certain noun/noun pairs by using semantic markers. Our predictions can then be modeled by these markers.

An experiment was conducted to test the above predictions. The purpose of the experiment was to show that, of a pair of potential garden path sentences, one was a garden path whilst the other was not, also that subjects do not use lookahead to resolve the ambiguity leading to the potential garden path. Remember, a garden path sentence is one which seems to lead people 'down the garden path'. As our theory in the previous section predicts, for these sentences:

[35] The chestnut blocks are red.
[36] The chestnut blocks the sink.

one will cause a garden path and the other will not. Remember, a garden path sentence is one in which the reader initially mis-analyses a portion of the sentence and must exert conscious effort to correct this mis-analysis. In Section 2.2, we stated that this effort is normally detected by an increase in reaction time to the garden path sentence. Hence, it is predicted that the garden path sentence will lead to a longer reaction time than its non-garden path partner. If one sentence has a longer reaction time when compared with its partner, we can assume that the sentence which took longer caused the subject to garden path [Crain and Coker 1979].

In both the above sentences, looking just one word ahead is sufficient to resolve the ambiguity as the second buffer contains the disambiguating word. So, a person looking at the second buffer, could resolve the ambiguity and the reaction times should be the same for both sentences. If a person uses lookahead to resolve the ambiguity, then the person will not need to garden path on these sentences. Hence, if one sentence has a longer reaction time, we can conclude that the person garden
pathed and did not use lookahead.

If the theory is wrong, then responses to both sentences will take the same length of time. If the theory is correct, the response to one of the sentences will take longer than to the other. As the hypothesis states, context may have a strong effect on the understanding of these sentences. Therefore, it is predicted that the sentence of the pair which requires the longer time may vary from context to context for any one person and also from one person to another.

3.3.1 The Pre-Test

First, it was important to decide whether there was a generally preferred reading for the noun/noun combinations which were to be used in the experiment. This data would also be used to establish the semantic marker pairs for the parser. To collect examples, a written survey was conducted which consisted of 21 fragments, as below. Each subject was asked to complete the series of words such that they formed a complete sentence. The examples were presented in two different orders to control for order effects and 50 subjects participated. The examples were:

[37] the grappling hooks  [38] the aluminum screws
[39] the granite rocks   [40] the map pins
[41] the top hooks       [42] the truck handles
[43] the sentry stands   [44] the boy screws
[45] the cook handles    [46] the jeep rocks
[47] the statue stands   [48] the sniper pins
[49] the arm hooks       [50] the chestnut blocks
[51] the toy rocks       [52] the bike handles
[53] the plastic blocks  [54] the building blocks
[55] the cover screws    [56] the flower stands
[57] the book pins

The part of speech use of the noun/verb/plural word was then checked. A tally was made each time the word in question was used as a noun and as a verb. This provided an indication of the preference for these word pairs. For example, if almost all subjects completed "grappling hooks" using "hooks" as a noun, then this combina-
tion was considered to be preferred. If most of the subjects completed "boy screws" using "screws" as a verb, it was concluded that the verb reading was preferred.

The above examples were then divided into three groups. The first group contained pairs which were strongly preferred as noun/noun combinations, the second, pairs which were not preferred as noun/noun combinations and the third group were examples showing an equal split among subjects, or no bias. The results by group are as follows:

<table>
<thead>
<tr>
<th>Noun/Noun Preference</th>
<th>Noun Uses</th>
<th>Verb Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37] the grappling hooks</td>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>[38] the aluminum screws</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>[39] the granite rocks</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>[40] the map pins</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>[41] the top hooks</td>
<td>34</td>
<td>9</td>
</tr>
<tr>
<td>[42] the truck handles</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>Verb Preference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[43] the sentry stands</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>[44] the boy screws</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>[45] the cook handles</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>[46] the jeep rocks</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>[47] the statue stands</td>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>[48] the sniper pins</td>
<td>7</td>
<td>42</td>
</tr>
<tr>
<td>[49] the arm hooks</td>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>[50] the chestnut blocks</td>
<td>13</td>
<td>37</td>
</tr>
<tr>
<td>Equal Bias</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[51] the toy rocks</td>
<td>21</td>
<td>26</td>
</tr>
<tr>
<td>[52] the bike handles</td>
<td>25</td>
<td>23</td>
</tr>
<tr>
<td>[53] the plastic blocks</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>[54] the building blocks</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>[55] the cover screws</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>[56] the flower stands</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>[57] the book pins</td>
<td>20</td>
<td>28</td>
</tr>
</tbody>
</table>

3.3.2 Testing the Predictions

The first experiment was based on the above data and collected reaction times to test the hypothesis that, of the pair of potential garden paths, one is a garden path. The above marker pairs lead to the following predictions, which we can now test. Subjects were asked to read the series of words presented and decide
whether they were a complete sentence, or just a fragment. The noun/noun combinations used were taken from the above list of pairs. The examples were in two groups, those with a strong non-syntactic bias and those with no non-syntactic bias. The first group will be called the 'biased examples'. Examples were picked which had a strong preference for or against the combination, but could be used as both a noun/noun combination and a noun/verb combination. A sentence was constructed for each combination, using it in the non-preferred way. A partner sentence was then constructed which was matched in syllables and words, but used the combination of words in the preferred usage. For example:

[58] The sentry stands are green.
[59] The sentry stands on guard.

It was predicted that [58] would be a garden path. The pre-test has shown that 'sentry stands' is not preferred as a noun/noun combination, although it is used as such in the sentence. According to the theory, the subject would attempt to use "stands" as a verb rather than as part of the headnoun combination. This would lead to the wrong analysis, and a garden path would result. In each sentence, the features of the word following the target word (the word with the features noun, verb, plural, i.e., 'stands') were sufficient to disambiguate the target word.

It was predicted that [58] would take longer to understand than [59]. However, the difference in reaction time might simply have been the result of the different syntactic structures of the two sentences. Therefore, for each sentence of the above pair, a control sentence was constructed. This sentence was matched in syllables, but the word preceding the target word was changed to a make the sentence a definite noun use (for example by switching to an adjective); similarly, for the second sentence. This produced controls of the form:

[60] The pencil stands are green.
[61] The army stands on guard.

It was predicted that the control for the non-garden path sentence, [61], would take the same time as the non-garden path member of the potential garden
path pair, [59]. To ensure that the longer reaction time for the verb usage was not due to structural and processing differences, the two controls ([60] and [61]), were compared. If they required the same time, then it could be concluded that the verb usage did not require a longer time to process. If [61] took more time than [60], i.e., the verb usage took longer to process, then it was necessary to check that the effect was greater than would be due to structural differences. If three of the sentences required the same time to read and one of the test sentences required a longer time, then the predicted result would have been achieved.

The above prediction was only valid if the subject judged both examples of the test pair to be a sentence. If the subject judged [59] to be a fragment then, we can conclude the noun reading of 'stands' was used in both sentences. In this situation, the reaction time for both sentences ([58], [59]) should be the same.

For each test combination, we now have four sentences. To keep the task valid, four fragments were also included so that the subject was presented with an equal number of sentences and fragments.

The test pairs used were as follows:

[62] The prime number few.
[63] The bold number few.
[64] The sniper pins were rusty from the rain.
[65] The sniper pins the victim in the woods.
[66] The sentry stands are green.
[67] The sentry stands on guard.
[68] The chestnut blocks the sink.
[69] The chestnut blocks are red.
[70] The granite rocks during the earthquake.
[71] The granite rocks were by the seashore.
[72] The map pins onto the wall.
[73] The map pins are bright red.

Each pair above is a pair of potential garden path sentences. The first of each pair was predicted to be a garden path sentence on the basis of the results of the pre-test.

The second test group, known as the 'non-biased examples', consisted of sentences with no strong non-syntactic bias as in:
The toy rocks the child.
The building blocks the sun.
The book pins the author.
The bike handles in the boy’s hands.

To provide controls for each of the test sentences in this group, each had four complementary sentences with a definite noun use, a definite verb use, a definite fragment, and a definite complete sentence. For example:

[78] The toy rocks are soft  
[79] The toy rocks when hit  
[80] The toy rocks the child  
[81] The toy rocks the child pulls  
[82] The toy rocks the child gently

definite noun use  
definite verb use  
complete or fragment  
definite fragment  
definite sentence

Each of these sentences, [74-77] will be interpreted as a fragment or a sentence depending on the reading given to the noun/noun combination. The results of the pre-test indicate that across subjects, there was no agreed non-syntactic bias for these examples. However, this finding may be interpreted in two ways:

a) It may be that each subject has no preferred interpretation for e.g. “toy rocks”. If this is so, then both readings for [74] will be available to him. In the face of this ambiguity, he may take longer to decide it is a complete sentence than, say [79] or longer to decide that it is a fragment than [81]. In addition, whichever decision he makes for [80], he will not produce a reaction time difference between [78] and [79], or between [81] and [82], greater than that due to differences in processing times between the two types of sentences or between sentence and fragment since there is no interference in these cases from non-syntactic preference.

b) On the other hand, it may be that each subject does have a preferred reading for [80], but that the preferences differ between subjects. Those who prefer to interpret “rocks” as a noun will quickly (relative to [79]) interpret [80] as a fragment and those who prefer to interpret “rocks” as a verb will interpret [80] as a complete sentence as quickly as [79], but will garden path on [78].
The sentences were presented in two different orders to control for order bias. The biased examples were presented before the non-biased examples in both orders. Each subject was given one order and all types of sentences were randomly ordered. All subjects were tested on all sentences.

3.3.3 Subjects

Forty-seven undergraduate students from Edinburgh University participated in the experiment. Most were from the Psychology department. All were unpaid volunteers and native speakers of 'British English'. Approximately half the students were tested on each order.

3.3.4 Procedure

A Commodore Pet micro computer was used to collect reaction times for the subjects. Each person sat in front of the display screen (VDU) and the instructions were read aloud by the experimenter. The sentences were then displayed on the center of the VDU. The subject was asked to decide whether the series of words presented was a complete sentence, or just a fragment. If the subject thought the series of words was a fragment, he pressed a key with his right hand. If he felt it was a sentence, he pressed a key with his left hand. Reaction times were measured from the presentation of the sentence until the response. The subject was told that all series of words would be syntactically and semantically well formed. He was first given sixteen practice sentences. Sentences were presented in groups of twenty with a short rest between each group and there were eighty-six examples in total. After the test, each subject was asked how he had done. Those reporting they did badly, or had trouble, were noted for later analysis.
3.3.5 Analysis of the Data

The reaction times and judgements of all the subjects were checked for irregularities. Remember that for each test pair in the biased examples, 4 sentences and 4 fragments were constructed. The sentence/fragment judgements of each subject were checked for the biased examples and the number of judgements differing from this design were tallied. The average number of different judgements per subject was 5 sentences (10%). Sentences with different judgements were not removed from the analysis. 18% of the subjects had differences in judgement on more than 11 of the sentences. For example, some of these subjects listed all examples as complete and others as all fragments. As the error rate for these subjects was over 20%, they were removed from the analysis.

If the test word was used as the same part of speech in both examples, then no "garden pathing" should have occurred and both examples should require the same time to read. Therefore, if one of the examples was judged to be a fragment, the prediction was that both sentences of the pair would require the same length of time. Very few subjects made this altered judgement and their results were not separated from the others. As a result some of the variances are slightly larger than they would be if these people were checked separately. Exceptionally long or short reaction times were regressed to their group's mean but without losing their significance or relative ordering.

For the "granite rocks" examples, approximately half the subjects considered one of the test sentences a fragment. It was only predicted that one of these would be a garden path when both examples were judged to be complete sentences. In order for this to happen, the noun/verb/plural word would have to be used as a noun in one sentence and a verb in the other sentence. If the example was judged to be a fragment, then it is assumed that the noun/verb/plural word was used as the same part of speech in both sentences.
For the "granite rocks" examples, the responses were split into two groups and the same analysis as above was performed on each group. The time reported below is for the test pair which was predicted to be significantly different. The results of the F test [Snedecor and Cochran 1967] among the sentences which were predicted to be the same was less than three, indicating that there was no significant difference among them.

A separate analysis of variance with repeated measures was performed for each set of four sentences in the biased group. (i.e., the G.P., non-G.P. and 2 controls). For the groups which the F test showed were significantly different, the Newman-Keuls test was used to determine which pairs were significantly different. These results are reported below.

A Student's t-test was performed on each of the test pairs and across their corresponding controls. These results also supported the findings presented below. A further analysis of variance was performed using the model Time = Reading Rate(Person) * Difficulty(Sentence) * Error. This model was transformed to log(Rate) + log(Difficulty) + log(Error). The times for the sentences were fitted against this linear model and the standard error was then checked for significance. A plot of the error was performed which confirmed that it was normally distributed. This model supported the findings presented below. These tests were made on each order and across both orders.

3.3.6 Results

The results of the test sentences are shown below. I will present each group of four examples and the mean reaction time in 100 ths/sec for all examples combined from both orders. The star (*) indicates that the sentence was predicted to take longer. All results are at the 1% level of significance, meaning that there is less than a 1% chance that the sentence will be found to be significantly different by the
F test if they are in fact, of equal mean reading times.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Reaction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[83] The bold number few.</td>
<td>222</td>
</tr>
<tr>
<td>[84] The prime number few.</td>
<td>315</td>
</tr>
<tr>
<td>[85] The bold number two.</td>
<td>254</td>
</tr>
<tr>
<td>[86] The prime integer two.</td>
<td>235</td>
</tr>
</tbody>
</table>

All subjects judged [84] to be a fragment. Hence the prediction was that all four sentences would require the same time to read. Although the time for [84] appears to be longer than the other reaction times, these examples were not significantly different, $F(3,102)=3.7, p<.01$. This is because of the great deviations present between subjects.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Reaction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[87] The shotgun pins were rusty from the rain.</td>
<td>341</td>
</tr>
<tr>
<td>[88] The sniper pins were rusty from the rain.</td>
<td>*363</td>
</tr>
<tr>
<td>[89] The sniper pins the victim in the woods.</td>
<td>268</td>
</tr>
<tr>
<td>[90] The sniper guards the victim in the woods.</td>
<td>254</td>
</tr>
</tbody>
</table>

The reaction times for these sentences were significantly different, $F(3,102)=22.4, p<.01$. The Newman-Keuls test showed that [88] was significantly different from [89] and [90]. It also showed that [87] was significantly different from [89] and [90]. This supports the prediction that [88] is a garden path, but does not rule out the theory that the time difference is due to a structural processing difference.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Reaction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[91] The pencil stands are green.</td>
<td>188</td>
</tr>
<tr>
<td>[92] The sentry stands are green.</td>
<td>*258</td>
</tr>
<tr>
<td>[93] The sentry stands on guard.</td>
<td>193</td>
</tr>
<tr>
<td>[94] The army stands on guard.</td>
<td>199</td>
</tr>
</tbody>
</table>
These sentences were significantly different, $F(3,102)=7.3, p<.01$. The Newman-Keuls test showed that [92] was significantly different from the other three, as predicted.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Reaction Time (100 ths/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[95] The wooden blocks are red.</td>
<td>179</td>
</tr>
<tr>
<td>[96] The chestnut blocks are red.</td>
<td>256</td>
</tr>
<tr>
<td>[97] The chestnut blocks the sink.</td>
<td>191</td>
</tr>
<tr>
<td>[98] The stopper blocks the sink.</td>
<td>242</td>
</tr>
</tbody>
</table>

These examples are significantly different, $F(3,102)=19.2, p<.01$. The Newman-Keuls test showed that [96] was significantly different from [95] and [97]. It also showed that [98] was significantly different from [95] and [97]. Although sentence [98] takes longer to process than [95], presumably due to structural differences, the garden path example corresponds to the faster sentence. This rejects the theory that the difference in reaction time is due only to structural processing differences for these examples.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Reaction Time (100 ths/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[99] The table rocks during the earthquake.</td>
<td>310</td>
</tr>
<tr>
<td>[100] The granite rocks during the earthquake.</td>
<td>418</td>
</tr>
<tr>
<td>[101] The granite rocks were by the seashore.</td>
<td>304</td>
</tr>
<tr>
<td>[102] The biggest rocks were by the seashore.</td>
<td>371</td>
</tr>
</tbody>
</table>

These examples were significantly different, $F(3,42)=5.6, p<.01$. The Newman-Keuls test showed that [102] was significantly different from [99] and [101]. Comparison shows that [99] was faster than [102], suggesting that some time difference was due to a structural processing difference. But this difference suggests that [101] should be the slowest sentence, rather than the fastest. These results again support the theory.
These examples were significantly different, F(3, 102)=8.9, p<.01. The Newman-Keuls test showed that [103] was significantly different from [105] and [106]. Sentences [104] and [105] were not significantly different at the .01 level. This result was unexpected. The result may be due to a garden path effect in [103] or to structural differences. The experiment does not provide enough data to test these hypotheses.

The error rate was also computed for the non-biased examples. These sentences were at the end of the experiment, and the subjects were observed to be confused, tired and making mistakes. The error rate for these examples was 45%, supporting this observation. Because of this, the results of this portion of the experiment are not reported here.

3.3.7 Discussion

Some of the results supported our theory. They showed that the reaction times were indeed significantly different at the .01 level for the test pairs. This indicates that one of the examples did cause a garden path. However, if Marcus's theory, as applied to humans, was correct then the reaction times should have been more or less the same. Therefore, Marcus's theory is incorrect and something else must be happening. This supports the theory that non-syntactic information is used to resolve the ambiguity in these potential garden paths. If the subject had used lookahead, then the reaction times for the test sentences would have been the same. As they were different, it is concluded that lookahead was not used.
It is very hard to test this theory. Since the pairs can be biased by context, then it is easy for a subject to change in the experiment and show no overall effect. Whenever a subject did not understand a test example, he could press 'fragment'. Many of the subjects considered 'the statue stands he saw' as unacceptable and admitted judging it to be a fragment for this reason. Sentences that are semantically odd can also take a longer time to read as the subject attempts to understand the sentence. Finally, the subject may be inconsistent in his reaction to odd sentences, again causing great variance.

Why is non-syntactic information used to handle the case of a word which can be a plural noun or a singular verb? For which other cases is non-syntactic information used to resolve the ambiguity and why in those places? Is non-syntactic information used to resolve all ambiguities, or only some ambiguities? If only some, is there a simple principle which tells us when non-syntactic information is used and why? We shall investigate these questions in Chapter 4.

3.4 Summary So Far

We have seen that PARSIFAL does not properly predict garden path sentences. In Section 2.4, I explained PARSIFAL and presented a modified version of it, called ROBIE. The main difference between PARSIFAL and ROBIE involves the use of non-syntactic information to resolve certain ambiguities. Conceptually, ROBIE consists of a syntactic processor and a non-syntactic processor. PARSIFAL attempted to do as much as possible with the syntactic information it had. Conceptually, in ROBIE, non-syntactic information is to be used in key situations to prevent the syntactic processor from making a, possibly wrong, decision. These key situations are exactly those in which an ambiguity cannot always be correctly resolved with two buffers of lookahead. Although the non-syntactic tests themselves have not been fully implemented in ROBIE, the timing of these tests has been clearly identified.
In the case of plural garden paths, the correct part of speech to use cannot always be correctly selected with only two windows of lookahead. The syntactic processor may not have enough information in some instances to select the correct part of speech. Instead, the non-syntactic processor is used to decide which part of speech to use.

This experiment has shown us that there exists a preference for certain modifier-noun pairs. ROBIE models the results of this experiment with the use of semantic markers to judge the acceptability of modifier-noun combinations. It is known that in the situation of a noun/verb word followed by a noun/verb which is plural, there is no strategy, based on limited lookahead alone, which can always correctly resolve the ambiguity. Therefore the Semantic Checking Hypothesis indicates the use of non-syntactic information. ROBIE has a list of preferred semantic markers which it uses to decide whether to use the latter word as a noun or a verb. If the markers are inadequate to make this disambiguation, then a syntactic heuristic is used, as will be explained in Chapter 8.

In ROBIE, each grammar rule alters the syntactic structure of the sentence and makes a contribution to its semantic interpretation. The parser is also more simple than Marcus's, using only two buffers of lookahead and fewer functions in the grammar. Each of these differences will be explained in detail in the following chapters.

In the next chapter, we will investigate the situations in which non-syntactic interaction is needed to resolve ambiguities and motivate several of the differences between ROBIE and PARSIFAL we have seen in this chapter.
Non-Syntactic Interaction

In this chapter, we will explore syntactic/non-syntactic interaction. More specifically, we want to decide WHEN non-syntactic information is used to make parsing decisions, and WHY at these times? In the last chapter, we saw a case of lexical ambiguity that can lead to a garden path. We have also concluded that non-syntactic information may be used to resolve this case. This would suggest a description of the garden path effect based on non-syntactic preference. In this chapter we will investigate in what other situations non-syntactic information could be used to resolve ambiguity and propose a new definition of the garden path effect.

What is non-syntactic information? In formal logic, semantic information is used to refer to that information which establishes the truth or falsity of a sentence. The information which can be used to establish these truth conditions is normally interpreted to be all information other than syntactic information. In other words, semantic information is all 'non-syntactic' information. In computational linguistics, non-syntactic information is interpreted to include the meaning of words, the meaning of a sentence, the knowledge of the world, the knowledge of the context and discourse, the information from intonation, and any additional information other than syntax that a person may use in processing. We will use non-syntactic information in this sense throughout the thesis. Semantic interpretation will be used to refer to the process of collecting the information necessary to establish the truth of falsity of a sentence.

In this thesis, we are concerned only with the nature of the syntactic level of processing and its interface with the non-syntactic processor. We will treat the non-syntactic processor as a black box, of which we can ask questions and receive simple answers. We will try to define what questions non-syntactic information is
used to answer and in what situations, but we will not explore how these questions are answered. Since we are concerned with the syntactic processor, in this chapter we will consider only syntactic ambiguities. The important role non-syntactic information has in parsing VP constituents will be explored in Chapter 10.

4.1 Each Rule Makes a Semantic Contribution

Before a non-syntactic processor could be used to resolve an ambiguity, it would have to have knowledge of the sentence. Thus, the non-syntactic processor must have knowledge of any sentence, whether non-syntactic decisions are made during the parse or not. To accomplish this in ROBIE, every grammar rule communicates information to the non-syntactic processor. Each rule performs several operations in the rule body. One of these operations is a message for the semantic interpretation of the sentence. This "rule by rule" approach is similar to that proposed by several linguists. [Montague 1972,1974], [Gazdar 1979,1980a,1980b,1980c], [Bach 1977].

It should be noted that the strategy followed in ROBIE is not as strict as the rule by rule approach, which states that each rule of syntax makes a functional contribution to the semantic interpretation of the sentence and at the end of the sentence this is all that is necessary for its interpretation. In ROBIE this restriction is not necessarily adhered to and further non-syntactic processing may be necessary.

As each syntactic rule sends a message regarding the semantic interpretation of the sentence to the non-syntactic processor, the non-syntactic processor does not normally provide any feedback to the syntactic processor, but merely "takes notes" of what is happening in the parse. Each grammar rule then makes a contribution to the semantic interpretation of the parse, although, for some rules, this contribution may be trivial.
The non-syntactic processor provides feedback whenever the semantic contribution does not make sense or is anomalous. If the contribution makes no sense, then two modes of feedback are possible. In the former mode, the parse is continued anyway, while in the second, the parse is stopped. These both seem psychologically plausible since the same thing appears to occur in human parsing. i.e., people can recognise as syntactically well-formed, semantically ill-formed sentences, as well as normal sentences. The sentence:

[107] Colourless green ideas sleep furiously.

is semantically anomalous, but still recognisable as syntactically well-formed. Being able to recognise this is an example of the non-syntactic processor continuing despite being unable to make sense of the sentence. In another context, this sentence might stop the reader and be impossible to understand. This would be an example of the non-syntactic processor stopping the parse.

By having each rule make a non-syntactic contribution, the above data can be accounted for. In ROBIE, every grammar rule makes a contribution to the non-syntactic interpretation of a sentence.

There are many non-syntactic ambiguities that do not affect the syntactic structure of a sentence. For example:

[108] The punch at the party gave Mary a hangover.
[109] The punch at the party gave Mary a concussion.

In these examples, the word sense of “punch” changes, but the parse tree for the sentences and the part of speech of the word “punch” remains the same. This is a problem at the non-syntactic level of interpretation and not at the syntactic level of analysis. As we are concerned with the syntactic processor, this problem is not relevant to our discussion here, but it will be looked at briefly in Chapter 10.
Non-syntactic information is not needed to decide that the following sentences are ungrammatical. (We will use a star (*) to indicate that a sentence is ungrammatical.)

[110] *The boy are run on the path.
[111] *The boy has kiss the girl.

These sentences are ungrammatical because of number disagreement between the subject and verb and violation of the rigid constituent structure of the verbal cluster. Nor is non-syntactic information needed to show that the ungrammaticality of the previous examples has been corrected:

[112] The boys are running on the path.
[113] The boy has kissed the girl.

One reason for this is that these sentences do not have any global syntactic ambiguity. However the following sentences have a global syntactic ambiguity and non-syntactic information is needed to select a unique meaning, if there is one.

[114] I saw the man on the hill with the telescope.
[115] I told the girl that I liked the story.

In [114], the PP (with the telescope) could modify 'the hill', 'the man' or the verb 'saw'. In [115], 'the story' could be liked or 'the girl'. Which meaning is desired cannot be selected on a purely syntactic basis. In the first set of examples, non-syntactic information is not needed to analyse the sentence syntactically, because the sentence is not syntactically ambiguous. In the second set of examples, non-syntactic information is needed, to select a unique interpretation, because the sentences are syntactically ambiguous. These sentences are known as globally ambiguous as there is not enough syntactic restriction in the sentence to remove all syntactic ambiguities. Most ambiguities are merely local in that the ambiguity can be resolved with the syntactic constraints imposed by the surrounding context.

In the unambiguous examples, if the semantic interpretation of the sentence does not make sense, no other non-syntactic interpretation is possible. In the
ambiguous examples, non-syntactic information is needed to decide which of several possible meanings is appropriate to the sentence. It is on this basis that we can distinguish these examples. In the first class, if the meaning is unclear, the sentence is uninterpretable. In the second class, if the meaning is unclear, there may be another acceptable interpretation.

For the parser, in the former case, there is no alternative but to complete the parse and fail or succeed as may be. It is in this case that the non-syntactic processor merely "takes notes".

In the latter case the non-syntactic processor has to choose between alternatives, as it is used to resolve a syntactic ambiguity. In the rest of this chapter, we will concern ourselves with the point at which the parser makes the decision between alternatives.

ROBIE, with its limited lookahead, can be used to predict when this decision must be made. If there is an ambiguity that cannot always be resolved with two windows of lookahead, then non-syntactic information may be used to resolve that ambiguity.

Which sentences fit into the first class and which into the second class? How can the parser tell when non-syntactic information is needed to make a decision and when it is not? This question can be rephrased to "WHEN could non-syntactic information be needed to make parsing decisions and WHY should it be needed then?"

4.3 Which Ambiguities are Resolved with Non-Syntactic Information?

In the last chapter we saw a case of syntactic ambiguity which people could have resolved on the basis of non-syntactic information. In this section, we will see several more cases where people are thought to resolve a syntactic ambiguity on a non-syntactic basis.
4.3.1 PP Attachment

It is generally accepted that PP attachment ambiguities must be resolved on a non-syntactic basis and not on a syntactic basis alone.

For example, to decide the PP placement of:

[116] Put the block on the cube in the box
[117] I saw the man on the hill with the telescope.

one must know the current context. In these examples, non-syntactic information is needed to decide whether the PP (in the box) modifies 'the cube' or the main verb, 'put'; similarly for 'with the telescope'. Clearly this decision cannot be reliably based on syntactic information alone and PP attachment needs to use non-syntactic information when choosing between alternatives. Because [116] is globally ambiguous, there is no strategy, based on syntax alone, even with unlimited lookahead, that can select the "correct reading". We have agreed that people must use non-syntactic information to resolve this ambiguity.

4.3.2 Reduced Relative Garden Paths

In the last chapter, we saw the sentence:

{118} The horse raced past the barn fell.

In this famous garden path, it is accepted that people mis-analyse the word 'raced' as a main verb and must then reanalyse it as the start of a relative clause. Examples of this type are not hard to find.

{119} The boat floated down the river sank.
{120} The professors instructed about the exam were confused.
{121} The boy read the story liked it.
{122} The postman delivered junk mail was unhappy.

For each of these there is another similar sentence that provides a pair of potential garden paths.
For example:

[123] The horse raced past the barn.
[124] The boat floated down the river.
[125] The professors instructed about the exam.
[126] The boy read the story.
[127] The postman delivered junk mail.

The start of the potentially ambiguous part of each of these sentences would match the pattern [noun][verb,ed]. That is, there is a noun followed by a verb with the feature "ed" in each sentence. It is this pattern that indicates that the verb can be either a main verb or start a reduced relative clause. If the reader chooses the wrong analysis, then a garden path results. This grammatical situation always produces a pair of potential garden paths. There is no strategy, based on syntactic information and limited lookahead that can tell which of each pair will be a garden path. As with the "plural garden paths", it is possible to construct examples with a theoretically unlimited number of words between the ambiguous point and the disambiguating word (see Chapter 9 for an example).

In Section 3.1, we saw several explanations why one of the pair would be a garden path. These claimed that the relative clause usage would always be a garden path. Such theories cannot explain why some of the relative clause readings do not cause a garden path. Does our new theory explain which of these are garden paths and why?

It has been suggested [Crain and Coker 1979], [Coker and Crain 1979], that this case is decided on a non-syntactic basis. They considered examples such as:

{128} The students instructed about the exam were confused.
{129} The professors instructed about the exam were confused.

They felt that one of these was a garden path, and the other was not, also that the reader made the verb the main verb of the sentence when the initial NP would fit the subject slot of the verb. A garden path would result whenever this verb should have been the start of a relative clause.
To test their theory, they performed the following experiment. The example sentences were presented in monotone through earphones to the subjects, who were then asked to judge the "truth value" of a possible paraphrase. Reaction times were collected for the judgement on the paraphrase to be made. Examples were of the above form, all of which were intended to be full sentences. If no non-syntactic decision should be made, then they felt the subjects would garden path on most of the examples. If the decision was made on a non-syntactic basis, then only half of the examples would be garden paths.

Their results showed that only about half of these examples (46%) did cause garden paths. This indicates that the explanations of Fodor, Bever and Garret [1974] in Section 3.1 are inadequate.

Their experiment showed that [128] was not a garden path sentence, whilst [129] was. They then concluded that the subject perceived whether the initial NP would fit the subject slot of the verb. If it would, the subject would accept the verb as the main verb and continue. If, however, the initial NP did not fit the subject slot, then the subject would make it a reduced relative clause. The subject would garden path if this non-syntactic decision led to the wrong choice.

Horses normally race, so "raced" is accepted as a main verb. Likewise, boys read more often than they are read to. Professors instruct normally, but students get instructed, so the analysis here is different and people do not garden path on both of [128] and [129], nor both of the following:

{130} The tenant delivered junk mail threw it in the trash.
{131} The postman delivered junk mail threw it in the trash.

According to Crain and Coker's theory, not all potential garden paths will be actual garden paths, but only some of them. The prediction of which will be garden paths is based on the non-syntactic "fit" of the NP into the subject slot. No strategy based on limited lookahead alone can always resolve this ambiguity. We have
seen that people seem to use non-syntactic information in this situation. This gives us another example of non-syntactic information probably being used to assist decisions during the parse, which fits our current theory.

4.3.3 THAT Garden Paths

The next examples we will look at are based on the ambiguity involved in the word "that".

132 I told the girl that I liked the story.
133 I told the girl that I kissed the story.
134 I told the girl that I know the apple.
135 I told the girl that I liked in 1978 the story.

The problem in these examples is that the word "that" can be either a complementiser for an embedded sentence or a relative pronoun for a relative clause. If the reader takes the word "that" to start the wrong type of clause, an error can result. The first three examples shown here differ from the previous types in that no matter which part of speech one makes "that", it is possible to analyse the sentence syntactically, even though it may be semantically anomalous. The problem arises when one version has been chosen, where semantically the other is required.

Sentence 135 often leads to a garden path. In this sentence, if "that" is used as a complementiser, then the PP 'in 1978' will be in a syntactically unacceptable position. Notice that the two buffer lookahead is insufficient to choose the correct reading. It may be that readers choose the 'complementiser' version and, because the PP is unacceptable, they garden path on this example.

Both readings are possible in 132. This means that, like the PP attachment examples, the syntactic processor, even with all the information in this sentence, cannot decide which reading to use. It is accepted that the reading eventually chosen depends on the discourse. If discourse can affect how the sentence is interpreted, then the non-syntactic processor must make the decision, not the syntactic processor. If the syntactic processor were making the decision, it would always make the same
decision and discourse would not affect which reading was chosen.

There is little data on the amount of processing difficulty people have with these examples. As suggested above, should the wrong choice be made, the sentence could still be finished, even though the 'rule by rule' semantic interpretation may have rejected it. For these examples, the garden path will only be noticed if the non-syntactic processor stops the parse when it makes no sense. If the non-syntactic processor is continuing, despite being unable to make sense of the sentence, these examples will be noted as strange, but will not cause garden paths. Similarly, for people, these examples will seem odd, but not cause a garden path.

Normal syntactic parsers have great difficulty with this problem. There seems to be no syntactic strategy that will always decide which use of 'that' to use. The explanation seems to be that people resolve this case using non-syntactic information. Since a syntactic parser uses only syntactic information, it has difficulty with this problem. People seem to be able to resolve this ambiguity easily because they do so on the basis of non-syntactic information. We will return to problems of 'that' in Chapter 5.

To find out whether [132] causes processing difficulty, it was tested in the second experiment. The reaction times showed that people took less time to analyse this than for an unambiguous relative clause reading (see Section 4.8). The subjects informally questioned afterwards said that they took 'that' in [132] to start a sentence, rather than a relative clause. These subjects said they were not aware of another reading. This suggests that [132] does not cause processing difficulty and the embedded sentence reading is preferred with no context. Again, no strategy based on limited lookahead alone can always resolve this ambiguity. Although it seems that non-syntactic information is used to resolve this ambiguity, we do not have enough evidence to establish whether it is non-syntactic information alone.
4.4 The Theory of WHEN and WHY

4.4.1 When is Non-Syntactic Information Used?

We have now seen several cases where a choice between alternatives is probably made by a non-syntactic processor. These are PP attachment, reduced relative garden paths, the 'plural garden paths' that were the subject of the last chapter, and "that" garden paths. For PP attachment, the question is; "does the PP modify the nearest NP or some other NP, or something else?"? In the reduced relative, the problem is; "do we have a relative clause, or a main verb?". For the 'plural garden path' case, the problem is; "do we have the end of the NP or the main verb". In the "that" examples, the problem is; "do we have a relative clause or an S-"? Notice that all of these are concerned with finding the end of the noun phrase.

For the examples we have covered so far, these are the only situations where there is a choice of alternatives. All of these examples could also be characterised by the question 'Do we have the end of the NP, or shall we make a larger NP?' This is exactly the "end of NP problem". Hence, the answer to the WHEN question is: 'when finding the end of the NP'. All other situations that the current parser handles can be resolved without the non-syntactic processor making a decision.

4.4.2 Why must the Decision use Non-Syntactic Information?

For each of the cases we have just seen, there exists a pair of potential garden path sentences. For each of these, we stated that there is no strategy, based on syntactic information and limited lookahead alone, to resolve all possible cases correctly. It is also not possible to predict which of each pair will be a garden path based on this information. To be able to resolve the ambiguity and predict the garden path sentence for all examples, the lookahead would have to be arbitrarily long. Hence, for each of these, there is no strategy that a deterministic parser could follow, based purely on syntactic information, to get these examples correct.
This, then, explains the WHY part of our question. Non-syntactic information may be used to resolve the ambiguity because the syntactic processor, using only syntactic information and limited lookahead, cannot choose the "correct" meaning for all situations.

<table>
<thead>
<tr>
<th>Type</th>
<th>Modifies</th>
<th>Syntax Help?</th>
<th>Non_Syntax Decides?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP Attachment</td>
<td>NP or other</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Reduced Relative</td>
<td>NP or other</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Noun/Verb/Plural</td>
<td>End of NP</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>That</td>
<td>NP or other</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Therefore, the theory of When non-syntactic information is used to resolve ambiguity and Why at those times, is this:

**WHEN** - When syntactic information with limited lookahead is not sufficient to determine whether a NP should be closed or not.

**WHY** - Because the syntactic processor could err.

ROBIE can handle all the areas I have investigated, without the use of non-syntactic information to make decisions, except in the above cases. It was predicted that non-syntactic information would be needed in those situations which cannot be resolved with two windows of lookahead. Conceptually the parser uses the non-syntactic processor to decide when to end a NP. ROBIE does not identify these cases automatically, rather, in the current implementation, these checks have been manually inserted into the grammar rules which require them. These rules are the ones responsible for the decisions in this chapter.

This also indicates the limits of the syntactic processor. It handles all situations that its lookahead is sufficient to resolve in a deterministic way. If the lookahead is not sufficient to resolve the ambiguity in general, then the non-syntactic processor chooses between the alternatives. It is only these sentences that are eligible to be garden paths.
We can now make a prediction of the cases in which non-syntactic information can influence the parse. If it is possible to build a minimal pair such that the distinguishing point is outside the two buffers, it is predicted that the decision will be based on non-syntactic information.

4.5 The New Garden Path Explanation

We now have a new explanation of a garden path. Previous explanations were based on syntactic information alone and have been shown to be inadequate. The garden path arose when the reader built the wrong structure and then had to "backtrack" to correct it. The proposed explanation for a garden path is this:

If a situation is encountered, in the course of reading a sentence, such that there is more than one alternative, but syntactic information, with limited lookahead, is not sufficient to guarantee the correct choice, then the reader uses non-syntactic information to choose one of the alternatives. The reader does this without regard to the following words in the sentence. If subsequent processing leads to an analysis differing from that demanded by the remainder of the sentence, the reader will "block" later in the sentence.

This explanation means that a garden path is now based on the necessity for a non-syntactic preference. Thus the prediction of what is and what is not a garden path is more sensitive to context than the previous syntactic accounts and explains why not all potential garden paths cause a garden path.

4.6 How to Recover from a Garden Path

Given the above explanation and account of how people garden path, how are garden paths detected and how do people recover from a garden path? The proposed answer will be given in the context of the parser. As the parser analyses a sentence, it processes the words, building structure and activating the appropriate pack-
ets. At some point the sentence will be detected as a potential garden path. This is realised when two rules of equal constraint match. The non-syntactic processor is then called to select which of the two rules to execute. Later in the parse, the parser may discover it has made an error, as no rule matches. This is the point of detection when the psychological "block" is struck. How is recovery then made?

When subjects read a garden path sentence during the experiment, it was surprising how much trouble they had. In experiment two, many subjects told me they could not "figure out" the sentence when it was a garden path. This shows that recovery from a garden path is, indeed, very hard. If recovery from garden path sentences was as simple as syntactic backtracking, then the subject should not have so much trouble finding the correct reading. I feel this difficulty may occur because the subject is very reluctant to reverse the decision at the non-syntactic level.

To recover, the error component takes control of the two processors. This research has not specifically investigated this component, hence it is not possible to say what it does. However, the error component could go back to the last non-syntactic choice point. Since it knows of potential garden paths, it may have saved the state at the time the non-syntactic choice was made. All non-syntactic choices are binary decisions and the error component may then simply re-start the analysis from that point, but with the opposite decision. The parse would then continue as before.

In the second experiment, for the subjects that only had a small amount of trouble, the time to read the garden paths was about 1.5 times the time for the normal sentence. Also, the point at which the ambiguity was incorrectly resolved for these sentences was approximately in the middle of the sentence, so according to the above theory, the person would have to re-scan the second half of the sentence at least. This is consistent with the time they have taken.
In this chapter we have answered the questions of WHEN and WHY non-syntactic interaction may be used by people to make parsing decisions. This has led to a different prediction of what a garden path sentence is and when it will occur.

In the previous chapter, we investigated a case of lexical ambiguity that could lead to a garden path. This investigation suggested that non-syntactic interaction is required to resolve the ambiguity. In this chapter, we have seen that this suggestion also applies to other end of constituent ambiguities, many of which could also lead to a garden path. We have also modified the model to account for these results. Whenever an ambiguity arises, which the syntactic processor cannot guarantee to resolve correctly because of its limitations, the non-syntactic processor could choose which alternative to use. This is contrasted with a theory that suggests that the non-syntactic processor is only used when the syntactic processor has incorrectly resolved an ambiguity. The main difference is that they interact when the syntactic processor might incorrectly resolve an ambiguity, rather than when it has incorrectly resolved an ambiguity.

In the next two chapters, we will investigate lexical ambiguities that do not lead to garden paths and see if the model, as it is now developed, is adequate. However, before we continue our investigation, it is necessary to present experimental data on some of the sentence examples which will be presented. The following experiment was designed to provide non-subjective data on how people react to certain sentences to be used in the rest of the thesis, as well as to provide some interesting data for future investigations.
4.8 The Second Experiment

4.8.1 Purpose

The purpose of this experiment was to test several types of sentences which were believed to be unusual and for which little experimental data was available, especially the garden path effect. The test sentences were divided into several groups, some being examples of global ambiguity, some classic garden paths, and some curious examples, reactions to which were of interest. Many of these examples of ambiguity are discussed in Chapter 5, while the garden path effect has been discussed in Chapters 2-4.

As in the first experiment and as was explained in Section 2.2, if the reader encounters an ambiguity which leads to a mis-analysis or confusion, it is believed that more conscious effort will be required than to read a sentence without this ambiguity. This use of conscious effort can be detected by an increase in reading time for this task.

4.8.2 Task

This experiment was a simple collection of reading times, as used by [Cirilo and Foss 1980], [Graesser, Hoffman, and Clark 1980], [Just and Clark 1973]. The subject sat in front of a Visual Display Unit (VDU) and instructions were presented on the screen of the VDU. The subject was then presented with a series of sentences. When he had read and understood each sentence, the subject pressed a key on the keyboard.

The instructions were as follows:

You will be shown a series of sentences. You should read each one, and WHEN you have READ it AND understood it, press the RETURN key - it is near the right of the keyboard. Doing so will show the next sentence in the series. All sentences are syntactically and semantically well-formed.

Press RETURN when you are ready to start ...
A Tekak micro-computer was used to collect reaction times. The time was measured from the presentation of the sentence until the key was pressed. The next sentence was presented immediately.

4.8.3 Subjects

Twenty-two undergraduate students from the University of Edinburgh participated in this experiment. All were unpaid volunteers. None of the subjects were familiar with the notion of global ambiguity or garden path sentences.

4.8.4 Examples

The examples were arranged in two orders to control for sequence effects. Each subject was given one order and each order contained all sentences, the sequence of the sentences being random. Between the test orders, the sequence of each sentence with its control was reversed. The predictions varied with each group of sentences. These will be discussed with the results. The subject was given ten practice sentences before the test sentences appeared. The sentences tested were as follows:

Global Ambiguity
[136] They can fish.
[137] They can walk.
[138] They can fruit.
[139] He looked up the street.
[140] He looked up the address.
[141] He lookup up the hill.
[142] They are flying planes.
[143] They are flying fish.
[144] They are flying home.
[145] Kissing aunts can be boring.
[146] Kissing aunts are boring.
[147] Kissing aunts is boring.
Garden Paths

The boat floated down the river sank.
The boat floated down the river quietly.
The horse raced past the barn fell.
The horse raced past the old barn.
The cotton clothing is made of grows in Alabama.
The cotton clothing is made in sunny Alabama.

"That" Ambiguity
I told the girl that I liked the story
I told the girl whom I liked the story.
I told the girl the story that I liked.

"Have Ambiguity"
Have the students take the exam.
Have the students taken the exam.
Make the students take the exam.
Did the students take the exam?
Have the students taken in the back room finished yet.
Have the students taken in the back room finished off.

Sentence Initial "That"
That deer ate everything in my garden surprised me.
That deer ate everything in my garden last night.
The deer ate everything in my garden last night.
That birds ate everything in my garden surprised me.
That bird ate everything in my garden last night.

"What" Ambiguity
What little fish eat is worms.
What little fish eat big worms?
The little fish eat big worms.

PP Attachment
I saw the man on the hill with the telescope.
I saw the man on the hill with the large tree.
I saw the man with the large hat on the hill.
I know the man on the hill with the telescope.
I know the man on the hill with the large tree.
I know the man with the large hat on the hill.

Noun/Modal Ambiguity
Let the paper note be read.
Let the paper will be read.
Will the paper can be re-used?
Will the paper bin be re-used?
4.8.5 Results

A Student's t-test was performed on each pair of sentences, for each order and for the combined results. Listed below are the mean times (in hundredths of a second) for all sentences. The result are presented in the tables below with the sentence numbers referring to the previous examples. Since there are so many examples, I will discuss them in their logical groupings. As in the first experiment, we chose the .01 level of significance for this experiment.

In the column on the right is the number of rules that ROBIE ran to analyse the same sentence. In Section 6.1, we will see that the relative parsing times of a model of the HSPM should match the relative times of human subjects. These figures are given to assist the reader in making this comparison. The phrase "failed at X", means the parser "blocked" or garden pathed at that point. Only the relative ranking of this measure should be compared and no claims of further comment will be made on this measure.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[136]</td>
<td>214</td>
<td>11</td>
</tr>
<tr>
<td>[137]</td>
<td>229</td>
<td>11</td>
</tr>
<tr>
<td>[138]</td>
<td>327</td>
<td>16</td>
</tr>
</tbody>
</table>

This was the first set of global ambiguity examples and will be discussed in Chapter 9. It was predicted, for reasons which will be explained, that there would be no significant difference between the first sentence and each of the other sentences in the triplet. Sentence [138] was semantically odd to many subjects, indicating the preference for the use of "can" as an auxiliary verb. In fact, several of the subjects chuckled when they read it. There was no significant difference between [136] and [137]. However, [138] was significantly different from [137] and from [136] at the p<.05 level. This level of significance was not accepted for this experiment, but further experimentation may produce a different result. Since the times
for all sentences were statistically the same, a theory that automatic backtracking is used to find all readings of an ambiguous sentence should be rejected, since backtracking should require an increase in reading time. The experiment did not test which reading or readings the subject perceived.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[139]</td>
<td>248</td>
<td>18</td>
</tr>
<tr>
<td>[140]</td>
<td>298</td>
<td>18</td>
</tr>
<tr>
<td>[141]</td>
<td>262</td>
<td>18</td>
</tr>
</tbody>
</table>

There was no significant difference in reaction times among the three sentences.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[142]</td>
<td>255</td>
<td>17</td>
</tr>
<tr>
<td>[143]</td>
<td>272</td>
<td>17</td>
</tr>
<tr>
<td>[144]</td>
<td>268</td>
<td>17</td>
</tr>
</tbody>
</table>

Again, there was no significant difference in reaction times among the three sentences, nor as before, was any difference predicted.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[145]</td>
<td>304</td>
<td>12</td>
</tr>
<tr>
<td>[146]</td>
<td>323</td>
<td>11</td>
</tr>
<tr>
<td>[147]</td>
<td>212</td>
<td>11</td>
</tr>
</tbody>
</table>

Sentence [145] had one more word that the other two sentences, making a comparison difficult. Even so, there was no significant difference between [145] and [146]. Sentence [147] was significantly faster at the 1% level than either of the other two.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[148]</td>
<td>938</td>
<td>failed at &quot;sank&quot;</td>
</tr>
<tr>
<td>[149]</td>
<td>371</td>
<td>25</td>
</tr>
<tr>
<td>[150]</td>
<td>1013</td>
<td>failed at &quot;fell&quot;</td>
</tr>
<tr>
<td>[151]</td>
<td>305</td>
<td>25</td>
</tr>
<tr>
<td>[152]</td>
<td>923</td>
<td>failed at &quot;grows&quot;</td>
</tr>
<tr>
<td>[153]</td>
<td>404</td>
<td>22</td>
</tr>
</tbody>
</table>
These are the garden path sentences which were discussed in Chapters 2-4. It was predicted that the first sentence of each pair would cause a garden path, but not the second. The first of each pair was significantly different from the second at the 1% level. These sentences proved to be extremely difficult for the subjects. After looking at the garden path examples for several seconds, many subjects asked what to do if they could not understand the sentence! The experiment did not test whether the subject had really analysed the sentence properly. A few subjects admitted afterwards that they couldn't understand the sentence, so went on to the next sentence. These show that the first of each pair causes a garden path, while the second sentence of each pair does not.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[154]</td>
<td>310</td>
<td>36</td>
</tr>
<tr>
<td>[155]</td>
<td>450</td>
<td>36</td>
</tr>
<tr>
<td>[156]</td>
<td>365</td>
<td>36</td>
</tr>
</tbody>
</table>

These examples were to test the difficulties with "that" as a relative pronoun versus "that" as a complementiser. In Section 5.2.10, we will see that this is one of the major ambiguities in which "that" can be involved. The results here will be important to our discussion in that section. The ambiguous version ([154]) was significantly faster at the p<.05 level than the two relative clause readings. The level of significance which can be used for a particular experiment is dependent upon the total number of examples in that experiment [Snedecor and Cochran 1967]. Because of the large number of examples in this experiment, we cannot accept this level of significance. This may indicate that the globally ambiguous example is, in fact, easier to process than the relative clause example. The subjects questioned afterwards said that they did not notice the possible relative clause reading for the first sentence. The experiment was not designed to test which reading the subject obtained, but I
feel this supports the theory that the subject preferred the embedded sentence reading.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>157</td>
<td>369</td>
<td>34</td>
</tr>
<tr>
<td>158</td>
<td>268</td>
<td>24</td>
</tr>
<tr>
<td>159</td>
<td>364</td>
<td>33</td>
</tr>
<tr>
<td>160</td>
<td>279</td>
<td>23</td>
</tr>
</tbody>
</table>

In Section 5.2.11 and Section 9.2 we investigate ambiguities involving the word 'have'. In these sections, it is important whether certain examples involve an increase in reading time. It was predicted that these examples would show that there was no time difference between reading "have" as an imperative versus "have" as a yes-no-question. There was no significant difference between [157] and [158]. There was a slight order effect in that the first sentence of each pair presented took longer than the second sentence. Sentences [159] and [160] showed the processing difference of imperatives versus yes-no-questions, (significantly different at the p<.05 level). In both sets, the Imperative was one second slower than the Yes-No-Question. This suggests that the 'have' examples were not different from the normal examples.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>161</td>
<td>832</td>
<td>40</td>
</tr>
<tr>
<td>162</td>
<td>930</td>
<td>40</td>
</tr>
</tbody>
</table>

These sentences were included to provide data for the discussion in Section 9.2. There was a very definite order effect with these examples. The times given are those obtained when the example occurred first in the order. However, when the example occurred second, the times were 300 and 490 respectively. The difference between the sentences was not significant. It seems that when the subject saw the first sentence of the pair, irrespective of which sentence it was, they had a considerable amount of trouble. One can see that the time was almost exactly the same at the
first exposure. These times showed that the subject had problems with these sentences the first time, but then learned most of the example, which assisted the understanding of the similar sentence the next time. It is not possible to tell whether the problems were due to the reduced relative, or 'have'.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[163]</td>
<td>1263</td>
<td>failed at &quot;surprised&quot;</td>
</tr>
<tr>
<td>[164]</td>
<td>367</td>
<td>34</td>
</tr>
<tr>
<td>[165]</td>
<td>393</td>
<td>34</td>
</tr>
<tr>
<td>[166]</td>
<td>665</td>
<td>42</td>
</tr>
<tr>
<td>[167]</td>
<td>363</td>
<td>34</td>
</tr>
</tbody>
</table>

These examples also relate to 'that' ambiguities as discussed in Section 5.2.10. Marcus [Marcus 1980] suggested that [167] was a garden path. The reader should use "that" as a determiner, rather than start the initial embedded sentence. Sentences [163] and [164] were significantly different at the .01 level. [166] also had the sentential subject problem and the longer time may be because of the structural processing difficulty in this type of sentence. In the parser's analysis, the phrase "birds ate everything in my garden" is first analysed as a sentence, and then made into a subject NP. This analysis requires extra rules for the parser, and may be part of the reason people also required a longer time to read this sentence. It was significantly different from [165] and [167] at the .01 level. [164] and [167] were predicted to be identical in processing time. [165] should also have the same time and is not significantly different from [167].

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[168]</td>
<td>432</td>
<td>31</td>
</tr>
<tr>
<td>[169]</td>
<td>479</td>
<td>23</td>
</tr>
<tr>
<td>[170]</td>
<td>404</td>
<td>23</td>
</tr>
</tbody>
</table>

These examples illustrate potential difficulties with the word "what", as discussed in Section 5.2.7. These examples were to test whether the first sentence caused a garden path. These times showed that the first example was not a garden
ROBIE degraded the initial sentence, 'What little fish eat' into the subject of the main verb, 'is'. This analysis predicts more rules will run and one can see that ROBIE's results reflect this. There was no significant difference between the three.

The third sentence occurred immediately after a garden path example in the second order. This increased its time in that order. In the first order, the second sentence occurred after a garden path, and this seems to have increased its time as well.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[171]</td>
<td>358</td>
<td>37</td>
</tr>
<tr>
<td>[172]</td>
<td>372</td>
<td>38</td>
</tr>
<tr>
<td>[173]</td>
<td>427</td>
<td>38</td>
</tr>
</tbody>
</table>

These sentences are globally ambiguous and they are generally considered to have several potential readings. The purpose of these examples was to see whether this global ambiguity caused an increase in processing time for the sentences. Although these results are interesting, they will not be discussed elsewhere in this thesis. There was no significant difference between these examples. One can see that the times overlap with the ones below (main verb 'know').

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Rules Run by ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[174]</td>
<td>346</td>
<td>37</td>
</tr>
<tr>
<td>[175]</td>
<td>443</td>
<td>38</td>
</tr>
<tr>
<td>[176]</td>
<td>439</td>
<td>38</td>
</tr>
</tbody>
</table>

Sentence [174] was significantly faster than the other two, p<.01. This result was unexpected and may be due to the different subcategorisation of 'know'. These examples were included out of curiosity only and will not be discussed.

ROBIE's simple use of semantic markers for PP attachment can not fully model how people presumably decide upon PP attachments, and no attempt was made to accurately model this. As a result ROBIE may have produced a different analysis than the subjects.
This and the following pair illustrate the difficulties of noun/modal ambiguities. In Section 5.2.8, we shall see why these are predicted to be garden paths. It was predicted that the second example would cause a garden path because of the fragment "will be". The reaction time indicates that the subjects did garden path on this example, as did ROBIE. [177] and [178] were significantly different at the .01 level.

These are essentially similar to the above pair, but this time, the problem was "can be". Again, the garden path effect is evident, though not as strong. Sentences [179] and [180] were significantly different at the p< .05 level.

4.8.6 Summary

This experiment has provided non-subjective reading times for many sentences. Several of these results will be important in the following chapters:

a) Garden path sentences do require a much longer reading time relative to their non-garden path partner. For most examples the reading time for the garden path sentence was much greater.

b) The "have" examples did not cause a garden path effect or show any significant difference in reading time.

c) The noun/modal examples exhibited the garden path effect.
No specific conclusions will be made here, rather these times are provided for discussion later in the text.

It should be noted that in the case of a globally ambiguous sentence, the parser returns a parse for only one reading. The reading which is selected can be considered to be arbitrary but is influenced by the principles of Minimal Attachment and Right Association as in Section 8.5.
The Role of Syntactic Context in Handling Part of Speech Ambiguity

In the previous chapters we have looked at resolving cases of lexical ambiguity which can lead to a garden path. This has suggested a model of the syntactic processor with limited lookahead. In this chapter we will look at cases of lexical ambiguity which do not lead to a garden path and investigate whether our model, as developed so far, can handle these cases. In doing this, we will see which examples of local ambiguity can be resolved on a purely syntactic basis and look at resolving part of speech ambiguity in greater detail.

Although many high level constituents can be 'moved' in English, the lower level structure of some constituents is relatively fixed. For example, after a determiner, one expects a noun rather than a verb. In this chapter we also wish to ask, 'How might this low level fixed order assist in the resolution of ambiguity?' We will not give a definite answer to this question, but will see that it is extremely useful in the resolution of ambiguity.

The examples of ambiguity shown in this chapter seem to cause no apparent problems to a person reading them. That is, all of these examples read easily and certainly do not exhibit the garden path effect. If ROBIE is to be psychologically plausible, then it is desirable that it handle these examples in such a way as to explain why people have no apparent difficulty with most sentences, despite the inherent ambiguity in them.

Occam's Razor tells us that, if there are two solutions to the same problem, then the simpler solution is preferred. If it is possible to handle all the examples of local ambiguity presented here, with no additional mechanism, device or feature than is needed for ordinary sentence parsing, then our goal above can be considered met. One possible explanation for people not noticing local ambiguities may be that there
is no special mechanism needed for them, so that nothing differing from normal parsing is necessary.

Conversely, if it is necessary to add special mechanisms and routines to the parser just to handle these examples of ambiguity, then this will not explain how people can understand these examples so well and can be considered a weakness in the model.

To say part of speech ambiguity can be handled deterministically, but with the use of special mechanisms would be no surprise and not very important. To say one can handle part of speech ambiguity deterministically with no special mechanisms is a more significant claim. In this chapter it is indeed suggested that many cases of part of speech ambiguity can be handled by the parser with no special mechanisms.

It should be noted that a non-deterministic parser does not need to tackle the problem of local part of speech ambiguity. If it should make an error, then it can backtrack and correct it. However, to handle ambiguity deterministically, we must never make an error. We will also see that many cases of ambiguity can be resolved using standard techniques which have been applied to non-deterministic parsers.

5.1 Syntactic Context

5.1.1 Word Data Structures

As a first approach to handling ambiguity, I asked, "If we construct a compound lexical entry for each word composed of the features of each part of speech the word can have and make no alterations to the grammar, how wide a coverage of examples will we get?"

This approach was used by [Winograd 1972] and was found to be very effective for the following reasons. If words have all the possible relevant features, then the tests for all possible parts of speech which a word can be used as will
succeed. In this way, all applicable rules will match. It may be that often only one rule will match, or that the first rule tried is the correct rule. The question is, how often will the rule which matches, be the correct rule? In Section 5.1.3, I will explain the answer to this question. Firstly though, it is necessary to explain how the word definitions were altered.

All words in ROBIE are defined in the syntactic dictionaries. Each word has a compound lexical entry incorporating all the features for all the possible parts of speech which the word could be. This is exactly as was done by [Winograd 1972]. For example, 'block' is defined as a noun and a verb, 'can' is defined as a noun, auxiliary verb, and verb and 'hit' is defined as a noun and a verb. The features for each of these parts of speech are kept in the dictionary and, when the word is looked up, they are returned as a single ordered list of features. These features are sub-grouped according to the part of speech they are associated with. Hence, when the word 'block' is looked up, the result returned is both the noun and the verb definition. In this way, all possibilities are returned. Below is an example of a dictionary entry:

```
word: block
  noun features: noun,ns,n3p,ngstart
  verb features: verb,v-3s,tenseless
```

In the above example, the word 'block' has the features noun, noun singular, noun third person, verb, tenseless verb and is marked that it can start a noun phrase and as a verb it agrees with any noun which is not 3rd person and singular. This multiple meaning is then carried during the parse, until the word is disambiguated.

In the English language, most words can be several parts of speech. This fact must be reflected in a parser of English and we do this with the multiple meanings above. When the parser has enough information to decide which is the correct one, it ignores (removes) the other possibilities. In this way, we have not built structure which is later thrown away, rather we have reflected an inherent parallelism of language. In Chapter 6 we will examine the psychological basis for this
approach.

5.1.2 Morphology

The first part of the disambiguation process takes place in the morphology. When ROBIE identifies a word which has a morphological ending, the morphology must adjust the features of the word. For example, when "blocked" is identified, the feature "ed" must be added to the list of features for "block". At the same time, a portion of the disambiguation takes place. If "block" is defined as both a noun and a verb, then "blocked" is not a noun. The morphology causes some features to be added, such as "ed, past" and some features to be removed such as "tenseless". As features which are no longer applicable are removed, so also are parts of speech and their associated features which are no longer applicable. For "blocked", the features "noun, ns, n3p" will be removed and the features "adjective, ed, past" will be added.

Similarly for the word "blocking". This cannot be a noun, so only the verb and adjective definitions are carried forward. When the morphology causes adjustments to the features for "block", by adding the features "en" and "adjective", it must also alter the tense and number. At this time, other features that are no longer appropriate such as "noun" will be removed.

The morphology will identify words such as adverbs, adjectives and verbs in a similar way. The morphology which is used is very similar to that of [Winograd 1972], [Dewar, Bratley and Thorne 1969] and the part of speech additions and deletions are taken from [Marcus 1980].

5.1.3 Disambiguation

The dictionary and morphology now give all the possible features for each word. We will see in this chapter that having the dictionary return the definitions in this way greatly assists our handling of ambiguity. The psychological motivation
for this is discussed in Chapter 6. The next step is to add a method for disambiguating the words. This is done by pattern matching of the grammar rules as follows.

Each rule matches the features of one or two buffers. If the word 'block' is in the first buffer, then a pattern [noun] or a pattern [verb] will match. These patterns do not relate to the other possible definitions of a word. If a rule pattern has matched on the feature 'noun' in the first buffer, then ROBIE assumes that this word is a noun. It would then be appropriate to disambiguate the word as a noun. This is exactly as in [Winograd 1972].

For every grammar rule which disambiguates a word to a certain part of speech, that part of speech is the same as the part of speech which the pattern of that rule assumed the word was. This means that the technique of having the patterns disambiguate the words is the only one necessary for ROBIE. (Although this is how the parser's disambiguation works in principle, in the actual implementation, the grammar function ATTACH performs the disambiguation. This will be explained in detail in Chapter 8.)

In a non-deterministic parser it is not essential to find the correct rule first. If the parser runs an incorrect rule, the parser may backtrack and change the category assignment. But in a deterministic parser, there will never be any backtracking and this solution cannot be used.

Since ROBIE does not backtrack, there is little danger of the pattern matching disambiguation making an error. Once a rule runs assuming a buffer contains a certain part of speech, it must be used as such in the parser. The general disambiguation scheme is; if a full pattern matches a word as a certain part of speech, then it is disambiguated as that part of speech.

The compound lexical entries and pattern-matching disambiguation alone will handle many examples of ambiguity. In the rest of this chapter we will see just what this can do for us.
Given the above mechanisms, multiple definition and disambiguation by the pattern matching, let us see how a few examples are handled. Consider:

[181] The falling block needs painting.

We will look only at the words "falling" and "block" in this example. The word "falling" is defined as a verb and an adjective in the dictionary and "block" is defined as a noun and a verb.

Whilst parsing this example, after the word "the" has initiated an NP and been attached to it as a determiner, the rules to parse adjectives are activated. The rule ADJECTIVE has the pattern: [adj], and matches the word "falling". "Falling" is then attached and disambiguated as an adjective. Recognition of "falling" as a verb does not occur. As there are no more adjectives, ROBIE will activate the rules to parse the headnoun. The rule NOUN with the pattern [noun] will match on the word "block" and it will be attached as a noun. Hence "block" will also be disambiguated without the verb use being considered by ROBIE.

Other ambiguities inside the noun phrase will be handled in a similar way. This approach alone will cover the situation of singular head nouns, verb/adjective ambiguity and most other pre-nominal ambiguities. This works because the noun phrase has a very strict word order. When an ambiguous word is found, only one of its meanings will be appropriate to the word order of the noun phrase at that point. This approach can be thought of as an extension of the basic approach of the Harvard Predictive Analyzer [Kuno 1965].

This strategy will also often disambiguate main verbs. For example in:-

[182] Tom hit Mary.
[183] Tom will hit Mary.
[184] The will gave the money to Mary.

In [182], "hit" is the main verb. In the dictionary, "hit" is also defined as a noun, (as in card playing). The parser will attach "Tom" as the subject of the sentence.
and then activate the rules for the main verb. Since 'hit' has the feature "verb", it will match that rule and be attached and disambiguated as a verb. Again other possible parts of speech are not considered.

The word "will" could be a noun or a modal as sentences [183] and [184] demonstrate. In [183], "will" cannot be part of the headnoun with 'Tom', so the NP will be finished as above. The rules for the auxiliary will then be activated and the word "will" then matches the pattern [modal] and is attached to the AUX.

In [184], the word "will" is used as a noun. Since it follows the determiner, then the rules for nouns will be activated. The word "will" then matches the pattern [noun] and attaches to the NP as a noun.

The same approach will also disambiguate 'stop' and 'run' in the following sentence. Since "stop" is sentence initial and can be a tenseless verb, the rule IMPERATIVE will match and it will be disambiguated as a verb. The word "run", which can be a noun or a verb will be handled as "will" in [184].

[185] Stop the run.

5.1.5 The Word TO

Now let us consider a more difficult example, the word "to". "To" is defined as a preposition and an auxiliary verb in ROBIE, as illustrated by these sentences:

[186] I want to kiss you.
[187] I will go to the show with you.

In [186], "to" is the infinitive auxiliary, whilst in [187] "to" is a preposition. This analysis is based on that of [Marcus 1980, p. 118]. Our two buffer look-ahead is sufficient to disambiguate these examples.

The buffer patterns for the above sentences are:
[to][tenseless] -> embedded VP
[to][ngstart] -> PP
By looking at the following word, "to" can be disambiguated. In [187], the word "the" cannot be a tenseless verb, so the first pattern does not match. In [186], the second buffer does not have the feature "ngstart", so the rule doesn't match.

However, the above patterns will accept ungrammatical sentences. To reject ungrammatical sentences, we can use verb subcategorisation as a supplement to the above rules. One cannot say:

[188] *I want to the school with you.
[189] *I will hit to wash you.

In English, only certain verbs can take infinitive complements. "To" can only be used as a auxiliary verb starting a VP when the verb can take an infinitive complement. Hence, by activating the rules to handle the VP usage only when the infinitive is allowed, the problem is partly reduced. Also by classifying the verb for PPs with the preposition "to", the problem is simplified. This is merely taking advantage of subcategorisation in verb phrases. To allow "to" to start a VP, when it is not allowed, would be ungrammatical. Taking advantage of this fact greatly reduces, but does not eliminate, the possible conflict. In ROBIE, the subcategorisation of verbs for infinitive complements has been implemented, but the verbs are not fully marked as to the type of PPs they will accept.

We have seen what to do if the verb will only accept a toPP or a VP. The final difficult situation arises whenever the following three conditions are true: 1) the verb will accept a toPP and a toVP 2) the item in the second buffer has the features "tenseless" and "ngstart" and 3) the toPP is a required modifier of the verb. It seems that there are very few verbs which have this subcategorisation [Gazdar, personal communications] and the distribution of words with toVPs and toPPs seems different, so this problem rarely arises. When it does, the principle of Right Association and Minimal Attachment apply as discussed in Chapter 6.

A free text analysis done on a cover story in TIME magazine [TIME 1978] resulted in 55 occurrences of the word "to". The two rules mentioned above in con-
junction with verb subcategorisation gave the correct interpretation of all of these. These rules were also checked on the MECHO corpus (Appendix B) and the ASHOK corpus [Martin, Church and Patil 1981]. There were no violations to these rules in either of these. For a full explanation, see Appendix C.

5.1.6 Adjective/ Noun and Noun/ Noun Ambiguity

Some readers may wonder how adjective/noun ambiguity and noun/noun ambiguity are handled in ROBIE. As stated in the introduction, this research has not investigated semantic problems within a single part of speech. Therefore this research has not investigated noun/noun ambiguity. ROBIE does handle adjective/noun problems for the MECHO world and this approach is explained below.

Theoretically, a nounphrase can have an infinite number of adjectives. In order to implement this in the parser, the packet PARSE_ADJ is activated after the determiner for the NP has been attached, or at the start of the NP if there is no determiner. The packet has essentially two rules. The first rule will attach an adjective in the first buffer to the partial NP which is the current active node. This rule may then apply if there is more than one adjective. When there are no more adjectives in the first buffer, the second rule matches. This rule is of low priority and has the pattern [t]. It will deactivate the packet PARSE_ADJ and activate the packet PARSE_NP. Adjective/Noun ambiguity can be characterised as: should the next word be attached as an adjective, or should the packet be deactivated?

Adjective/noun ambiguity is handled in a simple minded way. If the word following the ambiguous adjective/noun word can be a noun, then the ambiguous word is used as an adjective. In other words, all conflicts are resolved in favour of the adjective usage. This problem arises in these examples:

[190] The plane is inclined at an angle of 30 degrees above the \textit{horizontal}.
[191] A block rests on a smooth \textit{horizontal} table.
In [190], "horizontal" is a noun, while in [191], it is an adjective. The above algorithm handles these cases.

For sentences such as [192], with noun/noun ambiguity, the syntax in ROBIE is a flat structure.

[192] The soup pot cover handle is hot.

In ROBIE, the semantic interpretation receives an ordered list of headnouns. This ambiguity is then left to be resolved by the semantic interpretation component. The semantic representation looks like this:

headnouns for NP1: soup, pot, cover, handle

The semantic inferencing can then do whatever is needed for the application of the system. Since the non-syntactic processor is extracting the meaning of the sentence in parallel with the syntactic analysis for normal sentences (see Section 4.1), the semantic interpretation of the headnouns can be performed as the list of headnouns is built. In the current parser, the MECO semantics does this as a second stage. This mechanism is the same as the markers used to perform the evaluation needed by the Semantic Checking Hypothesis.

ROBIE makes no attempt to resolve all cases of adjective/noun ambiguity and often treats an ambiguous word as part of the complex headnoun. I feel that a better understanding of what people seem to do with adjective/noun ambiguity needs to be gained before it can be dealt with in a psychologically plausible way. For this reason the approach outlined above is used, although it is not intended to be psychologically plausible.

5.1.7 Why Does This Work?

In this section we have seen many examples of ambiguity being resolved. To handle these examples, we merely constructed a compound lexical entry for each word, composed of the features of each part of speech the word could be and allow
the pattern matching to perform the disambiguation. This technique has been used by Winograd [1972]. Why does this work so well? At the start of this chapter, we also asked, 'How does the fixed order of the structure of some low level constituents assist with the resolution of ambiguity?'

If we look at these questions from a processing viewpoint, we can see how their answers are related. English has a fairly strict structural order for all the examples presented here. Because of this, in each example we have seen, the use of the word as a different part of speech would be ungrammatical. Although these techniques have been used for non-deterministic parsers, their effectiveness has not been investigated for a deterministic parser. Perhaps this fixed word order has evolved to reduce potential ambiguities and make processing easier, with morphology as an alternative. It would be interesting to see how this relates to the frequently remarked fact that the morphological complexity and fixedness of word order at the clause level of a language seem to be inversely related. The examples in this chapter suggest that this may be true at not only the clause level, but also in low level constituents.

Most ambiguities are not recognised by people because only one of the potential ambiguities is grammatical. In many situations, when fixed constituent structure is taken into account, other uses of an ambiguous word are not possible and probably not even recognised. Since fixed constituent structure rules out most alternatives, we have been able to handle the examples in this chapter without any special mechanisms. In the introduction to this chapter, it was stated that a clean and simple method of handling ambiguity was desired. I feel that this goal has been met for these examples.

5.2 The Role of Agreement in Handling Ambiguity
Using the simple techniques presented in the last sections, we can handle many cases of part of speech ambiguity, but there are many examples we cannot resolve. For example, the second of each pair of sentences below would be disambiguated incorrectly.

[193] I know that boy is bad.
[194] I know that boys are bad.
[195] What boy did it?
[196] What boys do is not my business.
[197] The trash can be smelly.
[198] The trash can was smelly.

Many people wonder what role person/number codes and the relatively rigid constituent structure in the verb group play in English. Linguists can describe these, but cannot produce an adequate explanation of why they are there. In this section we will look at these mysterious items from a processing viewpoint. We will explore their role by attempting to answer the question, "What use is the fixed structure of the verb group and person/number codes." First, however let us now look at how Marcus's parser handled a few more examples of ambiguity.

5.2.1 Marcus's Diagnostics

Marcus [1980] did handle some part of speech ambiguities. The words 'to', "for", "what", "which", "that", "a", and "have" could all be used as several parts of speech. For each of these words he also used a 'Diagnostic' rule. These Diagnostic rules matched when the word they were to diagnose arrived in the first buffer position and the appropriate packets were active. Each diagnostic would examine the features of the three buffers and the contents of the Active Node Stack. Once the diagnostic decided which part of speech the word was being used as, it either added the appropriate features, or explicitly ran a grammar rule. Marcus did not give each word a compound lexical entry as we have done here.

Most of the grammar rules in his parser were simple and elegant, but the diagnostics tended to be very complex and contained many conditionals. In some
cases they also seemed rather ad-hoc and did not meet the goal of a simple, elegant method of handling ambiguity.

For example the THAT-DIAGNOSTIC :- (ignore the details of the grammar code)

```plaintext
[that][np] -> in the Packet CFOOL
'If there is no determiner of second
  and there is not a qp of second
  and the nbar of 2nd is none of massn,npl
and 2nd is not-modifiable
then attach as det
else if c is nbar then label 1st pronoun, relative pronoun
else label 1st complementiser.''
[from Marcus 1980, p. 291]
```

Notice that if the word "that" were to be used as a determiner, then it would be attached after the NP was built! This is his primary rule for disambiguating the word "that". Marcus's parser also had three other rules to handle different cases.

The WHICH-DIAGNOSTIC was more simple, but still a special case:

```plaintext
[which] -> in the packet CPOOL
'If the NP above c is not modified then
label 1st pronoun, relative pronoun
else label 1st quant, ngstart,ns,wh,npl''
```

It seems that these rules did not "elegantly capture generalisations" as did the rest of his parser. I consider these rules undesirable and feel that they should be corrected to comply with my criteria for simple and elegant techniques in resolving ambiguity. I wanted a method which used no special mechanism, or routine, other than that needed to parse grammatical sentences. These diagnostics are certainly special mechanisms and do not meet this goal. Can we cover the same examples in a more simple and principled way?

In this section, we will look at each of these diagnostics in turn and show how they have been replaced in the newer model. We will also look at a few other examples of ambiguity which Marcus did not handle, but are related to our discussion here.
5.2.2 Handling the Word TO

Marcus's diagnostic handling of "to" can be replaced by the method outlined in Section 5.1.5. This method was motivated to handle grammatical sentences and meets our criterion for a simple approach.

5.2.3 Handling the Word FOR

The problem with the word "for" is in deciding whether it is a preposition or a complementiser, as the standard Chomsky analysis would have it, in the following examples:

[201] I preferred for John to go.
[202] I preferred for John to hit Mary.
[203] I wanted a flower for Mary.
[204] I want the paper for Wednesday.

In [199] and [200], "for" is a complementiser with "to", whilst in [203] and [204], it is used as a preposition.

A conflict could arise in the use of this word only when the verb is subcategorised for both a forPP (a PP with the preposition "for") and a For-To embedded sentence, i.e., both the following sequences are possible:

\[
\begin{align*}
V & \text{ forPP toVP} \\
V & \text{ forPP toPP}
\end{align*}
\]

There do not seem to be any verbs in this class. In 'British English', sentences of the form [199,200] are unacceptable. This case would be the most difficult to handle for ROBIE of the cases we have discussed. It is interesting that although it is allowed in some dialects of 'American English', it is not allowed in 'British English'. The only time when a conflict seems to arise is when the verb appears in the context:

\[
\begin{align*}
V & \text{ NP forPP} \\
V & \text{ NP forPP toVP}
\end{align*}
\]
for example:

\[ [205] \text{I want a horse for John.} \]
\[ [206] \text{I want a horse for John to ride.} \]

To handle this case, Marcus built the NP (John) with an Attention Shift before he combined the NP with 'for' into a PP. Before he ran the PP rule, his parser tried to match the following rule:

\[ [\text{for}] [\text{np}] [\text{to}] \]

This rule could then clearly disambiguate the word 'for'. The solution worked well with the use of the Attention Shift and a three buffer lookahead. Neither are allowed in ROBIE. In fact, given the Chomsky analysis that these are complementisers, there is no way to distinguish between the use of 'for' as a complementiser or as a preposition unless we look past the NP to see if the word 'to' is present.

It seems that there is no way to handle this case without three buffers of lookahead and the Attention Shift, but this linguistic analysis is not the only one possible. Several linguists have suggested that 'for' is actually a preposition in both situations, e.g. Gazdar [Gazdar, Pullum, and Sag, 1980c]. He treats the forPP in [199] as a PP and part of the VP for 'want', not as an embedded sentence. By doing this, 'for' is not used as a complementiser and the ambiguity disappears. Other linguists e.g. Chomsky [Chomsky 1965], suggest that the PP is part of the verb phrase.

Let us assume that 'for' is always a preposition and see how these examples may be handled. (As we shall see, this is a description of what ROBIE currently does.) It is very easy to treat the forPP as part of the verb phrase, so we will look at how to build the embedded sentence analysis.

The parser will make the word 'for' a preposition and start the PP as a normal PP. After the PP has been fully built and dropped back into the first buffer from the Active Node Stack, the state of the buffers will be:
At this point the rule deciding the attachment of the PP will be active and the NP ("a horse") will be the bottom item in the Active Node Stack. Simultaneously, the rule that would have run to detect the For-To complementiser above, would be active. We need only to alter this rule so that it has the pattern:

\[(\text{pp \\& for})[\text{to}]\]

The embedded sentence will now be started and the NP of the PP will be the subject. The semantic interpretation of the embedded sentence uses the "for" in the subject as it sees fit. Analysis is then possible without the need for an Attention shift, or the three buffers.

This is one of the cases in Section 2.5.2 which Marcus required three buffers to handle. If we adopt the analysis of Gazdar above, we can handle this situation with only two buffers, whereas Chomsky's analysis is incompatible with ROBIE.

It is interesting that both analyses are possible for ROBIE, but the Chomsky analysis requires two additional items which the Gazdar analysis does not need, i.e., three buffers and the Attention Shift. For ROBIE, we have adopted Gazdar's analysis of the word "for". Following this approach, "for" is always a preposition, making the resulting grammar more simple, and the need for Marcus's For-Diagnostic disappears.

5.2.4 Ungrammatical Sentences

Before we proceed, let us look at an assumption Marcus made in his parser, that it would be given only grammatical sentences. This assumption makes life easy for someone writing a grammar, since there is no need to worry about grammatical checking. Hence he did not cater for ungrammatical sentences and the original parser accepted such examples as:

\[
\begin{align*}
&[207] \text{ *a blocks are red. } \\
&[208] \text{ *the boy hit the girl the boy the girl. } \\
&[209] \text{ *are the boy run? }
\end{align*}
\]
This simplification causes no problems in most sentences, but can lead to trouble in more difficult examples. If the parser's grammar is loosely formulated because it assumes it will be given grammatical examples only, then ungrammatical sentences may be accepted. If the syntactic analysis accepts ungrammatical sentences as grammatical, then it is making an error. In the next sections we will look at the consequences of this assumption as well as those of rejecting ungrammatical sentences.

5.2.5 Subject/Verb Agreement

We know that the verb group has a complicated but relatively fixed constituent structure. Although verbals have many forms, they must be mixed in a certain rigid order. We also know that the first finite verbal element must agree with the subject in person and number. That is, one cannot say:

\[210\] *The boy are run.
\[211\] *The boy will had been run.
\[212\] *The boys had are red.

etc.

Whilst Marcus's parser enforced these observations to some extent, he did not follow it throughout his parser. We want to enforce this agreement throughout ROBIE. Checking the finite or main verb, to be sure that it agrees in number with the subject, will lead to the rejection of the above examples. This was done by adding the agreement requirement into the pattern for each relevant rule as will be explained later.

Buffers 1 and 2 must agree before a rule relating the subject and verb or two verbs can match. This check looks at the number code of the NP and the person/number code of the verb and checks whether they agree. The routine for subject/verb agreement is very general and is used by all the subject/verb rules. The routine can only check the grammatical features of the buffers and could be done by expanding the buffer feature patterns. See Section 8.6 for the full details of this.

- 105
5.2.6 Plural Head Nouns

We saw in Chapter 3 that the case of a word which can be both a noun, plural and a verb, singular (noun/verb/plural word) after at least one singular head-noun can lead to a garden path by the end of the sentence. Let us go back and look at a simpler case of this. We saw only one subcase of all the logical possibilities for combinations of words which can be both a noun and a verb. We will now look at these possibilities and see that these cases can be disambiguated by simple rules using subject/verb agreement. The following examples illustrate all the possibilities:

[213] The soup pot cover handle screw is red.
[214] The soup pot cover handles screw tightly.
[215] *The soup pot cover handles screws tightly.
[216] The soup pot cover handle screws tightly.
[217] The soup pot cover handle screws are red.

Each of the words "pot,cover,handle,screw" can be either a noun or a verb. The "end of constituent" problem is to find out which word is used as the verb and which words make up the complex headnoun. The possible distributions of plural and singular among two words gives us four cases. We will deal with each of these in turn.

Case 1: In [213] each noun is singular. For this case all ambiguous words must be nouns and part of the headnoun. Due to subject/verb agreement, a singular noun must match a 3rd person singular (v3s) verb, i.e, one without the letter "s". This case excludes that possibility since none of the words have an 's' at the end. Hence they must all be nouns.

Case 2: In [214] "handles" is a plural noun and each word before it must be a noun. When a singular noun/verb word follows 'handles', the word (screw) must be a verb and "handles" is the last of the headnouns. It is not possible to use 'handles' in this situation as a verb, and "screw" as a noun because of subject/verb agreement.
Case 3: Sentences with two consecutive plural nouns as in [215], where both words have noun/verb ambiguity are often ungrammatical. (Do not confuse plural "s" with possessive "'s"). The following is an example of two consecutive plural head nouns, the first of which is not noun/verb ambiguous ('sells').

[218] He sells plows to farmers.

This case is grammatical and not relevant to the discussion here, because 'sells' is not noun/verb ambiguous. For the case where both words are noun/verb ambiguous, whenever the first plural is the main verb, then the second plural will be a noun and is dealt with by ROBIE later. For example:

[219] He cuts trees on the weekends.

If the first plural is a noun, then the second one cannot be a verb, unless it is part of a different constituent. An example of this is: (Sentences beginning with "?'"are considered grammatical but unacceptable to most readers.)

[220] ?The soup pot machine handles screws easily.
[221] The soup pot machine handles screw easily.
[222] Which years do you have costs figures for?
[223] Do you have a count of the number of sales requests and the number of requests filled?
(The last two are from [Martin, Church and Patil 1981])

Because there is a non-plural headnoun followed by a plural headnoun, this situation leads to the next type of example.

Case 4: Sentences [216] and [217] both have the same word initial string until after 'screws', but in [216] 'screws' is a verb while in [217] 'screws' is part of the headnoun. In this situation, where the final word in a series is plural, each word before it must be a noun. The word itself can be either a noun or a verb, depending on what follows. These can be recognised as a pair of potential garden path sentences, as discussed in Chapter 3. Therefore, this is the case to which the Semantic Checking Hypothesis applies and the predictions of Chapter 3 apply.

Mass nouns are defined as ambiguous between singular and plural in ROBIE. This is because they appear in singular or plural noun positions. If the mass noun
occurs in a location where it could be plural, it is treated as such. Hence if there is an ambiguity of the type discussed here with a mass noun, it is treated as a plural noun and the semantic check we have described earlier applies.

Due to number and subject-verb agreement, these facts have a linguistic base. They rely on the fact that a final 's' marks a plural noun, but a singular verb. If the verb is v3s (verb agrees with a 3rd person, singular noun, as with the 's'), then the subject of the verb must be singular; or else the sentence is ungrammatical. This is why all the words before the v3s word must be nouns. If any of these words were used as a verb, then subject-verb agreement would be violated. This is why [215] is ungrammatical. If the verb is v-3s (agrees with any noun phrase except 3rd person, singular i.e., no 's'), then the subject cannot be singular. [213] has no plural subject and so cannot have a v-3s verb. In [214] "handles" provides a plural subject, so "screw" which is v-3s can agree.

In this section, we have looked at resolving a simple case of noun/verb ambiguity. In order to resolve this ambiguity, it was necessary merely to exploit agreement between the subject and verb in number and person.

5.2.7 Handling WHAT and WHICH

For both "what" and "which", the ambiguity lies between a relative pronoun and a determiner. The following examples show various uses of both words:

[224] Which boy wants a fish? det
[225] Which boys want fish? det
[226] The river which I was has fish rel. pron.
[227] What boy wants a fish? det
[228] What boys want is fish. rel. pron.

There is some debate about the part of speech to be assigned the word "which". Some linguists consider it to be a quantifier [Chomsky 1965], whilst others consider it to be a determiner [Akimajian and Heny 1975, Chapter 8]. We shall adopt
the determiner analysis, making the problems for "what" and "which" similar.

To determine the correct part of speech for these two words, Marcus used the following diagnostics:

```
[w] -> in the packet CPOOL
  'If the NP above C is not modified then
  label 1st pronoun, relative pronoun
  else label 1st quant, ngstart, ns, wh, npl.'

[w] -> in the packet NPOOL
  'If 2nd is ngstart and 2nd is not det
  then label 1st det, ns, npl, n3p, wh;
  activate parse_det
  else label 1st pronoun, relpron, wh.'
  [Marcus 1980, p.286]
```

These diagnostics would make the word in question a relative pronoun if it occurred after a headnoun, or a determiner if the word occurred at the start of a possible noun phrase.

If we follow the approach in the last section, and give each word a compound lexical entry composed of the determiner and relative pronoun features, we find that these words are always made determiners unless they occur immediately after a headnoun. In other words, the "which" examples are all parsed correctly, but [228] is parsed wrongly. This happens because the determiner rule will always try to match before the rule for WH questions can take effect. This simple step gives the correct analysis if the ambiguous word is to be a determiner, but will still err on [228].

The rule to parse a relative pronoun and start a relative clause is active only after the headnoun has been found. At this time, the rule for determiners is not active. Therefore, if the word "what" or "which" is present after a headnoun, the only rule which can match is the rule to use it as a relative pronoun and it will be used as a relative pronoun. We have resolved the simple case of "what" as a relative pronoun using only the simple techniques of the last section. For these sentences:

```
[229] What block is red?
[230] Which boy hit her?
[231] Which is the right one?
```
ROBIE produces the correct analysis, but still errs on [228]. This error is because "what" is being used as a relative pronouns, but it does not follow a head-noun. Without any additional changes to the parser, we get two things. Firstly, if the word occurs after the headnoun, then the NP-COMPLETE packet rules are active and it will be a relative pronoun. In fact, since relative clauses can occur only after the end of an NP, this correctly resolves the relative pronoun uses. If the word occurs at the start of an NP, then it will be made a determiner.

This approach has exactly the same effect and coverage as did Marcus's diagnostics, but we have not needed any special rules to implement it. It will now provide the correct interpretation for "which", but will make some errors for the word "what". Marcus's "what-diagnostic" will treat "what" as a determiner whenever the item in the second buffer could start a NP. This is usually correct, but "what" will be treated as a determiner in all of the following:

{232} What boys want is fish.
{233} What blocks the road?
{234} What climbs trees?
{235} What boys did you see?
{236} What blocks are in the road?
{237} What climbs did you do?

In this thesis, we are adopting the following analysis for WH clefts such as [232]. The initial WH word, "what" is a relative pronoun and attached as the WH-COMP of the subject S node. The subject is the phrase 'What boys want'. The main verb of the sentence is 'is' and the object 'fish'. The exact details are not important, only that the word "what" or "which" is a not determiner at the start of a WH cleft. The following tree illustrates this analysis:

```
S
 / \  
S | VP
 / | \ | |
WHCOMP | NP | VP | Verb | NP
 | | | | |
what boys want is fish.
```
In sentences [232-234], the word "what" is not used as a determiner. In the analysis we are using, it is a relative pronoun and is used as the WH-COMP for the S. In sentences [235-237], the word "what" is used as a determiner. Marcus admits that this diagnostic produces the incorrect result in this case [Marcus 1980, p. 286]. His diagnostic will make "what" a determiner in all of these examples, as will my analysis.

One can also see that each of the above pairs is a pair of potential garden path sentences. For each pair, the two buffers contain the same words. Hence our two buffer lookahead is not sufficient to choose the correct usage of the word "what". There is no way to make "what" a relative pronoun in the case where the headnoun is plural, but a determiner in the case where the headnoun is singular for all arbitrary sentences using only two or three buffers.

With regard to the Semantic Checking Hypothesis then, it is suggested that this decision is based on non-syntactic information. I believe that intonation is critical in these examples. Unfortunately there is insufficient experimental evidence to determine for certain whether this is true. Furthermore, if one adopted a linguistic analysis where "what" was used as a relative pronoun in all these examples, then the problem would not exist. Finally, the problem of "what" and "which" as sentence initials, with no noun in the second buffer seems to arise very rarely. I have found no examples of this problem in free text analysis. (See Appendix C).

The current parser (ROBIE) cannot obtain the extra information which is provided by intonation to help resolve this case. As a result it follows Marcus's diagnostic and makes "what" a determiner in each of the above cases. This is because "what" is defined as a determiner which can agree with either a singular noun or a plural noun, as it was in Marcus's parser.
5.2.8 Noun/Modal Ambiguity

We will now consider noun/modal ambiguity as demonstrated by "can" and "will". Both can be either a noun or a modal (i.e., could, should, would, can, will, might, etc.):

[238] The trash can was taken out.
[239] The trash can be taken out.
[240] The paper will was destroyed.
[241] The paper will be destroyed.

Each of these words is entered in the dictionary both as a noun and a modal. Due to agreement requirements, the modal/noun word can only be grammatically used as a modal if the word following it is a tenseless verb, i.e., the pattern:

[modal][tenseless] -> modal usage

applies. Handling noun/modal ambiguity can be quite easy, when the noun modal word appears in the first buffer one merely has to look at the contents of the second buffer to see if it contains a tenseless verb. This can be complicated though, if the auxiliary is inverted or the sentence is an imperative. The following examples show how this can arise:

[242] Let the paper will be read.
[243] Will the paper can be re-used?

In sentence [242] the fragment "Let the paper" implies that "will" can only be used as a noun, as the sentence already has one tensed verb. In the parser, the noun/modal word is first encountered inside the NP packets and the parser must decide whether to use the word as part of the headnoun or to leave it in the buffer to be used as a modal verb. These rules do not know whether a verb has been found previously. Hence, not all information from the sentence is used. If all the information is available at the time the noun/modal ambiguity is being resolved, these sentences would be unambiguous and people would have no trouble reading them.

Subjects were asked to read the above examples in the second experiment. The results showed convincingly that they are potential garden paths. Many naive
readers had considerably more difficulty with them than with their more straightforward counterparts. This was predicted for reasons that will be explained below.

This result seems surprising. If the subjects used all information available at the time the noun/modal word was encountered, then they should have had no trouble with these sentences. The fact that these are garden paths indicates that the readers did not use all the information available to them. Notice also that the ambiguity can be reformulated as: 'Do we have the end of a noun phrase, or a complex head noun'?

We have already seen a case where people do not seem to use all the information available to them. In Chapter 4 we saw several end of NP problems which could lead to a garden path. In each of these, we showed that the ambiguity was resolved on the basis of non-syntactic information, without regard to the following words in the sentence. In other words, we saw that the reader did not use all the information available. There is one crucial difference though. In the previous cases, they used non-syntactic information because the syntactic processor with its limited lookahead was sometimes unable to choose the correct alternative. In this case, the information necessary has already been absorbed by the parser.

This suggests that the choice of alternatives is made locally inside the NP parsing rules, without regard to information about the type of sentence being parsed. In other words, the two buffer pattern applies regardless of the rest of the sentence. This assumes that a noun/modal word followed by a tenseless verb is being used as a modal. Let us look at why this might be true in the parser.

When the parser starts to parse a NP, it creates a new NP node and pushes it to the bottom item of the Active Node Stack. This operation makes the NP node the Current Active Node and parsing of the old Current Active Node is suspended. If the parser is parsing an S node, for example at the start of the sentence, then work on this node will be suspended until the NP node has been completed and dropped into the buffer. (see Appendix A for an example.)
Remember also that the pattern matcher for the grammar rules is allowed only to inspect the grammatical features of the two buffers. This means that the parser is unable to examine the contents of the Active Node Stack and, hence, the information that a tensed verb has already been found is unavailable to the NP parsing rules. This then suggests that the ambiguity will be resolved on the basis of local information only.

The structure of the parser gives us a computational explanation for the difficulty of these sentences. From a processing viewpoint, the trouble with these examples is understandable, whilst from a descriptive viewpoint, there seems no a priori reason why these sentences should cause difficulty.

This ambiguity is an end of NP problem and the choice of alternatives is made on the basis of limited and local information. This suggests that non-syntactic information may be used to resolve the ambiguity. There is one further possibility. The semantic choice mechanism is attempting to find the end of a NP. So far it has asked the question, 'Can this item be part of the NP?' However, the end of NP problem can be reformulated as, 'Is it better to use this as part of the NP, or as the start of the verb group?' It is conceivable that the end of NP mechanism uses "will" as the start of the verb group in the majority of occurrences, hence leading to the apparent modal preference in these examples:

[244] The trash can hit the wall.
[245] The paper will hit the table.

The exact question asked by the syntactic processor of the non-syntactic processor can probably never be determined. These sentences have made several suggestions. Due to lack of data, it is not clear exactly what people do and this would seem to provide an interesting area for further investigation.
Another problem is the word "her", which can be used as a pronoun or as a possessive pronoun. Note that we can say:

[246] Tom kissed her.
[247] Tom kissed her sister.

Clearly in [246] "her" is a pronoun and in [247] "her" is a possessive determiner. When multiple part of speech definitions were added to ROBIE and the simple disambiguation method used, ROBIE always made "her" a possessive determiner.

This difficulty arose in Marcus's parser because the rule to start a NP was ordered before the rule to parse a pronoun. These rules were copied directly into ROBIE's grammar. Since the word "her" has both the features "ngstart" and "pronoun", it could match both rules. Unfortunately, as Marcus's rules were stated, it always matched the NP starting rule, and hence was never made a determiner. This indicates one problem that can arise in the writing of a parser grammar.

To handle possessive determiners, PARSIFAL and ROBIE have a rule with the pattern:

[poss_np]

This rule will match a possessive pronoun after it has been made into an NP. It will also match any possessive NP, such as: "the boy's" or "the boy's mother's". The rule then adds the feature determiner to the NP, making it eligible for the NP starting rule. By degrading the possessive NPs to determiners, both parsers easily handle examples of left branching such as:

[248] The boy's mother's brother is his uncle.

Another problem arose in [246] because the possessive NP rule was not sufficiently constrained. It is possible to use "her" as a determiner only where the next word can be part of a noun phrase with that determiner. To enforce this, the second buffer is checked to be certain that its contents will take the determiner. Using this approach "her" in [248] would not be converted to a possessive determiner. The rule
DETERMINER can run only if the next item will 'take a determiner'.

This check is made by the syntactic category of the following word, rather than by a specially marked feature. This check could be done by having a list of all the possible categories as the pattern of the second buffer. As an implementation detail, this is in the form of an agreement check, merely to simplify this rule and to show its generality.

The only remaining problem occurs when the verb can take one or more objects and the item after the word 'her' can be either the second object, or an NP with 'her' as a determiner. For example:

{249} I took her grapes.
{250} He saw her duck.
{251} I gave her food for the dog.

The examples presented above are all examples of global ambiguity, which will be discussed in Chapter 9. In these cases the check of 'will the next word take a determiner?', may or may not lead to the wrong analysis. This problem also interacts with the top-down component of verb phrase parsing and the semantic restrictions presented by it which will be discussed further in Chapter 10.

The conflict between the determiner and possessive usage can be modelled as a conflict of rule priorities. If the possessive use is preferred, then this rule should match first. Conversely, if the object use is preferred, then the object rule should match first. Any error in reading these examples would be due to one rule having priority over the other, when the reverse should be the case. Finally, notice that with no help from either intonation or context, either analysis is possible. That is, there is not enough information in the sentence to determine a unique interpretation. In Chapter 9, I will provide a more satisfactory explanation of these examples.

5.2.10 Handling THAT
In ROBIE, 'that' is defined as a singular determiner, a pronoun, a relative pronoun and a complementiser. Marcus had four diagnostics to handle the word 'that'. We have seen one of these at the start of this section. In this sub-section we will see how these four diagnostics can be replaced in a simple way. Let us consider how to handle the uses of 'that' one at a time.

Firstly, as a determiner. The following sentences illustrate the problem in identifying this usage.

[252] I know that boy should do it.
[253] I know that boys should do it.

I have stated before that Marcus [Marcus 1980] assumed grammatical sentences. If determiner/number agreement is not given to a parser, then it will, incorrectly, make 'that' a determiner in [253], producing the wrong analysis. The way to prevent this is to enforce number agreement in the rule DETERMINER by insisting that the determiner agree with the noun in number. The determiner usage will be grammatical only when the headnoun has the same number. If we make this a condition for the rule to match, then 'that' will not be made a determiner in [253] and we will get the correct parse.

How is this number agreement implemented? The rules are matched by the features of the first two buffers. To make agreement more transparent, another constraint has been added to the patterns of the rules, the "agreement requirement". It is possible to specify in the pattern of each rule that certain buffers must agree in certain ways. For the above examples, buffers 1 and 2 must agree in number. The agreement check is restricted to the same grammatical feature checking that the rest of ROBIE uses.

For this case, the agreement check would make sure that one of the following patterns are true:

     [ns] [ns]  
     [npl] [npl]  


Rather than having the following two patterns, only one pattern is needed and the agreement check acts as a "macro" to provide the two tests above.

\[
[\text{det,ns}] [\text{noun,ns}]
[\text{det,npl}][\text{noun,npl}]
\]

The rule DETERMINER can run only if, as in the above case, there is a noun in the second buffer agreeing in number with a determiner in the first buffer. See Section 8.6 for a further explanation of this check.

Another way to interpret the agreement check is to imagine the rule checking that the result will be grammatical before it is run. A non-deterministic parser would run the rule, discover the lack of number agreement and then have to backtrack. By checking for grammaticality before the rule can match, backtracking is avoided.

We have just added an extra check to the rule matching. This seems to go against our goal of simple and restricted checking of rules. However, this is not a violation of our goal, because we have restricted it to using only the items which the normal pattern matching can check. This check is really only a "macro". The patterns of the rule could be re-written quite easily so that they contained these features. This was not done because it would lead to more complex patterns. I feel it is more transparent and general to have the agreement check separated.

In some grammar formalisms, this agreement check will be easier to implement. If one assumes an analysis where the agreement features are part of the main features, then the agreement would happen automatically, since it is part of the rule patterns. These two cases are handled properly because number agreement blocks the interpretation of the [253] as a determiner. This approach leads to the correct preference, when there is an ambiguity and accounts for the difficulty in [254] vs. [255]:

{254} That deer ate everything in my garden
surprised me.

{255} That deer ate everything in my garden
last night.

118
Experiment two showed that [254] is a garden path sentence, while [255] is not. In both sentences, it is believed the subject uses the word "that" as a determiner. 'Deer' is both singular and plural, so it fits the above rule. In [254], it must be used as a complementiser to make the sentence grammatical. The approach outlined above will use "that" as a determiner in an ambiguous case such as this.

These two simple techniques, word order and agreement, are sufficient to handle all the examples we have just presented. In addition, free text analysis has shown no violations to this approach (See Appendix C).

"That" can only be a complementiser when a "that S-" is expected. Hence the rules using "that" to start an embedded sentence are only activated when the verb has the feature 'THAT-COMP'. The rules in 'THAT-COMP' will fire when "that" is followed by something which can start an NP. This ensures that the S- will have a subject and means that "that" will be taken as a pronoun in the following sentences:

[256] I know that hit Mary.
[257] I know that will be true.

but it will be taken as a complementiser in these sentences:

[258] I know that boys are mean.
[259] I know that Tom will hit Mary.

It seems that unless the S- has a subject, the pronoun use of "that" is preferred. Otherwise one would have a complementiser followed by a trace, rather than a unmarked complementiser, followed by a pronoun.

The rule to handle pronouns in general is of low priority and will only fire after all other uses have failed to match. "That" is treated in the same way.

"That" will be identified as a relative pronoun only if it occurs after a headnoun and the packet NP-COMPLETE is active. This situation will be handled in the same manner as the usual relative clause rules and will then cover:

[260] I know the boy that you saw.
[261] I know the boy that hit you.
The most difficult case for "that" is when the verb is subcategorised:

V NP S-

That is, it can take an NP subject, followed by a 'that' S-. For these examples, ROBIE may have to decide if the series of words following "that" is a relative clause or an embedded sentence. Examples of this are:

[262] I know the girl that Tom knows hit Mary.
[263] I told the girl that Tom was running alone.

In [262], we have a relative clause, whilst [263] contains an embedded sentence. It seems necessary to distinguish between these because of movement. If the NP "the girl" is to be moved, the movement mechanism needs to note this fact. If the mechanism tries to move the NP, but cannot find a place to put it, the parser has made an error. If the parser did not try to move it and then discovered that it should have, it has again made an error.

There is no way that a deterministic parser with three buffer lookahead, let alone two buffer lookahead, can "see" enough to handle this case. In the following sentences, the lookahead would have to be more than three buffers. (Brackets indicate words in the buffers. The last word is the disambiguating word.)

[264] I told the girl [that][the][boy] hit the story
[265] I told the girl [that][the][boy] will kiss her.

It can be seen that in these sentences, the disambiguating word is outside our three buffers. How do people handle these and what should our parser do? In Chapter 4 it was shown that when the syntax could not resolve the ambiguity with its two buffer lookahead, the decision of which interpretation to use might be made using non-syntactic information. In Chapter 4, we stated that if context can effect the interpretation of the sentence, then non-syntactic information is being used to select the interpretation. The reader can experiment for himself and see that context does affect the interpretation of these sentences. Therefore we predict that non-syntactic information is being used to interpret these sentences and this problem should not be resolved on a syntactic basis, but a non-syntactic one.
This explains why some of these examples cause difficulty and others do not. The psychological evidence from cases using "that" is scant and I feel no conclusions can be reached here. My theory predicts that context will strongly affect these examples and, if they are strongly biased to the incorrect reading, a garden path should result.

One well known example in this area is [266]:

[266] I told the girl that I liked the story.
[267] I told the girl whom I liked the story.
[268] I told the girl the story that I liked.

These examples were tested in the second experiment. The results suggested that [266] was read faster than the other two examples. Many of the subjects were questioned informally after the experiment about their interpretation of the sentence. All reported only one meaning; the S-reading. None of the subjects said that they noticed the relative clause reading, hence the result. The experiment however, was not designed formally to distinguish these.

[266] provides another example of global ambiguity. The second experiment has suggested that in a conflict the S-reading is preferred.

Another possible explanation is that the "complementiser" usage is preferred syntactically. We will return to the question of preference in Section 6.5. Syntactic defaults have been mentioned for the other examples we have considered here. Since the syntactic processor is deterministic, it will never backtrack. It is, therefore, not possible for it to try a preferred reading and then an alternative reading. If a situation arises where the syntactic processor, with its limited lookahead cannot resolve an ambiguity in general, we have stated that the non-syntactic processor choses one alternative. If the non-syntactic processor does not have a preference, then it may be possible for it to chose a 'syntactic default'. We will return to the question of syntactic defaults and preferences in Section 6.5.

To handle the examples we have seen in this section, Marcus had four diagnostics, one of which was very complicated. I have just shown how to handle all
four cases of "that" without any special rules, merely substituting enforced agreement and rejecting ungrammatical sentences.

5.2.11 Handling the Word HAVE

Let us now look at the elimination of Marcus's HAVE-DIAG in relation to the use of agreement we have been discussing in this section. The problem with 'have" is illustrated by the following sentences:

[269] Have the students take the exam.
[270] Have the students taken the exam?

In these, we must decide if 'have" is an auxiliary verb or a main verb and whether the sentence is a yes-no-question or an imperative. The sentences have the same initial string until the final morpheme on 'take'. To handle this case, Marcus used this rule: [Marcus 1980, p. 211]

"[RULE HAVE-DIAG PRIORITY:5 IN SS-START
[have,tenseless][np][t] ->
If 2nd is ns,n3p or 3rd is tenseless
Then run imperative next else
if 3rd is not verb then run yes-no-q next else
%if not sure, assume it's a y/n-q and %
run yes-no-question next]."

This rule seems to be necessary in order to distinguish between the question and the imperative. If one tries to ascertain exactly what occurs, the apparent complexity is revealed. Note also that Marcus defaults to a yes-no-question twice in this diagnostic. The following sentences illustrate the distinction this rule makes.

[271] Have the boy take the exam.
[272] Have the boy taken the exam.
[273] Have the boys take the exam.
[274] Have the boys taken the exam?

It can be seen that YES-NO-QUESTION should run only when the NP following is plural and the verb has "en" (i.e., "taken"). For example, in [274] "the boys" is plural and the verb is "taken". None of the other examples above have both "boys" and "taken". This can also be understood as: the sentence is an imperative if the item
in the 2nd buffer is not plural and the verb is tenseless. [271-273] are Imperatives because either the noun (boy) is singular ([271] and [272]) or the verb is tenseless ([273]). The second part of the rule takes care of the fact that the third buffer must contain a verb for the imperative, as this would be the main verb of the embedded sentential object.

Let us look more closely at the reason for only [274] being a question. Firstly, if the sentence is a yes-no-question, then aux-inversion must occur. When this happens, 'Have' will be adjacent to the verb which was in the third buffer. In order for ROBIE to continue, the verb must have an 'en' ending, or 'have' and the next verb will not agree in aspect. This is the basis for discrimination in the earlier examples [271-274].

Secondly, in [271] and [272], the noun phrases are singular and both sentences are imperatives. This is because, if the sentence had been a yes-no-question, 'have' would need to agree with the subject which must then be plural.

Hence, in effect, Marcus's rule checks for number agreement between the subject and verb and that the fixed order of the verb group is obeyed. Let us now look at other situations where this is necessary.

PARSIFAL would accept the following ungrammatical strings:

[275] *Are the boy running?
[276] *Has the boys run?
[277] *Has the boy kissing?
[278] *Has the boy kiss?

For a yes-no-question, the inverted auxiliary must agree with the verb after it has been inverted. To stop these ungrammatical items we must enforce verb agreement. The pattern for the rule YES-NO-QUESTION should be:

[auxverb][np][verb], agree(auxverb,verb),agree(verb,np).

This constraint enforces agreement of the verb and auxiliary verb and the subject and verb. Again this check is based only on the linguistic features of the buffers. See Section 8.6 for further details.
Such a constraint effectively blocks the ungrammatical items. (The parser will fail if the auxiliary has been inverted, since the auxiliary will not be parsed.) Also the subject NP must agree with the auxiliary verb, so we can also add 'agree(auxverb,np)' to the rule, as we did with the HAVE-DIAG! So, by fixing the yes-no-question rule, the HAVE-DIAG is redundant.

The reader will notice that the HAVE-DIAG handling of the agreement as stated requires an NP in the second buffer and three buffers. This NP can only arrive in the second buffer via an Attention Shift. Since we have removed the Attention Shift and use only two buffers of lookahead, it is not possible to implement this rule in ROBIE. We will return to this problem in Chapter 9.

5.2.12 The Word A

The final diagnostic of Marcus's we will look at is the A-HUNDRED-DIAGNOSTIC. This diagnostic was developed to select the correct usage of 'a' in the following sentences.

[279] A hundred boys are in the class.
[280] A hundred pound rock is in the car.

Marcus considered "a" to be a determiner in [279] but part of the number phrase in [280]. His diagnostic used a second Attention Shift and all three buffers to select the correct interpretation. Whilst this diagnostic was often correct, Marcus admits [Marcus 1980, p. 214] that it would fail on the second of this pair.


Marcus then presented informal experimental evidence to show that [282] is a garden path for at least half the subjects he tested [Marcus 1980, p. 214]. This diagnostic has several things in common with the HAVE-DIAGNOSTIC in the last section. It requires 3 buffers, the Attention Shift and is not always correct. As with the HAVE-DIAGNOSTIC, we will not use the A-HUNDRED-DIAGNOSTIC.
Marcus also demonstrated that there is no strategy based on three buffer lookahead which can always resolve this ambiguity correctly. We have seen a situation similar to this in Chapter 4. In that chapter we saw several examples of ambiguity which we could not disambiguate on the basis of limited lookahead. For each case we saw that the ambiguity was resolved on the basis of non-syntactic information.

The cases we looked at in Chapter 4 were all end of NP problems. However, they suggest that the general statement of WHEN non-syntactic information is used is:

WHEN - When syntactic information with limited lookahead will not always be sufficient to choose between alternatives.

One can now see that this statement applies to the A-HUNDRED-DIAGNOSTIC examples, which suggests that this ambiguity could be resolved on the basis of non-syntactic rather than syntactic information. Unfortunately there is insufficient psychological data on these examples to confirm that this suggestion is, in fact, true. Although Marcus required three buffers to resolve this ambiguity, our theory suggests that this decision may be made based upon non-syntactic information, rather than on the purely syntactic basis which Marcus suggests.

We have now shown how to replace all the diagnostics Marcus used. In doing this, we enforced number and verb agreement on the rules before they could run. This was motivated to reject ungrammatical items, rather than handling of ambiguities. Whilst there are still a few problems which we will return to in Chapter 9, the approach reported here has the same coverage as Marcus's diagnostics, and provides a better explanation of why people have trouble on certain sentences.

5.3 Possible Uses for Agreement in English

Linguists can describe the use of verb agreement and person/number codes. It is quite clear that these must be enforced in grammatical sentences, but most
linguists do not offer an explanation of why we enforce number agreement and the fixed order of the verb group. In other words, our understanding of agreement is descriptive, but not explanatory. What role does number agreement and the fixed order of the verb group play?

In this section we have seen a possible explanation of this puzzle. We have seen several occurrences of ambiguity, for each of which, we have found a parallel situation that could lead to acceptance of ungrammatical sentences by ROBIE. We then used person/number codes or the fixed structure of the verb group to block these unacceptable readings. Most of our ambiguity problems were also handled by this method. Although this has been used before with non-deterministic parsers, we did not know how well it would work in a deterministic parsing environment.

In this section we have looked at language from a processing viewpoint. From a linguistic viewpoint, word order and agreement could describe language, but could not explain it. But from a processing viewpoint, we saw that word order and agreement were essential to the resolution of ambiguity in language and help to explain why people can resolve ambiguity so easily.

Once person/number codes are taken into account, the ambiguity problems are reduced, for in each case, only one of the ambiguous possibilities was grammatical. It seems that person/number codes might reduce the ambiguity in natural language parsing and this hypothesis needs further investigation.

Marcus had a few rules to resolve part of speech ambiguity, but they were ugly and ad-hoc. We have seen that we can replace these rules very simply by merely exploiting agreement.

This now concludes our investigation into the resolution of lexical ambiguity. We have seen how to handle cases of lexical ambiguity which can lead to a garden path and cases which do not lead to a garden path. We have developed and improved our model to make it psychologically plausible. In the next chapter we will investigate how compatible the model is, as developed so far, with the relevant
psycholinguistic literature.
The Psychological Status of Deterministic Parsing

6.1 Psychological Criteria

At various points in the thesis, arguments for certain strategies to make ROBIE psychologically plausible have been put forward, but how does it stand in comparison with the psychological evidence? In this chapter many of the important results on psychological plausibility and their main points will be explained. We will then ask, 'Can our model account for or at least agree with these results'?

While it is claimed that the model presented here is a psychological model of the Human Sentence Parsing Mechanism (HSPM), it is not claimed that it is a complete model. My claim is a lesser one; that the overall design of ROBIE can describe, and explain, much of the data collected on how people perform in various linguistic contexts. This claim only applies to the areas this research has specifically addressed. Many details of ROBIE are clearly incorrect and for these no claims are made. Only the major principles such as determinism and limited lookahead, and the timing of the interaction of syntactic and non-syntactic information are considered relevant. It is known that the current model is not a sufficient model of the entire HSPM, but it is felt that it is closer than previous models. The success of this model will provide a foundation on which to build a better model. The shortcomings of the model will show which areas to develop in order to build that better model.

To evaluate this model, we can start with the criteria of Fodor and Frazier, [Fodor and Frazier 1980] who say the human sentence parsing mechanism must answer the following questions:

a) 'Does it succeed in parsing all and only the sentences and non-sentences of the language that the native speakers succeed in parsing?'
b) "Do its relative parsing times for different sentences match the relative parsing times of human subjects?"

c) "When it makes parsing errors, do these resemble the errors made by human subjects?"

Clearly these criteria are barely adequate, but as no working parser today can meet all of them, they are a start. It is very important that the model not be capable of too much and out-perform people. If the model can process sentences which people are not able to understand, then it is too powerful. Conversely, any parser that does not have a grammar covering all of the language will not meet the above criteria. No current computer model has the full range of senses, knowledge, information and capabilities which people have. Because of this, there is no way to design successfully a psychologically real parser today.

One reason that it is impossible to show that a given parser is psychologically real, is that no one is sure exactly how the HSPM works and what is its design. In fact, we may probably never be sure how it works. This same problem puts limits on the extent to which we can dis-prove a model. If ROBIE does something clearly wrong, then it can be dis-qualified, but if it does not do anything clearly wrong, it must be considered correct until it can be shown in error by a better model. More and better data need to be collected before better models can be designed. It will be shown throughout this chapter that the limitations of ROBIE are very close to the limitations actually observed in people.

6.2 Kimball's Principles

We will begin our discussion by looking at the general principles which the human parsing mechanism seems to follow as described by Kimball [Kimball 1973]. He proposed seven principles of Surface Structure Parsing. We will examine each of these to guide our discussion. In the following section, we will first look at
his principle, then a brief explanation of it and finally how this principle fits the current parser.

1. (Top Down) Parsing in natural language proceeds according to a top-down algorithm.

Top-down parsing implies that a parser has expectations about what will follow, even before it has seen the data. The point most important to our discussion of top-down parsing is the need to reflect the expectations of possible following items.

ROBIE is partially top-down. When Marcus [Marcus 1980] designed his parser, he motivated the need to reflect expectations in the parse. Packets were introduced for this and provide the top-down component. This top-down component is essential to ROBIE. We have already seen several cases of ambiguity where these expectations are essential (such as handling the words 'to' and 'that'). Marcus has motivated the need to also be bottom-up, which PARSIFAL and ROBIE are as well. By design, these parsers meet this principle.

2. (Right Association) Terminal Symbols optimally associate to the lowest non-terminal node.

This principle reflects the fact that right branching structures seem to be preferred in natural language. Kimball's evidence is based on sentences of the form:

[283] Joe figured that Susan wanted to take the cat out.
[284] The girl took the job that was attractive.
[285] Joe called the man who smashed his new car up.
[286] Joe said that Martha expected that it would rain yesterday.
[from Kimball 1973]

In each of these examples, the final modifying item can be associated with more than one location, but the preferred reading for each of these is to attach it to the lowest and rightmost item. This principle is used to explain the preferred reading for sentences of this type. Bever [Bever 1970] had tried to explain these examples as
part of memory limitations and Church [Church 1980] has explored this problem from the same viewpoint. This principle has started much discussion and will be dealt with separately when we discuss the 'Sausage Machine'.

3. (New Nodes) The construction of a new node is signaled by the occurrence of a grammatical function word.

This principle is fairly self explanatory. The main intent of this principle was to explain why the lack of function words can make processing more difficult when compared to the same sentence that has the function words added. If it is interpreted more strictly, this has problems, (see [Frazier and Fodor 1978], [Church 1980]) but we can satisfy the basic intent. Kimball's examples are:

[287] He knew the girl left.
[288] He knew that the girl left.

While it is true that grammatical function words do start new nodes, it is not true that nodes are started only by the presence of a function word. The parser follows Kimball's principle of 'New Nodes' closely. Prepositions start new PPs, an auxiliary verb starts the AUX, and determiners start NPs. As Kimball predicted, the parser has difficulty when the grammatical function words are not present. In Marcus's parser, prepositions did not start PPs, instead the PP was built after the NP was parsed by the Attention Shift. But when the parser was modified to follow this principle, the extra mechanism of an Attention Shift was no longer needed. (see Section 2.5.2)

This is one of the improvements of ROBIE when compared with Marcus's parser. His parser did not follow this principle, and was conceptually more complicated. ROBIE follows this principle and is conceptually simpler.

4. (Two Sentences) The constituents of no more than two sentences can be parsed at the same time.
Kimball motivates this rule with the following pairs of sentences:

[290] ?That that Joe left bothered Susan surprised Max.
[291] The boy the girl kissed slept.
[292] ?The boy the girl the man saw kissed slept.

Church [Church 1980] has fully explored this principle. He introduces the A-over-A closure principle:

The A-over-A early closure principle: Given two phrases in the same category (e.g. noun phrase, verb phrase, etc), the higher closes when both are eligible for Kimball closure. That is, (1) both nodes are in the same category, (2) the next node parsed is not an immediate constituent of either and (3) the mother and all obligatory daughters have been attached to both nodes. [Church 1980]

Church has a very convincing discussion on how his deterministic parser explains this principle. Neither Marcus nor I have investigated this principle and the center embedded sentences it explains in detail. However, [Cowper 76] presents a theory of this phenomenon which is consistent with the model presented here.

5. (Closure) A phrase is closed as soon as possible, i.e., unless the next node parsed is an immediate constituent of that phrase.

In ROBIE, a node is closed by ATTACH, when the node is attached to its mother. This node then, is closed when we are sure it is finished. This approach follows roughly the above principle. This principle interacts with Right Association (above) and Kimball wonders whether it is actually distinct. Church [Church 1980] has an extensive discussion of this point along with the counter proposals of Frazier and Fodor [Frazier and Fodor 1978]. My account for this point will be covered in the discussion of the Sausage Machine below.

6. (Fixed Structure) When the last immediate constituent of a phrase has been formed and the phrase E is closed, it is costly in terms of perceptual complexity ever to have to go
back to reorganise the constituents of that phrase.

Kimball uses this principle to explain why garden paths are so difficult to handle. This is the essence of determinism. Once a decision is made, it cannot be reversed. The current parser has not been designed to recover from mistakes, so violation of this is impossible.

7. (Processing) When a phrase is closed, it is pushed down into a syntactic (possibly semantic) processing stage and cleared from short term memory.

In previous chapters, the "rule by rule" approach to semantic note taking has been motivated. It has been proposed that the semantic interpretation stage happens continuously, not only after a node has been closed. In ROBIE, the node is closed when it is attached to its mother. Because of this, the node will no longer be in the Active Node Stack, or any of the buffers. Once a node has been attached, it is not possible syntactically to examine its structure. This restriction then, meets this principle.

In summary, the seven principles of Kimball can all be met in ROBIE. Most of the principles are necessary results of ROBIE's design and principles. The remainder are necessary results of Church's work on memory limitations.

6.3 The ATN as a Psychological Model

The ATN was proposed as a psychological model in [Kaplan 1972]. In this next section, we will not explore the validity of the ATN as a psychological model. Instead, we will use the criteria which Kaplan established and used in that paper to see how it fits ROBIE.

Several experimenters, Mackay and Bever [Mackay and Bever 1973], Wanner [Wanner 1968] and Bever [Bever 1970], performed experiments to establish
the reality of the deep structure-surface structure distinction. Kaplan observes:

'These experiments suggest that an adequate model of sentence comprehension must incorporate some mechanism for recovering a deep structure representation of a given stimulus word string' [Kaplan 1972, p. 80]

Kaplan presents three other requirements for adequacy which our model must meet. These are:

1) 'A perceptual model must process strings in essentially temporal or linear order, for this is the order in which sentences are encountered in conversation and reading.'

ROBIE is a left to right parser just as Kaplan's ATN was. Whilst there are parsers which do not observe this, many parsers do, including mine.

2) 'It must process strings and provide appropriate analysis in an amount of time proportional to that required by human speakers. For example, since perceptual difficulty does not rapidly increase in length as the length of the sentence increases, the amount of time required by the model should be at most a slowly increasing function of sentence length.'

The time taken by ROBIE to parse a sentence is roughly linear in the number of words in that sentence. The parse time should not be an exact function of the number of words in a sentence, but rather a function of the complexity of the sentence. In ROBIE, the time varies, within sentences of the same length, according to their complexity. This will be fully explained in Chapter 8. The complexity of a sentence is difficult to measure, so we will assume that, if the parse time grows roughly linearly with the number of words, subject to the following point, then it meets this requirement. ROBIE certainly meets this point.

3) 'The model should discover anomalies and ambiguities where real speakers discover them and for ambiguous sentences the model should return analyses in the same order that speakers do.' [Kaplan 1972, p. 80]
This point has been the main motivation for all the work in this paper. We have tried to demonstrate for each ambiguity handled, that the parser does as people do. We have used the performance of people in an ambiguous situation to decide what strategy ROBIE will use. We have also shown how and why ROBIE will fail on the same sentences as people fail on, i.e., the garden paths.

For some ambiguities, psychologists cannot agree on what people do, but there is no reason ROBIE cannot do whatever it is people do once it is agreed. In this situation, requirement 3) is applicable. Kaplan assumes that all readings of an ambiguous sentence are perceived and in a certain order. I feel that there is insufficient evidence for this conclusion. This point will be discussed in full in Chapter 9 and an alternate explanation proposed.

Kaplan's first examples are:

[293] The dog bit the cat because the food was gone.
[294] Because the food was gone, the dog bit the cat.
[295] The editor the authors the newspaper hired liked laughed.
[296] The editor authors the newspaper hired liked laughed.

[Bever's 1970, 24a-b, 27a-b]

In these examples, [294] is supposed to be more difficult to process than [293], and [295] is more difficult to process than [296], even though Kaplan admits that both are exceedingly difficult since they are center embedded.

For [293] and [294], Kaplan's account varies by one step in the ATN. My grammar has not been extended to deal with these specific examples, but the fronted phrase will require several additional steps in the syntax and an extra inference at least in the semantic interpretation. ROBIE should have as much extra difficulty as the ATN.

In [295], there is the noun/verb/plural end of clause problem which was discussed at length in Chapter 3. Because [296] has a determiner before "authors", it does not have the noun/verb/plural ambiguity problem and hence will not require a semantic check, making it easier to process. Kimball's Principle number 4, 'Two
Sentences explains why these are difficult and Church's account of what to do in this situation shows these are unparsable for his parser. It is accepted that people also have trouble here. These examples illustrate the power the ATN has and how it can out perform people in some situations.

The next examples Kaplan uses are these:

[297] They are fixing benches.
[298] They are sleeping monkeys.

Kaplan then demonstrates that six more arcs need to be traversed before a successful parse for [298] than [297]. This account is based on the strategy that "sleeping" is intransitive whilst "fixing" is transitive. This seems right, but the alternative reading for both sentences is possible as these two illustrate:

[299] They are fixing agents.
[300] They are sleeping pills.

This account would predict that these alternative readings are not possible or should cause greater perceptual difficulty, which they do not seem to do. Our next examples are:

[301] The red plastic box....
[302] *The plastic red box...
[303] The large red box...
[304] *The red large box...

[Bever 1970, 67 a-d]

Kaplan accounts for these by adding 'nounness' to the adjectives. This approach is certainly not a feature of the ATN only. If desired, nounness could be added to the current parser to handle this. In the current parser, the non-syntactic processor can abort any of these examples should it make no sense. If the non-syntactic processor is in "accept anything" mode, then these are acceptable. I feel this accounts better for the performance of people on these examples.

Finally Kaplan shows that his model can predict the preferred reading for sentences such as:

[305] They are frightening monkeys.
[306] The Irish water boils.
This prediction is based on the ordering of arcs. By adjusting the order in which arcs are attempted, it is possible to get the desired readings. Unfortunately, his account does not explain why the preferred reading of [305] differs from the preferred reading for [297]. It also does not explain how the same person can have different preferences in different contexts. Tyler [Tyler and Marslen-Wilson 1977] has convincingly demonstrated that prior context has a major effect on the preferred reading of these examples. Kaplan's arc order account, as presented, does not account for these facts. In Chapter 9, it will be shown how our theory of interaction with the non-syntactic processor can account for these more satisfactorily.

Sentences [306] and [307] show the noun/verb/s problem which was the subject of Chapter 3. It was explained that a non-syntactic decision, based on knowledge, intonation and context, is what affects this reading. Kaplan's simple arc order account is not adequate to cover the data presented in Chapter 3 and Experiment One (Section 3.4). It also seems that the arc order account is unable to explain how preferences change, but the non-syntactic decision account explains how this can be.

The other difficulty with the ATN model is in garden path sentences. In Section 4.8, it was shown that people have considerable trouble trying to understand garden path sentences. The backtracking account of garden pathing does not provide an adequate explanation of this. If recovery from a garden path sentence is as simple as backtracking, then people would not have as much trouble recovering from a garden path as we discussed in Section 4.8.

In summary, Kaplan presented several very interesting examples and showed how his ATN parser could account for them. We have seen how ROBIE can also explain these same examples. Whilst this alone is insufficient evidence that ROBIE is psychologically plausible, ROBIE does better than Kaplan's ATN.
6.4 The Sausage Machine

Lyn Frazier and Janet Fodor (FF) in [Frazier and Fodor 1978] propose a two stage model of the Human Sentence Parsing Mechanism (HSPM), called the Sausage Machine (SM). This model has sparked a debate between its supporters and those of the model with which it is compared, the ATN. In the next section, the advantages of the SM and how well FF's data fit ROBIE will be looked at.

FF proposed that the syntactic analysis of sentences by hearers or readers is performed in two stages. The first stage combines words in phrasal nodes as they are received. They call this the 'Preliminary Phrase Packager' (PPP) or the 'Sausage Machine'. The second stage combines these phrases into sentences. This stage is called the 'Sentence Structure Supervisor' (SSS).

They did not propose a specific mechanism for parsing and did not even make specific proposals of exactly how the devices should work. Their entire discussion was very general and hence it misses many important points. For example, they claimed that the PPP can see several words at a time. They did not give a number, guessing that it may be seven plus or minus two. All the NPs they used in the paper were of simple enough structure so that typically an NP with a PP seemed to fit into the PPP's range.

They have been accused of not having said enough to make their theory testable. In [Fodor and Frazier 1980], they answered this accusation by saying: this did not show that, 'the model, in so far as it is specified, is false'. They defended themselves by saying that they had not said enough to be shown wrong yet. It could be argued, as others have, that they have not said enough to allow others to decide whether their model is correct or not. It would be fair to let them not specify detail, as long as they did not criticise other models to a level of detail which they have not resolved. However, in this paper, they criticise the ATN on very specific
details, details that are not even thought out for the SM yet. The fine interaction of various components of the system is very important to any model; that is partially why we build working computer programs. Because of this lack of detail, I feel this model cannot be fully evaluated.

Their first main point is the Principle of Right Association (RA). This is a slightly different version of Kimball's Right Association. FF feel that there is some data that Kimball's RA cannot account for. The difference is the way it interacts with their second point. We will return to this later. Right Association states that 'terminal symbols optimally associate to the lowest non-terminal node' This predicts the preferred interpretation of:

[308] Tom said that Bill had taken the cleaning out yesterday.
[309] Joe called the friend who smashed his new car up.
[310] John read the note, the memo and the letter to Mary.
[311] The girl took the job that was attractive.
(from [FF 1978, p. 297])

In each of these sentences, the preference is to attach the final modifier to the lowest right node. This is what their principle would predict. This principle also predicts the difficulty in the following sentences.

[312] Joe looked the friend who had smashed his new car up.
[313] John read the note, the memo and the newspaper to Mary.
[314] The girl applied for the jobs that was attractive.

FF then tried to explain how this principle might be true in their parser. Their explanation was complex and depended on memory limitation. Church [Church 1980] has answered their memory limitation argument and revealed several problems in their explanation. He then provided a much more satisfactory account. His parser has a strict limit on the number of items which can be on the Active Node Stack. It is not possible for Church's parser to parse the sentences which violate FF's principles, because those sentences require too many incomplete constituents to be stored on the Active Node Stack. We will accept Church's explanation for the time being, but will look at an alternative explanation below.
Their other main point is what they called 'Minimal Attachment' (MA). This says 'Each lexical node (or other node) is to be attached into the Phrase marker with the fewest possible number of non-terminal nodes linking it with the nodes which are already present'. [FF 1978, p. 320] This principle accounts for the preferred attachment of 'for Susan' to the VP in:


They suggested that this accounted for the preference for the conjunctive analysis of NP NP in center embedded sentences and the preference for the first clause to be a main one, as we have seen in Chapter 3. They also feel that this accounts [Wanner, Kaplan and Shiner 1975], for 'that' as a complementiser rather than a relative clause when after a NP. It even predicts the use of 'that' as a determiner over the 'comp' usage.

Even though I disagree with the evidence they presented to justify this, let us accept it for now. However, not everyone has accepted the claims of this paper as true. Wanner has replied to their claims and defended the ATN as a model of human sentence parsing [Wanner 1980].

In this paper he defended the ATN against the arguments of FF, saying that they said that he could not model Minimal Attachment and Right Association in a principled and independent way. In this paper he demonstrated that he could. He also attacked the SM and showed it to be inadequate.

Wanner explained various defects in their proposals, which will not be repeated here. The full discussion is in [Wanner 1980]. FF have made several claims not mentioned here as they are neither relevant to our discussion here nor specified in enough detail to be considered.

Wanner critised FF for the six window lookahead. He asked, what if the sentence is less than six words long? Do RA and MA still operate? Wanner demonstrated that they do and stated that this lookahead is not justified. Wanner's criticism was valid when one considers the 'short garden paths' such as 'the prime
number few.". The data presented in Chapter 3 demonstrated that all the information available in a 3 window lookahead is not always used, so therefore not all the information available in a 6 window lookahead will always be used. In that chapter, it was demonstrated that by using a three window lookahead and utilising all information in it, one will get incorrect predictions of garden paths. They have also not explained how and why the size of the window has varied. I feel that our account, with a window of two buffers, is much more satisfactory.

We will now look at the summary of their main points.

The main point of Frazier and Fodor are these two principles:

1) Right Association... an ambiguous constituent should be 'attached into the phrase marker as a right sister to existing constituent and as low in the tree as possible' [Wanner 1980, p.211] [FF 1978, p. 294]

2) Minimal Attachment... an ambiguous item "is to be attached into the phrase marker with the fewest possible number of non-terminal nodes linking it with the nodes already present" [FF 1978, p. 320]

To see how this is accounted for, we will first look at Wanner's account of their data. In accounting for this data, Wanner first gave some background to the ATN and presented these possible arc types:

- WORD arc - analyse specific words such as "that" or "to"
- CAT arc - analyse grammatical categories such as Noun and Verb
- SEEK arc - analyse whole phrases such as NP, VP or S
- SEND arc - terminate a network (node)
- JUMP arc - express optionality [Wanner 1980, p.216]

He then stated the two principles in terms of these arcs:

Right Association: Schedule all SEND and JUMP arcs after every other type of arc.

Minimal Attachment: Schedule all CAT and WORD arcs before all SEEK arcs.

He presented an argument that these arcs characterise the principles correctly. FF agreed in their reply to him [FF 1980]. (Wanner went through their examples in detail). Wanner claimed to show that the ATN can account for the facts
which FF have claimed he could not account for. I feel that he accounted for the data in the ATN model better than FF did in their model.

FF agreed basically with this re-formulation. In their reply to Wanner, [FF 1980], however, they pointed out several problems with Wanner's account. First, Minimal Attachment. Wanner's re-formulation as 'CAT before SEEK' does not say what to do when two SEEK arcs conflict. They then presented some sentences which they predicted the ATN would err on.

SEEK arcs provide top-down parsing by predicting what structure will be needed next, before the next word is seen. Their counter examples depend on the SEEK arc building an unjustified structure when a more justified structure can be built. The relevance for ROBIE of their counter examples depends on the assumption that ROBIE has SEEK arcs. This prediction of nodes before the data has been seen could lead to backtracking. Therefore ROBIE does not have SEEK arcs.

FF have some trouble with the sentence:


They tried to explain the preference for this as the interaction of MA and RA and did this on a purely syntactic basis. For each example and counter example they have presented, ROBIE has to make a non-syntactic check. Therefore, our account is based on non-syntactic preference, while theirs is on syntactic preference. They tried to explain the preferred placement of the PP, (for Susan), to the VP (to obtain) in:

[317] John bought the book that I had been trying to obtain for Susan.

on a syntactic basis. We have asserted in Chapter 4 that PP attachment must take into account the selectional restrictions of the verb, the current context, intonation and discourse. FF have tried to account for preferences, based on non-syntactic information, on a purely syntactic basis. It has been demonstrated in this thesis that non-syntactic interaction plays a very important role in the HSPM.
It is not clear what role non-syntactic interaction should play in the SM. Without non-syntactic interaction, I see no principled way in which preferences can be altered in different discourses. FF have not specified when RA and MA can be over-ridden to gain the non-preferred reading, and what mechanism can control this change. In ROBIE, the non-syntactic processor assists with resolving these ambiguities. If it makes a different decision based on the current context, then the non-preferred reading will be selected. Any model of the HSPM must explain when and why non-syntactic information is used during parsing. I feel that the model presented by FF does not account for the data which has been presented in this thesis.

While clarifying Right Association, FF introduced the need for the SM to forget information from earlier parts of the sentence. For example, they said that the system should forget that "called" could take a particle. However, there is another way to look at the same problem. In ROBIE, if the VP with "called" as the main verb is not the Current Active Node, then the Packet that contains the rule to attach the particle is not active. In sentence [318], the VP with "smashed" will be below the VP with "called" on the Active Node Stack, so the particle rule will associate "up" with "smashed".

[318] Joe called the friend who smashed his new car up.

Rather than forget, ROBIE has no way of knowing the answer.

While accepting Wanner's characterisation of their principles, FF also attacked the ATN with a long, detailed and complicated argument. The main weakness in the ATN stems from mis-guided SEEK arcs, which is irrelevant to our account, as ROBIE has no equivalent of a SEEK arc. The top-down component of ROBIE expresses what may follow, but does not actively seek those items.

Wanner has shown that the ATN can describe the strategies of MA and RA, but cannot explain why these strategies are present. So even though the ATN can show sufficient description, it cannot show necessity. Wanner even wonders this
when he asked: "Why does the parser employ these strategies as opposed to others?"

[\cite{Wanner 1980, p.233} He then admitted that no clear answer was available to this question for an ATN.

In the next section, It will be shown that not only can I describe the principles above in the same way, but I can explain why they must be true.

6.5 Accounting for RA and MA in ROBIE

6.5.1 Production System Rule Order

In some production systems, the following principle gives the order in which to test each rule: 'The most constrained pattern is tried first'. For if the rules were tried in the opposite order, the more constrained rules would never be matched. This provides an order for rules of unequal constraint. The following rules will be tried in the following order:

\[
\begin{align*}
\text{[so]} \text{[that]} & \rightarrow \quad \langle \text{action1} \rangle \\
\text{[to]} \text{[tenseless]} & \rightarrow \quad \langle \text{action2} \rangle \\
\text{[that]} & \rightarrow \quad \langle \text{action3} \rangle \\
\text{[noun]} \text{[noun]} & \rightarrow \quad \langle \text{action4} \rangle \\
\text{[noun,npl]} & \rightarrow \quad \langle \text{action5} \rangle \\
\text{[verb]} & \rightarrow \quad \langle \text{action6} \rangle \\
\text{[t]} & \rightarrow \quad \langle \text{action7} \rangle 
\end{align*}
\]

A pattern with one word is more constrained than a pattern with two features, since there is only one lexical item which can match the first, but several lexical items which may have, say 'tenseless'. A rule with no pattern [t] will always be the tried last. This is necessary to handle many ambiguity issues. For, if the rules were tried in the opposite order, the more constrained rules would never be matched. I emphasise, this same principle says that all default rules (rules with no pattern) will have lower priority than any other rule.
6.5.2 Right Association

ROBIE does not have the types of arcs which were listed for the ATN earlier, but the rules can be divided into several roughly similar groups. The equivalent of the SEND and JUMP arcs would be the default rules in a packet. If something is optional, typically a packet has a rule to handle the marked case and a default rule to handle the unmarked case, that is the default rule has no pattern. All equivalents of the SEND and JUMP arcs will have no pattern in the current parser. Hence, according to our above ordering, these rules will be tried last. Thus Wanner's explanation of Right Association is a necessary result of ROBIE's design.

6.5.3 Minimal Attachment

The deterministic parser has no SEEK arcs. A grammar rule in ROBIE with the pattern [np], does not create an NP. Instead, this pattern will match only if a NP node has already been started. But in the ATN, the arc with the NP on it will cause a push to the NP subnetwork and try to build a NP. Ordering this SEEK arc is the problem under discussion here.

On the subject of no SEEK arcs, Marcus states:

"The pattern that triggers on a specific constituent, say a NP or an S, does not initiate parsing of a constituent of that sort. Instead, the pattern will only trigger if a constituent of that sort is already in the specified buffer." [Marcus 1980, p.22]

If a pattern has the feature NP, this does not make ROBIE try to parse an NP. Instead the pattern will match only if a node with that feature has already been built. This can be contrasted with the SEEK arc of the ATN. The SEEK arc tries to build a node of the type which was specified on it. SEEK arcs are like recursive subroutine calls.
Because ROBIE does not have SEEK arcs, the problems of ordering them are not relevant. The CAT and WORD arcs will be scheduled first as Wanner has shown necessary.

MA as characterised by Wanner states that essentially the parser should be data driven and should reflect the incoming words. Another way to understand the principle of Minimal Attachment, is that the word should be used locally if it fits. Since ROBIE has no access to the Active Node Stack, except for the active packets, then it is unable to see if the word could be used higher up. If the word could be attached to the lower node, then the grammar rules must be written to handle it there. If these rules are there, then the optional use will be grabbed and this will behave exactly as Minimal Attachment.

One can see that ROBIE explains RA and MA as necessary side effects to the handling of some types of ambiguity. FF and Wanner are both unable to show, simply, why these principles are true. In this section we have seen that they must be true in ROBIE.

6.6 Some Predictions

Chapter 5 explained that words have a compound lexical entry in the dictionary incorporating each part of speech definition for that word. When the word is looked up, this compound lexical entry is returned. This was introduced purely to make the handling of ambiguity automatic. How well does this fit the psychological explanations?

We are interested in this question for two reasons. Firstly, as a psychological justification of the lookup routine. We have seen that it was necessary to have all meanings arrive at once so that the automatic handling of ambiguity will work. We will see that the method that is necessary for ROBIE to use, is psychologically valid and the same method that the lexical access literature prescribes.
Secondly, to check the prediction of the two buffers. The parser predicts several things in relation to buffer timings and lexical access. It predicts that all meanings are accessed at once and for how long the multiple meanings will be around. If ROBIE is to match two buffers at once, then some words will retain their multiple meanings until the next word has been perceived.

For example in handling the word 'to', I have shown that the correct approach is to check the word following it to see if it is a tenseless verb or is something that can start a noun group. This means that 'to' will not be disambiguated until the next word has been perceived. Since the parser does not have the Attention Shift and words are not allowed to be placed on the Active Node Stack unless they are dominated by a non-terminal, the word 'to' must be disambiguated when the following word has been perceived. The parser has many similar examples, so in many cases a word will not be disambiguated until the next word is perceived.

Not all examples of ambiguity need both buffers to be filled before the word is disambiguated. For example, the first word following a determiner that could be a noun, will be attached as a noun and the second buffer does not need to be filled. It should be noted that the grammar may change as it is altered and expanded, so a rule that needed two buffers now, may be re-formulated in the future. This makes it very hard to say exactly which rules make the two buffer prediction.

Let us look at the implications of the two buffer prediction and see how well they fit the psychological data.

6.7 Lexical Access

We will now look at the work done on Lexical Access. The research in this area has been trying to find the answer to the question: "When a word is first perceived, how is it looked up in the human "dictionary". Are all its possible meanings returned, or just the appropriate meaning?"
There are two main schools of thought on the lexical access question. Any time the prior syntactic and semantic context will affect the result returned by the lexicon, these two schools of thought make different predictions. The Prior Decision Hypothesis predicts that in a heavy biasing context only the appropriate meaning would be returned, while the Post Decision Hypothesis would predict that all meanings are accessed in all situations, but the non-appropriate meanings are eliminated soon afterwards.

6.7.1 Swinney, Cairns and Kamerman

A supporter of the Post Decision Hypothesis is Swinney. He performed two experiments to test this hypothesis using a cross modality task. In the first experiment, target words were presented simultaneously with the occurrence of an ambiguous word. The results of this experiment showed that words related to both meanings of the ambiguity were facilitated in both a weak and strong biasing contexts relative to unrelated words, whether or not the related meaning was consistent with the biasing context.

In the second experiment, he presented the target word later in time. In this experiment, the test phonemes were presented three syllables after the test word (about 650 msec). This experiment showed that after the three syllables had passed, only the appropriate meaning was facilitated. This then provides evidence that all meanings are accessed at once, but after about 650 msec, only the appropriate meaning remains. His conclusion was "immediately following the occurrence of an ambiguous word, all meanings for that word seem to be momentarily accessed during sentence comprehension" [Swinney 1979, p. 653]

Given that a multi-syllable word is recognised in its first syllable [Marslen-Wilson 1980b], then the next word will have been figured out by the third following syllable. In this case, the two buffers will be filled and the disambi-
guating process can happen, as the word will have been fully parsed. Therefore Swinney's experiments support the compound lexical entry. However, since all the ambiguities in Swinney's examples were within the same part of speech, we cannot judge whether this experiment has any bearing on the two buffer lookahead.

Cairns and Kamerman [Cairns and Kamerman 1975] further investigated this hypothesis. They were, again, trying to decide whether one meaning is accessed in context, or all possible meanings. They tested this in two different ways, a phoneme monitoring experiment and a sentence completion experiment.

They presented two hypothesis: The Short Term hypothesis says that all lexical information is retrieved and stored in working memory, probably until clause end and the Immediate Memory hypothesis says that the lexical decision is made immediately after the retrieval of information and only one meaning is carried on. Both of these fall under the Post Decision Hypothesis. They tested these two hypotheses with the phoneme monitoring experiment.

They concluded that, 'The results of the phoneme monitoring experiment support the immediate decision hypothesis rather than the short term hypothesis.' They said "the process which produces increased monitor latency following an ambiguous lexical item is completed roughly two words later in the sentence". Again these results support the use of compound lexical entries, but all their examples were within the same part of speech, so we cannot judge whether their results have any bearing on the two buffer lookahead.

6.7.2 Tanenhause, Leiman and Seidenberg

[Tanenhause, Leiman and Seidenberg 1979], investigated the lexical access question from a slightly different viewpoint from Swinney. The test words in all of Swinney's examples always had the same part of speech. In the work of the above authors, lexical ambiguity was investigated in syntactically biasing contexts where
the part of speech of the word was actually different in the different examples. Swinney's work was not as interesting to us because there was no syntactic ambiguity, but this work may have a bearing on our buffer timing question. They tested words such as 'watch' in the following examples:

[319] I bought the watch.
[320] I will watch.

This was done using a variable delay naming paradigm to see which meanings of a word were facilitated at various times after its occurrence. The subject heard the sentence over headphones. At the end of the sentence, the test word (for example 'look') was presented on the screen and the subject pronounced the word out loud. If the word was facilitated, they predicted that reading it would be faster than an unrelated word. The time delays were at 0 msec, 200 msec, and 600 msec. They predicted, as ROBIE would, that all meanings were accessed at 0 msec, but by 600 msec, only the contextually appropriate meaning would be facilitated.

Their experiment showed that all meanings were facilitated at the 0 msec time, but at 200 msec and 600 msec, only the relevant meaning was facilitated. This is consistent with the results obtained by Swinney. They concluded: 'Both noun and verb readings of the ambiguous word were initially accessed with the appropriate reading selected within 200 msec on the basis of syntactic context'.

This is the result ROBIE would predict. All possible readings are looked up initially and when the word is disambiguated, only the relevant meaning remains. All of their examples were biased by the syntax in such a way that there was no need to see the word following the target word to disambiguate it. All target words were also at the end of the sentence, so there were no words following it. For example in [319], the word "watch" was immediately after the determiner, so it had to be a headnoun. Similarly, in [320], the word followed the modal, so lookahead would not be used to resolve the ambiguity. Therefore, ROBIE would disambiguate all of these examples as soon as the word was recognised.
We can now decide if their experiment had anything to say about the lookahead predictions. They showed that all meanings were facilitated at 0 msecs but the facilitation is over 200 msecs after the onset of the target word. We know that it takes 200 msec to identify a word. [Marslen-Wilson 1980b] Therefore the disambiguation has taken place as soon as the target word was identified and without using the next word. This implies that the disambiguation took place before the second buffer could have been filled. This seems to contradict the lookahead prediction, but, as I stated above, all the examples they used can be resolved using the techniques in Chapter 5. In that chapter, we saw that the rules to resolve these cases do not use the second buffer. Since the lookahead is not used, the word can be disambiguated as soon as it is identified. This is exactly what their results suggest. Unfortunately this means that their experiment has nothing to say about our use of lookahead. To resolve the lookahead question with this approach, further experiments are necessary.

An example of the type of experiment which needs to be performed is as follows. In ROBIE, both meanings (auxiliary verb and preposition) of the word "to" would be available until the next word has been identified. I explained in Chapter 3 that this ambiguity is not resolved until the features of the second buffer have been checked. The ambiguity of the word "that" between a complementiser and a relative pronoun will remain until the features of the next word have been checked in the following sentences:

[321] I know that boy did it.
[322] I know that boys did it.

We have also seen (experiment two, Section 4.8) that the fragment "will be" can lead to a garden path. This is based on the fact that lookahead is used to disambiguate the word "will". It is also predicted that all meanings for "will" are present until the next word has been perceived. That is, all meanings of "will" are present until the next word ("be" or "was") is recognised in the following sentences:
We have investigated these works to see if there is any experimental evidence that would support or reject the predictions of the two buffer lookahead. While the predictions from ROBIE are compatible with the data presented here, no conclusions can be drawn as there is no data on the key situations where disambiguation rests on the following word.

6.8 Learning the Deterministic Parser

Many psychologists believe that if a parser is to be psychologically plausible, the grammar must be learnable. I have not investigated this, but Berwick [Berwick 1979] has. It is his opinion [personal communications] that determinism is definitely a boon to acquisition for this reason: during a deterministic parse, if the parse fails, the parser knows it was correct up until that point. One can then build a rule in the style of [Berwick 1979,81] that will fix this situation.

When a non-deterministic parser fails, it could be in two situations. Firstly it may have failed as above and a rule needs to be added to fix the situation, or secondly, it failed because it was pursing the wrong branch of the non-deterministic computation. If it had failed for the latter reason, then backtracking is used and a rule should not be added. When a non-deterministic parser fails, it is not possible to tell which of the above cases apply. Hence learning is much more difficult.

Berwick has shown that it is possible to learn most (70%) of the "core-grammar" of Marcus. The part that his system could not learn was the diagnostics. I have eliminated these, so the learning should be simpler. Unfortunately, Berwick did not investigate how to learn number agreement which is vital to our approach.

In summary, learning is not easy, but it is certainly easier to learn a grammar when it is being used by a deterministic parser.
6.9 Timing of the Parser

In Section 6.3, Kaplan stated that the parse time should be at most a slowly increasing function of sentence length. In this section we will see whether this is true for ROBIE.

 Parsing time was collected on the parses of 130 different sentences. These times were collected on an PDP-10 timesharing computer. Because of the timesharing, there is an unknown amount of time spent during the parsing of each sentence on system and PROLOG overheads. Whilst this means that the times are slightly misleading, the time does give a good indication of the total effort required by the parser and the resulting times are very consistent. Based on this, ROBIE's speed was 50 msecs per word (.05 sec per word) plus 100 msec per sentence of overhead. This includes the syntactic and non-syntactic analysis. The analysis presented here was purely informal and no special statistic tests were used.

The sentences tested were from two groups. Neither group contained garden path sentences, as they cause the parser to fail and hence do not provide parse times. First were the MECHO examples in Appendix B. The mean number of words in this group was 14. The shortest sentence was 3 words and the longest was 32 words. The average over 60 sentences was 51.72 msec per word with a standard deviation of 11.18 msec plus the 100 msec overhead. The time to parse a sentence grows roughly linearly in proportion to the number of words, although it must depend on the complexity of the sentence as well. For example, the mean time taken to parse sentences, of length 12 words, was 47.67 msec per word, with a standard deviation of 6.5 msecs. To parse sentences of length 17 words, the mean was 57.33 msecs per word, with a standard deviation of 7 msecs. This shows the deviation due to complexity and the growth in length. It is not the case that all examples over 17 words average longer than 57 msec/word to parse. In fact, most do not.
Below is a graph of the above data. The vertical axis is the time taken in msec, per word in the sentence. This is computed after the 100 msec overhead is removed. The horizontal axis is the number of words in the sentence.

Graph to show Parse Time Per Word against Number of Words in a Sentence for examples from the MECHO problems.

I also collected timings for 70 examples from Marcus and Church [Marcus 1980], [Church 1980]. The average length of these sentences was 7 words, the shortest having three words and the longest 11 words. Average parse times for these examples was 48.2 msec per word with a standard deviation of 17 msec. These examples were chosen for complexity and linguistic coverage. For sentences of length 6 words, the mean time was 45 msec and the deviation was 10 msec. Therefore the timing is roughly 50 msec per word.

Below is a graph of these examples. The vertical axis is the time taken in msec, per word in the sentence. This is computed after the 100 msec overhead is removed. The horizontal axis is the number of words in the sentence. One can see that the parser time grows
slowly as a function of the number of words in the sentence.

Graph to show Parse Time Per Word against Number of Words in a Sentence for examples from Marcus and Church.

parse time in msecs/word

Figure 2: Examples from Marcus and Church

6.10 Summary

This is the end of our survey of related work in psychology. As the reader can judge for himself, the current parser does very well when compared in this way. We have shown that it can easily account for the principles of Minimal Association, and Right Association, the Seven Principles of Kimball, and the relevant lexical access literature. The examples presented here do not prove that ROBIE is a psychological model of the HSPM, but I feel they show that it has many strengths and is better than other existing models.

In summary the most important psychological elements are: definitions return all possible features, there are no special ambiguity handling rules, automatic disambiguation, the non-syntactic processor's interaction with the syntactic processor, the production system grammar, and only two buffers.
This now completes the development of our model of normal processing in the
HSPM. This model consists of two processors, a syntactic processor and a non-syntactic
processor. We have seen that conceptually the two processes work in parallel during the pro-
cessing of a normal sentence. During processing of some sentences, the syntactic processor,
will at key points, ask the non-syntactic processor to make a decision in order to resolve an
ambiguity. We have seen that these key points are those situations which the syntactic pro-
cessor cannot guarantee to resolve correctly with its two buffer lookahead.

The rest of this thesis is devoted to further details and areas for further investiga-
tion. The next chapter will describe other approaches to parsing and Chapter 8 will provide
more details of how ROBIE works. Some readers may wish to skip these two chapters and
continue with the discussion of problems for future work investigation including 'Have' and
global ambiguity in Chapter 9.
Related Work

In this chapter, we will look at other approaches to parsing. The work described in this chapter has had some influence on my own work, even though the influences of some works is very slight. These works are presented to enable the reader to gain a perspective on my work in relation to other works.

7.1 Deterministic Parsers

7.1.1 Resolving Noun/Verb Ambiguity

The first attempt to handle part of speech ambiguity in a deterministic parser was by [Milne 1978]. This paper concerned itself with noun/verb ambiguity only. Certain words were defined as both a noun and a verb in the dictionary and these two definitions were returned under a single node. Marcus’s parser was also modified so that ATTACH disambiguated the words as has been done in ROBIE. This approach used a function called "MAKE" which could explicitly disambiguate a word in the grammar, providing an explicit means of disambiguation and words were often disambiguated before they were attached. Finally, special rules, similar to Marcus’s ‘Diagnostics’, were added to handle specific cases.

The approach was effective and these strategies handled a very large number of cases. When a word which could be both a noun and a verb arrived into a buffer, the parser would Attention Shift and activate a special packet of rules for disambiguating the word.

The grammar then had special rules to handle the possible cases. For example to handle:

[325] I want to kiss you.

The parser had a rule with the pattern: [to][noun/verb] and the action: make 2nd a verb. This rule would make any noun/verb ambiguous word following “to” into a verb. These rules were simulated on free text and had an extremely high success rate. Even
though this method was very effective, it had a major deficiency as well.

The objection to this approach is that it used very special purpose rules to handle ambiguity. For each case of ambiguity, there was a special rule to decide what to do. The desire for a more elegant method of handling ambiguity motivated most of the work in Chapter 5. This approach is completely surpassed by the work in this thesis.

7.1.2 Church's YAP

Church [Church 1980] designed a parser called YAP (for Yet Another Parser). This is a deterministic parser in the form of a finite state machine. The main focus of Church's work was limiting the memory which the parser can use and, hence, keeping the number of possible internal states finite. It is possible to avoid backtracking by having infinitely many states, one for each possible step of each possible path. If there are an infinite number of states, it is not possible to determine the unique successor configuration from a given state and, hence, the system would not be deterministic. Church's work limited the number of states to avoid this problem.

Church's YAP, was designed using the features of Marcus's parser. Like mine, it is very similar in design to Marcus' parser. Church, however, added a few essential differences. YAP's central feature is the "= WALL = " . This Wall divided the upper and lower buffers in the parser. Below the Wall were the three lookahead buffers. Church called these "down1, down2 and down3". Above the Wall, Church had an "upper buffer". This replaces the Active Node Stack used both in Marcus's parser and in ROBIE. Both buffers grow towards the Wall. The upper buffer built constituents downward, that is mothers looking for daughters. The lower buffers built constituents upward, that is daughters looking for mothers. Here is a snapshot of Church's parser: [Church 1980, p. 44]
In place of the production system grammar of Marcus and Milne, Church has a
'deterministic finite state control device'. Remember that Marcus could match a rule from
any of the three buffers, plus the Active Node Stack, plus the lowest S node being built.
Church could test the features of all three lookahead buffers (down1, down2 and down3). He
could also test the features of all three upper buffers (up1, up2 and up3). This means that,
where Marcus could check five items, Church could check six items, while ROBIE checks
only two.

Church did not use packets, but instead used dotted rules on the constituents in
up1. This approach is essentially isomorphic to the packets, but eliminates the need for
explicit packet control. See Section 8.4.1 for an explanation of the dotted rules.

Nor does Church use the Attention Shift as used by Marcus. In YAP, the Wall is
moved down by one node to Attention Shift. This has the effect of creating a new buffer, as
in Marcus's approach. When Church's parser recovered from an Attention Shift, it shifted
the Wall back up one node, thus restoring the buffer to its original position and the newly
built item was then in the second buffer. This is very similar to my non-AS approach, the
only difference being the method by which the total nodes are counted. To avoid the Attention
Shift ROBIE may create another node on the Active Node Stack. This has the same
effect as shifting the Wall. Church's parser is similar to this in that it has a static number
of lookahead buffers. In ROBIE, it is always the next two nodes from the bottom of the
Active Node Stack. In his parser it is always the next three nodes below the Wall.

There is one final difference between the Attention Shift of Marcus, and Church's
Wall movement and ROBIE. In these two parsers, the Attention Shift built a new node,
then returned to the state previous to the Attention Shift, but with the new node fully constructed. This is very similar to a SEEK in an ATN parser. In YAP, it is not possible to attach nodes to the old Active Node Stack while in an Attention Shift because the Wall will be in the way. PARSIFAL never attached items to any old node in the Active Node Stack while in an Attention Shift. Both of these parsers could Attention Shift at any time.

In ROBIE, it is possible to attach items to the Active Node Stack while "movement" is taking place. Because of this, the Active Node Stack may not remain unchanged as above, so the similarity to the SEEK arc and Attention Shift does not hold. (See Section 8.5.3)

Church's grammar only had a few basic functions, as ROBIE does. However, his grammar could also execute arbitrary LISP code, making it very powerful. His parser grammar is based on the linguistic analysis of Bresnan [Bresnan 1978].

Church used diagnostics similar to those used by Marcus as discussed in Chapter 5, but harder to read and more complex ones. He had a very good method of constraint passing, which handled long distance number agreement, etc. Although this approach is very interesting, it required several complex extensions to the parser. Whether it can meet ROBIE's desire for simplicity is not known.

Church also investigated the problems of Right Association and Minimal Attachment, Closure, Conjunction and Gapping. In these areas he resolved many issues not considered in this thesis.

7.2 The ATN

The most widely used type of parser today is based on the Augmented Transition Network (ATN). This system, developed by Woods [Woods 1970] can be explained thus:

"Essentially, a transition network grammar is a finite-state transition diagram which has been generalised to a pushdown store automaton by adding a recursion mechanism and then further generalised (up to the power of a Turing Machine) by the addition to the machine configuration of a set of registers which can hold arbitrary pieces of tree structure and by the addition
of arbitrary conditions and actions which can set and test these registers on the arcs of the network. [Woods 1973, p.112]

In the formalism presented by [Woods 1973], the transition network grammar has 5 basic types of arc. These are CAT, TST, JUMP, POP and PUSH.

In CAT the arc can be taken, if the word being scanned has the CAT specification. When one of these arcs is traversed, the input word is "consumed"

TST is like CAT, except that, rather than a simple category test, there can be arbitrary conditions on the arc. One can see that the CAT arc is just a subset of this.

JUMP causes a transition from one state to the next without altering the input string.

POP is a pseudo arc, used to signal that a given state is finished and the system returns to the last PUSH.

PUSH (or SEEK) arcs provide the basic mechanism for recursion. When a PUSH arc is executed the current state is remembered and the system pushes to another network to build a constituent. This is the top down part of the system.

In ROBIE, the above tests are not allowed and there is no equivalent of the TST arc or the PUSH arc.

A transition network grammar with the features we have described is a "basic transition network" (BTN). Woods says that, although this is a good way to describe context free languages, the real advantage of the transition network grammar comes from the addition of arbitrary register setting actions and arbitrary conditions on the arcs. Adding these capabilities results in an "Augmented Transition Network grammar" (ATN). This has the power of a Turing machine if the use of these arbitrary conditions is unconstrained.

A Recursive Transition Network (RTN) does not have the arbitrary conditions and tests described above, and extends the BTN by adding recursive calls to the subnetworks.

A parser using an ATN grammar is generally a top-down analyzer. It has the disadvantage that the same low-level phrase may be analysed many times in the same way.
In order to reduce this need to re-parse the same item many times, the standard solution is to keep a Well-Formed Substring Table (WFST). In the Woods system, this was implemented by examining the table whenever a PUSH was encountered. If there was an earlier PUSH to the same state in an equivalent configuration, then the structure previously built was POPed and the PUSH was not needed.

An ATN as proposed by Woods, could handle part of speech ambiguity effectively by ignoring it. If it assumed a word was a noun, and later discovered an error, then it could backtrack and try again. Since backtracking is not allowed in a deterministic parser, this solution is unacceptable to my work.

The objection to the Turing Machine power is that it is desirable to have:

"a grammar model that is just powerful enough to do the job of characterising natural languages (so that the limitations of the machine can be interpreted as hypotheses about language). Clearly, a model with the power of a Turing machine and therefore the capability of doing anything that can be effectively characterised, fails to meet this specification." [Woods 1973, p. 124]

Woods, then, explains that the ATN must be constrained so that it does not have this power. He constrains it this way: 'There can be no cycles of all JUMP arcs and the arbitrary conditions must be computable.' These limitations are necessary to ensure that the parse will terminate.

He also says that the number of registers should not be constrained in the parser and that the grammar designer should be able to invent a new register. Adding registers could avoid determinism and so these are not relevant to ROBIE. The WFST described above is computationally very expensive. If there are a small number of alternatives, then it is more expensive to use the WFST than to backtrack. If there is a lot of backtracking, then it is worthwhile.

Woods used the Selective Modifier Placement facility (SMP) to provide semantic guidance in parsing. This enables him to make a semantic check before attaching a modifier to a constituent. This will be explained in more detail in Chapter 10.
He also has a semantic interpretation at each POP arc. If the interpretation fails, then the item built is rejected. Woods argues that semantic analysis is currently much more expensive than syntax, so it is better not to rely on the former. This point is relevant to implementing parsers (engineering), but as we are making assumptions about how semantic interpretation happens in people (psychological plausibility), we cannot know if this argument is valid. If the semantic representation is as trivial to update and check as the syntax in the HSPM, this argument will disappear. However it seems that, according to current knowledge, the semantic interpretation is far more expensive than syntax, so it is best not to use it too interactively.

7.3 Chart Parsers

The next type of parser we will look at is the Chart parser. Whilst the Chart parser can handle ambiguous constructions very well, it was not designed to be psychologically plausible. Hence, it is only partially relevant to our discussion.

We will use the MIND system designed by Kay [Kay 1973] as an example of Chart parsers. This system has been designed as a "tester of transformational grammars". It is very flexible and the user has many interactive facilities with which to change any rule in the grammar and observe the effects on the system.

The system consists of a single common data region called the "chart" and a master scheduler which decides when to call other programs into operation. All programs operate on the chart.

The chart is a directed graph with labeled "vertices" and "edges". Each node of the graph is a vertex, between each of which are one or more edges. The edges connect nodes into larger nodes, denoting the larger node's grammatical category. Several edges between the same two vertices can be considered mutually exclusive alternatives. Once an edge is added to the chart, it is never removed or changed. This avoids the backtracking issue, but we will look at the implications of this soon.
The system is not strictly designed to be left to right and, in fact, can work right to left if the user so desires. It is also possible to switch between top down and bottom up approaches quite easily. When there is an ambiguity in the parser, an edge is added for each interpretation of the ambiguity. The labels on edges can be very complex.

The parser will connect vertices with edges properly labeled, as it tries to apply each rule to the chart. At the end of all the applications, the chart will have edges for all possible readings, no matter how many are possible or how obscure they are.

For any sentence then, the scheduler will apply the rules of the grammar to the chart and create all possible interpretations. This has two undesirable properties from the point of view of psychological plausibility.

First, all readings are always found with the same effort. Even readings which people must search to find are presented immediately. The parser could be modified to follow the semantic timing which we use, but Kay did not do this.

Secondly, the parser will never garden path. Since it always builds all possible structures, the parser will never backtrack or make an error. Hence there is no reason for it to backtrack or fail on a garden path sentence. We know that people fail on garden path sentences, so the Chart must be rejected as a psychological model.

The Chart never backtracks, because it builds all possible edges when there is an ambiguity. This violates our principle of 'no wasted structure'. For example, in the sentence:

[326] The horse raced past the old barn.

the chart will have built both interpretations of 'raced' by the time the final punctuation is added. At this point, one analysis will be thrown away. This is exactly the wasting of edges the above principle prohibits.

In the MIND system, ambiguities were resolved with the 'Disambiguator, an interactive component which asked the user what to do with each ambiguity.' Kay believed this could be done automatically, but he presents only the interactive approach.
Although it is superficially very simple, it requires an enormous control structure in the background. The great benefit of using the chart is that it produces all interpretations in parallel. In handling global ambiguity, (which happens to be the only examples Kay demonstrates) it is ideal. All meanings are available simultaneously. For this reason the chart parser is important.

[Martin, Church, and Patil 1981] have built a large, efficient chart parser which is designed to return all possible parses. As the main emphasis in this work has been on pure parsing, it is not directly relevant here. However, this system has shown the large number of possible parses to be found if all parses are sought. This helps to demonstrate that the chart is a useful method for parsing although it does not meet the criteria laid down in this thesis.

7.4 A General Syntactic Processor

Kaplan [Kaplan 1973] explains that the key principles of both Wood's ATN and Kay's Chart can be encompassed under the General Syntactic Parser (GSP).

This is an abstract machine which maps strings of trees to strings of trees. GSP uses a chart and has four variables indicating the current edge, its tail and assorted information about it. Kaplan shows that the chart can be implemented with only a few, very primitive constructions and, using grammar compilers, many grammars can be implemented.

This system was presented as a general super parser which by adjusting a few parameters could be made to model an ATN or a Chart. In order to accomplish this, many special global variables and functions are added to the system. Whilst this system is interesting, it has had no influence on the work described here. It is presented to illustrate that there is a similarity between the ATN and the Chart parser. Whether GSP could model a deterministic parser is not known.
Steedman and Ades [Steedman and Ades 1980]. [Ades and Steedman 1980] have proposed a psychological parsing model consisting of a stack and five simple grammar rules which they have demonstrated on a few simple examples.

Their categories are of the form \(X/Y\) which means "an X lacking a Y" [Ades and Steedman 1980, p.14]. For example, \(S/NP\) means an S node lacking a NP. Their rules provide an automatic way of combining items of class Y with a node of type \(X/Y\). They also explain the "stacking constraint". This states that movement in sentences happens as if the items being moved were stored on a push-down stack. Their parser consists of a single stack on the top of which new items arrive. Sentences are parsed by combining items until there is only one item left.

They claim that their simple rules and the stack nature of the parser account for the movement phenomenon in English and several syntactic constraints. This parser is attractive because of its simplicity, even though it has been tested on a very small grammar. The main psychological attractions are the simple grammar rules and the stack.

With a few extensions, this stack parser is almost isomorphic to the parser we have seen here. Our Active Node Stack is equivalent to their stack, with the two buffers being the top items of the stack. All their rules combine the top two items of the stack. Because ROBIE has two static buffers, the second and third items of the stack are usually combined.

Here is a comparison when parsing the sentence "I will marry her" at the time the NP "her" is about to be added to the VP. On the left is the state of the Steedman and Ades Parser, on the right, ROBIE.

<table>
<thead>
<tr>
<th>Steedman and Ades</th>
<th>ROBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell3: NP - her</td>
<td>B2:</td>
</tr>
<tr>
<td>cell2: VP/NP - marry</td>
<td>B1: NP - her &lt; packet&gt;</td>
</tr>
<tr>
<td>cell1: S/VP - I will</td>
<td>ANS: VP - marry</td>
</tr>
<tr>
<td></td>
<td>S - I will SS-VP</td>
</tr>
</tbody>
</table>

168
Notice that ROBIE has an extra stack cell, the 2nd Buffer. After the next step, the two states will be:

Steedman and Ades  
ROBIE

B2:

cell2: VP - marry her  
B1: VP - marry her

cell1: S/VP - I will  
ANS: S - I will SS-VP

One difference between the parsers is that ROBIE can combine nodes on other than the top of the stack. e.g. It can combine the two buffers. This is rarely necessary and might be eliminated in a different grammar. Steedman and Ades have not committed themselves to a deterministic or non-deterministic parser. I feel that if items were combined between the second and third stack cells, rather than the top two, their parser would be very similar to mine.

The grammar for ROBIE was based on Chomsky’s [Chomsky 1973, 76, 77] EST theory. There is no simple way to automatically combine rules using the feature of EST as is possible with complex categories. In several sections, we have implied that PSG [Gazdar 1980] is better for our purposes than EST. I feel it would also be beneficial here. This method is very interesting, but needs more development.

7.6 Other Approaches to Parsing

There are many other popular approaches to parsing. While knowledge of the following systems has influenced my thinking in some way, none of the following systems were designed to be psychologically plausible and had no major influence on this work.

a) Semantic grammars as proposed by [Burton 1976]. This approach utilises an ATN parser, but instead of using syntax, it uses semantic networks. While this method works well in a limited query system, it is not easily transportable to other areas.

b) Definite Clause Grammars as explained by [Colmeraur 1978], [Pereira and Warren 1980, 80b]. These parsers have been implemented in PROLOG and are very efficient.
Using the backtracking facilities of PROLOG, small but impressive systems can be built rapidly. These parsers have been compared with the ATN [Pereira and Warren 1980] and shown to be as efficient and conceptually cleaner.

c) Conceptual parsing as done by [Schank 1973]. This approach is very interesting but does not use any form of syntax. This work is not related to the syntactic processor, but rather the non-syntactic processor.

There are several other top-down, depth first search algorithms for non-deterministic parsing. None of these are have had more than historical influence on my work. This includes: [Winograd 1972], [Sager 1973], and [Charniak 1973].

7.7 The Timing of Semantic Interaction

In Chapter 4, we saw when non-syntactic information was extracted by ROBIE during the parse of a sentence and when non-syntactic information was used to make decisions. We noted that each rule makes a contribution to the semantic interpretation of the parse. The non-syntactic processor can stop the parse if the contribution makes no sense. At points where the syntactic processor, with limited lookahead, finds two rules of equal constraint matching, non-syntactic information is used to decide which rule to run. This happens at the end of a NP and in examples of global ambiguity.

In the following sections, other major approaches to the timing of semantic interaction timing be discussed. We will look at four other approaches to the timing of semantic interaction. These are: at the end of the sentence, at the end of each clause, and two approaches to continuous interpretation. For reasons previously stated, we will not look at parsing methods which do not involve syntax such as [Wilks 1975] and [Schank 1973].

For the sake of comparison, we can look at two well known systems and two newer proposals and see when non-syntactic information is used. These are Woods LUNAR system [Woods 1972], Winograd's SHRDLU [Winograd 1971,72], Mellish's 'incremental evaluation' [Mellish 1981] and Marslen-Wilson's on-line approach. [Tyler and Marslen-
7.7.1 At the End of the Sentence

In LUNAR, the semantic analysis did not start until the entire sentence had been parsed. If the semantic analysis failed to work, then the parser backtracked and attempted to find another parse. This next parse was then passed on to the semantic interpretation component to see if it was acceptable.

Whilst this approach was important historically, it was not psychologically valid [Marslen-Wilson 1980]. It implies that a person had no knowledge about the meaning of the sentence until it was finished. This is obviously not true. The second problem with this approach was the lack of feedback to the syntactic component from the semantic component. When the semantic component rejected a sentence, it has no way to tell the syntactic component what was semantically unacceptable. Instead the parser had to backtrack to find another parse.

Riesbeck [Riesbeck 1975] has objected to this approach, saying:

"Why should consideration of the meaning of a sentence have to depend upon the successful syntactic analysis of that sentence? This is certainly not a restriction that applies to people. Why should computer programs be more limited?"

As was described in Chapter 4, each of ROBIE's grammar rules make a syntactic contribution to the sentence as well as a contribution to its semantic interpretation. In this way, ROBIE has a partial semantic interpretation of the sentence as the parse proceeds. Therefore, the consideration of the meaning of a sentence does not depend upon a successful syntactic analysis of that sentence.

7.7.2 At the End of the Clause

Winograd's SHRDLU [Winograd 1971, 76] called the semantic interpretation component at two points during the parse. First, after the head noun had been found and
secondly, after all the objects had been located. This represented calling the semantic interpretation at the end of each clause and this system was more effective than the LUNAR approach. Ritchie [Ritchie 1976] criticises this timing of non-syntactic processing and demonstrates that it is not adequate. Mellish [Mellish 1981] echoes this criticism.

Mellish points out that syntactic ambiguity is very common, and often the only way to resolve this ambiguity is to make use of non-syntactic information. If non-syntactic interpretation is only done at the end of the clause, then non-syntactic information cannot be used to resolve ambiguities inside the clause. Similarly, if non-syntactic interpretation is only performed at the end of the sentence, it could not be used to resolve ambiguities inside the sentence. In Chapter 4, we have seen that non-syntactic information is used to resolve syntactic ambiguities at several points in the sentence. This is not possible using either of these approaches to semantic timing.

7.7.3 Incremental Evaluation

Mellish [Mellish 1981] presents an approach to semantic interpretation called "incremental evaluation". He recognises that as each word is parsed, information is gained about the sentence, although this information may be only partial. As each word is parsed, this partial information is incremented, refining the meaning of a constituent. When enough information is present to gain the "full meaning" of the constituent, further semantic processing takes place. This theory is implemented in a parser for the MECHO system [Bundy, et. al. 1979b].

We are not concerned with the details of Mellish's semantic processing, only the timing. His approach is similar to the approach I have outlined in this thesis. Each grammar rule is selected on a syntactic basis and the rule makes a contribution to the semantic interpretation of the sentence. His program demonstrates the viability of this approach.

The following theory is very similar, but is not based on a working computer program. The authors of the previous three approaches have not claimed that their approaches
are psychologically plausible. Rather they have addressed specific issues of computer text processing. The final theory we will consider does not have a computer model, but is claimed to be psychologically plausible.

7.7.4 The On-Line Theory

The last major semantic timing approach we will consider is the 'on-line' theory. This is "an interactive model of sentence processing in which syntactic parsing and semantic composition interact on-line to assign both syntactic and semantic analysis." In this theory, the syntactic processor and semantic processor interact continuously during the parse, the semantic processor guiding the syntactic parse. Supporters of this theory include [Kimball 1973], [Marslen-Wilson 1973,75], [Tyler and Marslen-Wilson 1977], [Marslen-Wilson and Tyler 1980], and [Coker and Crain 1979].

The distinction between this approach and that which I have outlined is relatively fine. In the on-line theory, semantic information is used to make parsing decisions at each step, whilst in my approach, it is used only when the syntactic processor can't decide. It is proposed that semantic decisions are made for garden path situations [Coker and Crain 1979] and resolving some cases of global ambiguity [Tyler and Marslen-Wilson 1977].

The on-line theory does not distinguish which rules have a choice of alternatives, as the above cases do, and which rules have no choice. We saw in Chapter 4 that, for the majority of the rules in ROBIE, should the non-syntactic interpretation fail to make sense, there is no alternative but to fail the parse. In my explanation, this is overcome by having each rule make a contribution to the non-syntactic interpretation of the sentence. If this contribution makes no sense, then the rule will abort (depending on the mode).

In the on-line theory, the semantic interpretation must agree before the rule can run. In the context of ROBIE, this would mean running the same non-syntactic contribution that will be made by the rule, as part of the rule matching. In the on-line theory, the rule could not match if the semantic contribution made no sense. There is a fine distinction
between the semantic test that may be made to run a rule and the semantic contribution made by the rule. It must surely be more efficient to separate these. In the on-line approach, they have added an extra step of semantic interpretation, even though the next step may fail. In my theory, the rule would match and then the parse would fail. The timing difference between these approaches is so fine, that they can probably never be distinguished. Marslen-Wilson says: [Marslen-Wilson 1980b]

"Note that this (experiment) does not completely rule out the possibility that syntactic processing is autonomous, since it is possible to imagine a system in which the intermediate products of autonomous syntactic analysis are made immediately available to subsequent interpretative processes, and that it is the effects at this level the experiment is tapping. In fact, this alternate proposal can probably never be excluded on the basis of experimental data."

This means that these two explanations can probably not be distinguished experimentally.

Marslen-Wilson and Tyler have done several experiments to test this theory. The first is the speech-shadowing paradigm [Marslen-Wilson 1973,75]. In this experiment, the subjects were asked to repeat a sentence while they heard it (shadowing). They were then given sentences which contained errors and mispronounced words. In some cases the subjects would correct the error and restore the mispronounced word.

Marslen-Wilson interpreted these results as showing that the on-line theory was valid. The experiment involved speech comprehension, understanding and production. With so many major functions interacting, it is hard to be sure that the experiment was measuring an effect of comprehension. The results of this experiment are consistent with the theory I have explained and do not really help to distinguish the theories.

Tyler [Tyler and Marslen-Wilson 1977] tested the on-line theory for examples of the form:

[327] If you walk too near the runway, landing planes....
[328] If you've been trained as a pilot, landing planes....
If you watch them as they swoop down for the kill, hunting eagles...

Since it's forbidden by law, hunting eagles...

[Tyler and Marslen-Wilson 1977]

This experiment demonstrated convincingly that prior context influenced which reading was preferred for the ambiguous example. I will return to this experiment in Chapter 9 and give details of it in that chapter. These are particularly good examples on which to test this theory, since they can lead to global ambiguity.

In [Marslen-Wilson and Tyler 1980], two experiments are presented to investigate the on-line theory. Both of these used word-monitoring tasks in which the subject was told to react to a target word. Examples in the first experiment were of three forms; Normal Prose, Syntactic Prose, and Random Word order. For example:

The church was broken into last night.
Some thieves stole most of the lead off the roof.

The power was located into great water.
No buns puzzle some in the lead off the text.

Into was power water the great located.
Some the no puzzle buns in lead text the off.

[Marslen-Wilson and Tyler 1980]

In each of these, the target word is 'lead'. The sentences were presented over headphones and the subject was asked to respond to the test word. There were then three types of task. In the Identical task, the subject was told in advance which word to expect. In the Rhyme and Category tasks, the subject was told to look for a word that rhymed or was in the same category as the word. For example, in the Rhyme task, the word might be 'bread' and in the category task, "a kind of metal".

This experiment was designed to test word-recognition, as well as the on-line theory of semantic interaction. Marslen-Wilson's investigation is primarily focused on speech recognition. In this thesis, I have not investigated this and can make no comment on it. The speech recognition aspects of this experiment will be ignored. It is difficult to separate which phenomena are due to speech recognition and which to the parsing stage.

Marslen-Wilson has argued [Marslen-Wilson and Welsh 1978], [Marslen-Wilson 1980b] that semantic interaction has an important role to play in speech recognition. I feel
that there is nothing in this theory that is incompatible with the theories I have presented here.

The results of the experiment showed that the times for the Normal prose are faster than those for the Syntactic Prose, which are again faster than those for the Random Word Order. (372 mssecs Normal Prose, 407 mssecs Syntactic Prose, and 439 mssecs for Random Word order). Similarly, the times taken to perform the three tasks increased. These results were interpreted as supporting the theory that syntactic and semantic information is brought to play in spoken-word recognition.

Marslen-Wilson felt that this data supported the on-line theory. He postulated that the increasing times for the three tasks were due to the lack of effective semantic information to assist the parse at each step. The amount of work required to perform these tasks is not known and this experiment did not exclude the possibility that each task took an incrementally longer time than the previous one to perform. The experiment did not show how much processing must be performed before each task can be performed.

It is assumed that the Identical task can be responded to before phonetic analysis, the Rhyme task after phonetic analysis and before syntactic analysis and the Category task only after full analysis. The time difference for the three tasks would then show how long each stage of processing took. These assumptions are not checked by the experiment. It could be that the subject performed full analysis of the sentence before any of the tasks were performed. The longer reaction times would then show the relative difficulties of the tasks.

For example, if the Identical task could be performed after word recognition, but before syntactic analysis, this would mean that the word-recognition stage was totally independent of the syntactic stage. This is not necessarily true. It could be that full syntactic and semantic analysis is obligatory. This being the case, all three tasks will be responded to after the sentence has been fully processed. Again, the experiment does not preclude either choice.

The main result of interest is that the time taken to perform each task increased with the increasing complexity of the sentence. The relative time differences within the
Marslen-Wilson felt that the increase in reaction time for the three types of sentences was due to lack of syntactic and semantic information to assist these decisions.

My explanation for the increasing times is as follows. In Normal Prose, the speech recognition, parsing and response to the task processes are all normal. For Syntactic Prose, the semantic interpretation fails on a rule by rule basis as has been explained. Marslen-Wilson assumed that the sentences were parsed syntactically, so the semantic processor would have to be in the mode where even items that made no sense were accepted. It is fair to assume that, for the semantic processor to decide the item made no sense and then agree to continue, more time will be needed than for it to decide the word makes sense and continue. This is assumed because I believe that the semantic processor will do its best to make sense of the item before it gives up, hence taking more time. Since this is so, Syntactic Prose should take longer, because the semantic processor will be continually trying to make sense out of nonsense. Marslen-Wilson assumed the tasks took place before the full analysis was finished. If we assume that full syntactic and semantic processing is obligatory and that it is not possible to respond to any of the three tasks until after the sentence has been fully processed, then for Syntactic Prose the semantic interpretation will not be totally coherent, so the responses to the tasks will take longer to perform.

For Random word order, the syntax and the semantics will be continually failing and, hence, very time consuming, so a longer reaction time is predicted on this basis. If the tasks are based on the results of the analysis, then the semantic interpretation will not be coherent enough to make decisions.

As a final complication, the examples of Syntactic Prose are supposed to be syntactically, but not semantically, well-formed. Sentence [302] is not syntactically well-formed. The main verb of the test sentence is "puzzle" and it has two PPs, one with "in" and one with "off" in addition to the quantifier "some". If it is truly Syntactic Prose, then it should be possible to substitute modifiers in this sentence, without changing the main verb, to make it well formed, assuming the the categorisation for modifiers depends on the main
verb, and changing the main verb also changes the acceptable modifiers. For [332], it is not possible to substitute other modifiers because "puzzle" does not take the two PPs as modifiers. i.e., the following is unacceptable.

[334] *The boys puzzle some in the class off the teacher.

Marslen-Wilson has indicated [pers. comm.], that many of the other examples used in the experiment also have the problem that they are not syntactically well-formed. Because of this, the increased reaction times may result from processing difficulties.

This experiment does show that semantic information has an effect on the processing of sentences, but I feel it does not preclude my explanation.

Crain and Coker [Coker and Crain 1979] tested the 'on-line' theory with the naming paradigm explained in Chapter 4. The examples they used to test this were the same reduced relative clause examples that we have discussed in Chapter 4. They concluded that semantic decisions are made for this case. They then generalise over all of language and say that this shows it applies to all rules in the parser.

They did not distinguish as I have, between rules that have a choice to make and rules that do not. This is in common with the error of Tyler and Marslen-Wilson. Clark [Clark 1973] called this the 'The language-as-fixed-effect fallacy'. He pointed out the danger of testing a specific case of language and generalising over all of language.

These experiments have tested special cases, but not distinguished the cases where I have said that non-syntactic information does not play a role. As a result, their experiments do not refute my explanation and they support my theory of when interaction is used.

Hence, although the "on-line" theory is very attractive, I feel my theory helps to distinguish the two cases where the non-syntactic processor chooses between alternatives and when it has no alternative but to reject the sentence.
7.8 Linguistic Analysis

There are two major linguistic views which have influenced this work. The first is a transformational approach to linguistic analysis [Chomsky 1957, 65, 73, 75, 75b, 76, 77] and the second a non-transformational approach [Gazdar 1979, 80, 80b, 80c]. In this section, a brief explanation of the main points in these two theories which are relevant to this work will be presented.

7.8.1 Extended Standard Theory

The current parser uses the Extended Standard Theory (EST) analysis of sentences for three reasons. Firstly, this is the system I was taught when I learned linguistics, and hence the system with which I was most familiar. Secondly, it is the most widely used and accepted analysis today. Thirdly, Marcus had written his grammar in EST and I made only a few changes to his grammar. As the reader is assumed to be familiar with this system, it will not be described in detail. Rather, the major features of this system that we will compare in Gazdar's approach will be described.

The major area in EST which we are concerned with is the transformations. In EST all sentences are of the form S-> NP VP. In an embedded sentence, the subject NP may have been moved to a higher sentence (raised) as the subject of a higher sentence. For example, in the sentence:

[335] Mike seems to have left.

The underlying structure is:

[336] It seems that Mike has left.

In EST, it is assumed that the word 'Mike' has been raised from the lowest sentence to a higher sentence. Transformations have the overall effect of changing the word order between the surface structure and the deep structure.

Another example of a transformation is "auxiliary inversion". This transformation maps sentence [337] into sentence [338].

179
One can see the change in word order. The use of transformations greatly increases the number of possible sentences which can be generated from a grammar and greatly increases the parsing problem. We will return to this below.

Chomsky's analysis of the auxiliary, roughly, is

\[
\text{AUX} \rightarrow (\text{MODAL}) (\text{HAVE EN}) (\text{BE ING})
\]

This analysis is very descriptive and when the above transformation is taken into account, provides a good explanation of the English auxiliary system. This will be contrasted with Gazdar's analysis below. In EST, non-terminal symbols have no internal structure. They are merely the name of a node, e.g. S, NP, VP, PP, etc. It is assumed that the reader is familiar with the other details of EST.

7.8.2 Phrase Structure Grammar

Gazdar has proposed the theory of Phrase Structure Grammar (PSG). All movement rules, bounded and unbounded and all rules making reference to identity of indices in EST are removed. Gazdar's grammar is further restricted to be context-free. Gazdar argues that this yields two main advantages. Firstly, the class of grammars is restricted, enhancing learnability. Secondly, the resulting grammar can be parsed in a time which is proportional to the cube of the length of the sentence or less. This is not true for the recursive or recursively enumerable sets of grammars which include a transformational component.

There are several major differences between Gazdar's system and Chomsky's that are of importance here. The first is the lack of transformations. Whilst word order may change in EST, in PSG it does not. The above pairs of sentences are not considered to be related in the syntax. From a parsing viewpoint, this approach is very attractive, as a parser using this analysis would not need a mechanism to handle movement. This results in a conceptually simpler parser.
The second major difference is the category symbols. We noted above that category symbols in EST have no structure. In PSG, complex category symbols of the form S/NP are used meaning a S lacking a NP. This is the same as those used by [Steedman and Ades 1980]. These complex symbols provide many advantages for coordination, the treatment of EST movement phenomena and gap detection. In addition, nodes have features which are passed up and down trees according to the "Head Feature Convention". In this way, constraints are passed from one node to another. The Head Feature Convention is normally implemented bottom-up and it is not clear how it might be implemented in a top-down/bottom-up system.

Finally, the analysis of the English Auxiliary system is very different. Whilst Chomsky has a different analysis for the auxiliary verbs than the main verbs, they are no different in Gazdar's system. We have seen Chomsky's analysis above. In PSG, the auxiliary verbs are no different than other verbs. As a result, the trees for a sentence are very different between EST and PSG. See Section 9.3.4 for an example.

Gazdar captures many of the regularities in the grammar that Chomsky captured with transformations, with "Meta Rules". A Meta rule is a rule schema for producing new rules from existing rules. For each current rule in the grammar which matches the left side, it produces a new rule as described on the right side:

The AUX Inversion Meta rule [Gazdar 1980c, p. 46]:

\[ SAI: \begin{array}{c}
\left[ V^- V V^- \right], \ldots, \\
\left[ +FIN \right] [a] \\
\left[ +AUX \right] \\
\end{array}
\rightarrow
\begin{array}{c}
\left[ V^- V V^- \right], \ldots, \\
\left[ +FIN \right] [a] \\
\left[ +INV \right] [a] \\
\end{array} \]

(In this rule, V- roughly means a verbphrase. The feature [a] must be the same for both sides. The phrases will be initially marked +FINite and +AUXiliary. After the rule has run, the V- will be marked +INVerted.) The features +FIN and +AUX belong to the V- node. This Meta rule will produce a new rule as on the right from the rule on the left for all symbols 'a' in the grammar. These rules are not transformations, but merely a mapping of rules to rules.
In this chapter we have seen several alternative approaches to parsing and descriptions of several works which have influenced my work. At this point we have finished developing the model, and we have seen that it compares favorably with the literature. In the remaining chapters, we will see a few details of the parser for those readers interested in the workings of the parser.
The Parser

In Section 2.4, we saw in general how ROBIE differs from PARSIFAL and how it works. This chapter will provide an introduction to the parsing mechanism itself and allow better understanding of the claims made in the thesis. This chapter will also give the details of deterministic parsing in general and the implementation of the principles discussed in this thesis. This chapter is intended only for those readers which desire a more detailed description of the parser. Some readers may prefer to proceed to the following chapter.

There are several major differences between Marcus's PARSIFAL and ROBIE. These include: ROBIE's syntactic processor has the non-syntactic processor select whether to run a rule or not in certain key situations, ROBIE calls the semantic interpreter at each rule, ROBIE does not use the Attention Shift, ROBIE does not use any diagnostics, ROBIE attempts to match rules based on the patterns of only two buffers, versus three for PARSIFAL. Finally, only a few functions are allowed to be performed by the grammar rules, and certain features are transferred automatically by the attachment function.

8.1 Overview of the Parser

At this point, let us summarise a few important points with regard to the parser. It consists of an Active Node Stack and two buffers. Each grammar rule can match only the first two buffers. (The parser maintains three buffers solely for the conjunction rule, but the other rules do not match against the third buffer.) The parser does not use Marcus's Attention Shift. It does use a "Movement Stack" which is described in Section 8.5.3. The parser conceptually relies on non-syntactic information in the situations in which syntactic information may lead to an incorrect parse. These situations can be predicted with the use of the two buffer lookahead as we did in Chapter 4, but the current program does not detect them.
dynamically. Rather the relevant grammar rules have been programmed to call the non-syntactic check. Because the current parser does not have a complete semantic component, it relies on semantic markers for modifier-noun combinations and PP attachment and syntactic heuristics when these are not adequate. These are merely 'stop-gaps' to allow the program to work as it is conceptually defined.

When the sentence has been typed by the user, the parse begins. First ROBIE checks that all the words in the sentence are in its dictionary. If ROBIE discovers an unknown word, it calls the dynamic definition routine explained below. It then activates the grammar rules to start parsing the sentence and releases control to the rule matcher.

As its last action, each grammar rule calls the rule matcher recursively. The rule matcher then searches the patterns of each rule in each active packet, comparing the pattern with the current state of ROBIE. When a matching rule is found, the rule matcher turns control over to the grammar rule which performs the appropriate actions.

As ROBIE builds constituents, each rule calls the semantic routines as will be explained later. Words are attached to the syntax tree by ROBIE and the semantic routines produce predicate calculus assertions representing the semantic information gained by this attachment. At key points (e.g. before a headnoun or a PP is attached), a non-syntactic test is made to make certain the attachment is reasonable. The non-syntactic processor also extracts the main verb, syntactic object, syntactic subject and other sentence modifiers. These are stored in an intermediate database during the parse. When the sentence has been fully parsed, the intermediate database is ready to be given to the full semantic inferencing rules, which then produce the assertions necessary for the problem solver.

There is no true 'grammar' interpreter. Each rule is a PROLOG program executed directly by the PROLOG interpreter. The actions of the rules change the state of ROBIE, so that the rule matcher sees a new state each time it is called.

Since the rule matcher calls the grammar rule, which then calls the rule matcher, ROBIE is tail recursive. The PROLOG compiler optimises this tail recursion automatically into iteration. When optimised in this way, ROBIE's stack size does not grow during the
parse above the size of the actual constituents being built. Hence it is just as efficient on memory as if it were iterative.

There are currently 104 grammar rules in 29 packets. The vocabulary of the system is 700 words, but can be easily expanded. The morphology greatly increases the number of words recognised. (To over 2,000 words).

I will use the following rule as an example of the grammar rules. This rule is expressed in English-like notation to make it easier to read.

Rule DETERMINER in packet PARSE-DET:
To analyse a determiner, if you have the feature "det" in the first buffer then:-
1) attach the first Buffer to the bottom of the Active Node Stack as a determiner.
2) tell the non-syntactic processor you have a determiner.
3) deactivate the packet containing this rule, PARSE-DET
4) activate the packet PARSE-QP-2
Recursively call the rule matcher.

The rule 'determiner' will match if the parser state was:

Active Node Stack:
2: < open> S [SS-START,CPOOL]
1: < open> NP [PARSE-DET,NPOOL]
B1: the
B2: shy

and leave ROBIE in the state below.

Active Node Stack:
2: < open> S [SS-START,CPOOL]
1: < open> NP det-the [PARSE-QP-2,NPOOL]
B1: shy
B2: boy

The above grammar rule looks only at the features of B1: The rule bodies are very simple, each rule being able to perform only a few functions.

The program runs on a PDP-KL10-91S. The parser occupies 45K of core and the dictionaries, with a vocabulary of 700 words, add another 10K. The parser runs on 'Doc-10 PROLOG version 3' which occupies another 35K of core. Marcus's parser occupied over 200K of core, including MACLISP and the parse time was approximately .1 sec/word, including a case-frame interpreter. This is over twice the size of ROBIE, and only half as fast.
A grammar rule is restricted in the following two ways. Firstly, it can only perform a combination of the seven grammar functions listed below within the rule body. Secondly, the pattern matching for a rule is restricted to the syntactic features of the first two buffers. The only exception to this pattern matching restriction is whenever a non-syntactic test is made before a rule can match.

It should be noted that the "agreement test" follows these restrictions. An agreement test can only use the syntactic features of the first two buffers. These tests were only separated from the normal pattern to reduce the number of rules by taking advantage of similar patterns.

8.2.1 Grammar Functions

There are seven grammar functions, each of which is explained below.

a) activate a packet: This is used to reflect expectation and to control the parse. The form of the function is: activate < packetname> in < packetlist> to get < new Packet list>.

b) deactivate a packet: This is much like a POP or return from node building would be. It indicates that the current packet is no longer needed. Deactivate can only deactivate the current packet. It has the same arguments as "activate".

c) attach: This function will be fully explained in a later section. It takes four arguments: attach < Buffer> to the < Current Active Node> as a < part of speech> and call the result < New Current Active Node>.

d) new_node: This generates a new node with the features specified, or with no special features. An example of its use would be: make a new noun phrase node with the features "np" and "name" and return the result as 'NP1'. The features can be left out if inappropriate.
e) lookup: This is used when transformations insert specific lexical items into the buffer. For example: lookup the word "you" in the dictionary and call the result, B1.

f) addfeats: This is used to add certain features to a node. For example: add the features "major" and "ded" to the Current Active Node and return the result C1. It is rarely used and seldom are the features it adds checked by the rule matcher. The features it commonly adds are to assist a reader of the output.

g) semantics: This isn't really a function, but is an operation performed by every grammar rule. The non-syntactic processor is given the name of the rule which is executing, the current active node and the two buffers. The processor then extracts whatever information it desires. It should be noted that this function does not return a result and could operate in parallel with the above functions.

There are two additional grammar functions. There are two calls to "coerce" and one call to "percolate" in the grammar rules. Coerce automatically disambiguates a word, whilst percolate is used to transfer features to the AUX node if it has no lexical daughters. The uses of coerce will be explained in Section 8.2.3. The percolate call is a special form of transfer to set the tense of the auxiliary and verb phrase.

These are the only functions the grammar rules can perform. The buffers are automatically kept filled so that they are the next three items below the Active Node Stack. (Three solely to accommodate conjunction.) This is done by an explicit shift of the next unseen word into the third buffer. This could be done by a hidden mechanism, but is easy to do explicitly.

The rule matcher can look only at the features of the first two buffers to decide which rule to run. Also, once a node is attached to its mother, it is never again looked at. This has an interesting implication. The parser is never able to look inside a buffer at its internal structure. It can only look at its top-level features. This means that the syntax tree could be thrown away after the semantic interpretation has extracted the information it needs! The syntax tree does not really need to be carried around in its entirety, only the top
level features of each constituent.

8.2.2 Closing nodes

It is believed that the closure of nodes is of special significance psychologically [Kimball 1973]. Once a node is closed, it is very expensive computationally to re-open and alter it. As a result, when nodes are closed reflects when the parser is certain that the node is complete.

The question of closure is not explicitly addressed in this thesis. Since I am not concerned with the size of the Active Node Stack, there is no need to close items to conserve space. A node is open when first created. This means that daughters can be attached to it. When an open node is attached to another node, the open node is closed. In this way, closure of nodes is automatic and no separate node closing mechanism is needed.

Marcus's parser did not distinguish between words, open nodes (item under construction) and closed nodes (completed items). In ROBIE, there are three types of nodes: a word node, a closed node, and an open node. This distinction helps to make the closure of nodes more clear.

Open nodes are distinguished from closed nodes by the presence of a 'hole'. Each open node has one 'hole' and when the node is closed, the hole is 'plugged'. Each node has a list of daughters, the last of which is a variable. This variable is the 'hole'. When a daughter is attached to a node, it is unified with the hole, hence becoming the next daughter. The use of holes is an efficient implementation detail to save recursively building trees in PROLOG.

Whenever we attach an open node to another open node (for example when attaching a PP to some NPs), these holes can be selected to choose where constituents are attached. For example, when a second modifier is attached to a NP with a modifier (i.e., an NP with a PP), the second modifier may modify the NP or the PP attached to it. The 'attach' function can choose with which hole (the NP hole or the PP hole) it should be uni-
The generalisation of this technique could be used to implement pseudo-attachment [Church 1960] in ROBIE. For example, Church used pseudo-attachment to attach a PP to all its possible mothers. In ROBIE, for each node to which a PP could be attached, there is one 'hole'. Instead of pseudo-attaching the PP to all possible open nodes, one could associate the PP with a list of holes. At some later point ROBIE could decide to which hole to attach the PP and then unify them. In this way, it is possible to keep the PP attachment options open. Pseudo attachment could be done by unifying all these holes and the PP together as the same item. PROLOG structure sharing would make the PP the daughter of all the nodes. I thank Lawrence Byrd for pointing this out. This would also provide the list of 'possible candidates' as used by [Mellish 1981]. Although pseudo-attachment could be implemented this way, it is not clear that this would be a good idea. We will return to pseudo-attachment in Section 9.3.2.2.

Although this technique could be implemented in PROLOG, I do not feel that there is sufficient psychological evidence to justify it. This approach also depends on accessing the items of the Active Node Stack, which is disallowed in the parser. Again, I do not feel there is sufficient evidence to cause us to violate our principles.

8.2.3 Attachment

The most important function in the grammar is ATTACH. This function automatically performs three operations. It disambiguates words, transfers features and closes nodes.

In Chapter 5, I explained how automatic disambiguation takes place. Whenever a rule matches a pattern assuming it is a certain part of speech, it is disambiguated to that part of speech.

For reasons of efficiency, the current implementation does not follow this exactly. In ROBIE the function ATTACH does the disambiguation rather than the pattern matching.
These two are isomorphic for the following reason. Each time a buffer matches a pattern, (for example [noun] or [ns]), the parser is assuming that that buffer is a particular part of speech (i.e., noun, verb, auxverb, etc.) For example if the parser matches the pattern [ns], then is assumes the word is a noun. For every pattern in the grammar, the corresponding grammar rule will later attached the buffer which matched that pattern as the same part of speech as the pattern assumed the word was. So instead, the ATTACH function disambiguates the word when it attached. It does this by disambiguating the word to the part of speech it is attached as. This method is the same as that used by [Milne 78]. Although different in implementation, it is isomorphic to the approach described in Chapter 5.

It can be noted that with one exception the current parser performs exactly as if the pattern matching performed the disambiguation. The one exception is the IMPERATIVE rule. This has the pattern [tenseless] and is active at the start of a sentence. The rule body does not attach the tenseless verb, but instead inserts the word 'you' into the first buffer. The parser will make this the subject NP. A later rule will attach the tenseless verb as the main verb. Since the IMPERATIVE rule ran assuming the word was a verb, the word is disambiguated to a verb by a special function (coerce) in the rule IMPERATIVE.

In the above section I have indicated when nodes are closed. The action of closing a node is simply to unify the 'hole' which is its rightmost daughter with 'nil'. The node is then closed and will not accept any additional daughters, in the current implementation.

The automatic transfer of features can be illustrated by the NP building rules. When the determiner 'the' is attached to a NP, the number features 'ns, npl' are transferred to the NP node by ATTACH. When the headnoun is attached to the NP, ATTACH will try to transfer the number features of the noun to the NP by intersecting with the number of the NP already present. This approach was suggested by Marcus [personal communication]. For example, if the headnoun is 'block', then the feature 'ns' will be intersected with the features 'ns, npl' already there, resulting in the feature 'ns' on the NP node. This intersection of features is similar to the approach used by Church [Church 1980].
The agreement check will ensure that the features agree before a grammar rule is run, but this check does not change the features of the nodes. ATTACH will then update the number, as described above, when the rule runs.

This automatic transfer allows features to percolate up the syntax tree and has the same effect as the explicit transfers in Marcus's parser.

One note for future improvement. Currently in ROBIE, nodes are only attached after they are completed. That is, even though the mother is known, they are attached to their mother only when all the daughters have been found. This is purely an implementation detail. Often, the mother of a node is known before it is completed. For example, whilst parsing a PP, the next NP started will be attached to the PP once it is built. It could be attached to the PP while it was being built and save a space on the stack. This seems desirable as it would reduce the size of the Active Node Stack by one item. Also, AUX nodes could be attached to the current S node while they are being built and various relative clauses could be attached to the NP they will modify whilst being built.

In terms of the current implementation, this approach would probably take more memory cells, rather than less. Items in the Active Node Stack are stored as a list, so the addition of an item, means the addition of one pointer in the list. If it were attached to its mother, then a pointer would be needed to indicate its mother, as well as a pointer to indicate that the mother has one or more incomplete daughters.

8.2.4 Implementing the Production System Grammar

Production systems of the form used in the current grammar (one antecedent) are very easy to implement in PROLOG. Although the exact details are not important to the claims made here, I will explain this briefly.

Each rule has a pattern consisting of the rule's packet name, priority, the features which must be present on the two buffers, the agreement check, and the name of the rule. For example, the pattern for the rule DETERMINER is:
Based on the current active packet and priority of the rule ('10' in this rule), the rule matcher will automatically match this pattern. The pattern gives the features which must be true before this rule can run. In this case, Buffer1 must have the feature "det". If this condition is met, the rule matcher runs the body of the rule which is, called "DETERMINER". The 't' means that the rule matcher doesn't care what is present.

If it does not match, then a fail is generated. This then causes the pattern matcher to find the next pattern and the process is repeated. By automatically backtracking, all the priorities and active packets are checked in order. This use of backtracking is purely for the production system implementation in PROLOG and does not affect the claim of determinism since the production system could be fully hashed and compiled so no backtracking would be needed.

8.2.5 Rule Order

One question is whether rules should be ordered or not. Many people favour unordered rules, at least in the virtual sense. I have never been an advocate of this, mostly because rules need, at least conceptually, to be tested in order, in an implementation. There is seldom a conflict between rules and ordering them can help to simplify the patterns.

In ROBIE, rules are ordered two ways. Each rule has a priority. This is one of \{5,10,15\}, 5 being the highest priority, and 15 the lowest. All the priority 5 rules will be tried in all the active packets before any of the priority 10 rules. This provides order across packets. In order to follow the principle "the most constrained rule is matched first" as was explained in Chapter 6, the rules are ordered inside each packet. This is done by having one rule before the other in the grammar. PROLOG tries each rule one at a time.
8.3 The Dictionaries

I have already explained the general format of dictionary entries and the morphology in Section 5.1.1. The parser can have any number of separate dictionaries. Each of these are loaded additively. Hence, the vocabulary of ROBIE is very easy to expand. It can also be extended "on the fly" as explained below.

The root of each word is entered in one of the dictionaries. This entry is a compound lexical entry composed of all the features for each part of speech the word can be used as.

Function words differ from other words, in that the root word of the function word can be a feature. For example, "are" and "is" and "be" can all have the feature 'be'. For open class words, it is not possible to make the root word itself a feature as I have done for function words. The grammar can then test for the presence of a specific function word by having that word as a pattern.

Like Marcus's, my parser has a dynamic definition capability. If ROBIE fails to find a word in the dictionary and the morphological routines do not locate the word, then the system warns the user of the unknown word. It then asks the user for a word like the unknown word, on a purely syntactic basis. The user types a similar word and the parse continues. This is very useful for demonstrations and makes it possible to incorporate unknown words at any time.

8.4 The Packets

A grammar rule can match only on the features of the first two buffers. This is compared to the five nodes which Marcus could check and the six nodes which Church could check. However, one more piece of information is needed, the Packets. The packets are the structure of my production system and reflect the top-down component of the parse. This top-down information is essential to my method.
Because I have packets, I need to have two functions in the grammar to manipulate them (activate and deactivate), as well as a separate stack for them to reside on. This adds a few mechanisms to the parser. Church [Church 1980] does not use packets. Instead, his parser must look at the items of the Active Node Stack. This is partly the reason that his parser needs the extra items to look at. In the next section, I will compare his method, using "dotted rules", with what I do. I feel that these approaches are essentially isomorphic, but his is conceptually simpler.

8.4.1 Dotted Rules

The phrase structure expansion of a node in ROBIE is concealed in the packets and grammar rules which relate to that node, but in YAP, this is represented by "dotted rules". These are best explained in [Martin, Church and Patil 1981]. The next part of the discussion will be based on this paper [ibid, p. 8]. A dotted rule is defined as a "context-free grammar rule, with a dot inserted to indicate how much of it has been parsed". For example, if we have the context-free rule:

\[339\] VP -> V NP PP

we can add a dot showing how much of the item has been built. At the start of the VP, the dot will be inserted before the "V". After the verb has been found, the dot will be as follows:

\[340\] VP -> V. NP PP

After the object has been found, the dotted rule will be:

\[341\] VP -> V NP. PP

Using this notation, we have a method of keeping track of the item we are building and how much of it has been built. It has been claimed that the packets are merely a less transparent notation for the same structure. For example, if we are parsing a VP and do not have the main verb yet, corresponding to \[339\], we will have the packet PARSE-VP active. After the verb has been found, \[340\], we will have the packet SS-VP active. After the
object has been found, the packet OBJECT will be deactivated, this is the same as situation [341].

It has not been explored thoroughly, but I believe there will be a one-to-one correspondence between the packets and the dotted phrase-structure rules above. Therefore ROBIE could do without packets and instead the rules would match on three items: the two lookahead buffers and the phrase structure dotted rule of the bottom of the Active Node Stack.

These dotted rules have one other advantage over the packets. The dotted rules provide a better formalism for the top-down component of the parser and make subcategorisation more explicit, hence helping with gap finding. This then assists with WH movement and conjunction. Church uses these rules for this purpose and they work very well.

In ROBIE, the rules need to match three items, the two lookahead buffers and a notion of 'state'. This notion of state could be implemented either by the packets, or by the dotted phrase structure rules. The possible benefits of dotted rules were discovered too late, and the advantages were too small to warrant implementing them in the current parser.

8.5 A Few Notes on the Grammar

There are three areas of the grammar which this research has not investigated in detail. These areas have been the focus of research by other authors, and the approaches I have adopted has been based on their work.

8.5.1 Passive

The approach to passive used in ROBIE has been copied directly from [Marcus 1980]. The pattern [be][en] indicated that the sentence is a passive, and a grammar rule in the packet BUILD-AUX adds the feature 'passive' to the main verb. When the VP is started, the packet PASSIVE is activated. The rule in this packet inserts a trace into the first buffer. This trace is then bound to the current syntactic subject by the non-syntactic
processor. The non-syntactic processor will note that the sentence is passive and use the logical subject as required. The justification and implications of this approach are in [Marcus 1980].

8.5.2 Conjunction

This research has not investigated the problem of conjunction. ROBIE has a very simple and slightly ad-hoc method for handling conjunctions that works for many simple cases, but is incorrect in general. The method used will handle:

- [342] The boy and the girl hit Mary.
- [343] The boy hit the girl in the park and in the head.
- [344] The boy hit and kissed the girl.
- [345] The boy hit the girl in the park and street.

but not:

- [346] The boy hit Sue and kissed Mary.

When the parser encounters a conjunction, it is pushed onto the Active Node Stack and the packets CPOOL and PARSEVP are activated. These packets contain the rules to begin noun phrases and verb phrases. When the constituent following the conjunction has been fully parsed, the conjunction word is dropped from the Active Node Stack into the second buffer. At the same time, the constituent before the conjunction word is dropped into the first buffer. The rule X.AND.X will then attempt to match. If the contents of the first and third buffers are syntactically the same, they will be conjoined.

This technique can conjoin single words, NPs, VPs, APs, PPs, etc. It will be wrong in the case where the first constituent after the conjunction is a sub-constituent of a larger item, as in [346]. This approach does not use any non-syntactic information or verb subcategorisation.

As pointed out in the introduction, no one has yet researched a satisfactory method of parsing conjunctions with less than three buffers. Church [Church 1980] has developed a promising approach to conjunction for a deterministic parser based on verb subcategorisation.
The investigation of movement phenomena has not been a focus of this research. A method for handling subject-auxiliary inversion, relative clauses and WH-questions has been implemented, but it is partially ad-hoc and will not be described in detail in this thesis. To handle movement, a "movement stack" similar to the ATN 'HOLD List' [Woods 1973] has been added to the parser. Items are pushed onto the movement stack when they are first parsed and inserted into the buffer when the appropriate gap is detected. This approach is very similar to that used by [Charniak 1981]. This movement stack seems necessary, as a place is needed to hold the node undergoing movement until the gap is found. The author feels that this approach has limited implications for the two versus three buffer issue, but may have interesting implications regarding the use of non-syntactic information.

The method used is adequate to handle the examples of movement which appear in Appendix B, but would fail on more difficult examples. For a discussion of movement phenomena and its implications in a deterministic parser, see [Marcus 1980].

8.6 The Agreement Checks

Throughout the earlier chapters, we have seen several "agreement checks". These were used to ensure a certain type of agreement was upheld before a rule could be run. In these earlier sections, it was stated that each test was really a shorthand notation to simplify the patterns of the rules. It was also stated that these tests were restricted to checking the patterns of two buffers. In this section we will see each of these tests and how they are implemented. There are four agreement checks: Determiner-agree, Complex-noun, Affix-agree and Verb-noun-agree.

Determiner-agree is used to be certain that the determiner and noun agree in number and that the word following a determiner can "take a determiner". This test is as follows:

([[]] and [t] represent a test which is always true.)
Complex-noun is used to decide if a singular noun should be used as a noun or not. This is used only after the NP has at least one head noun. If this test succeeds, the word in the first buffer will be attached as a noun.

[]  [pronoun or name] or
not[modal] [tenseless]

Affix-agree is used to be certain that an auxiliary verb agrees with the following verb. It is needed for Auxiliary inversion examples.

[have] [en] or
[be] [en or ing] or
[modal] [tenseless] or
[do]

Verb-Noun agree is used to ensure that the subject and main verb agree in number and person. This test is only a simple decoding of the person-number codes.

[ns,n3p] [v3s] or
[not(ns or n3p)] [v3s] or
[ns,not(n2p)] [13s] or
[t] [v3s] or
[n1p,ns] [v1s] or
[npl or(n1p,ns)] [vpl2s]

This is all the tests and one can see that they really are only a syntactic shorthand.

8.7 The Semantic Database

In ROBIE, semantic interpretation has been added in two ways, the semantic interpretation and the non-syntactic checks. As described in Chapter 4, each rule has a call to the semantic interpreter. In Marcus's parser, the semantics was done purely by demons.
For psychological reasons, I feel my approach is better. The second usage is to decide which rule to run when two or more rules match. This is used more rarely as was explained in Chapter 4.

The semantic database is really a cross between a representation of the syntax tree as predicate calculus assertions and some semantic information. Most of the information in it is derivable from the syntax tree. This database provides a universal middle representation of the sentence, which is suitable for almost any application. Although many of the items in it were developed for the MECHO system, the representation could provide input for translation, or database query.

It should be noted that this semantic interpretation is only partial. It provides the interface to a larger system and a much more elaborate semantics could be added. However, for the current application, the MECHO front end, the semantic interpretation is as large as was necessary. The database is used by the semantic inferencing program for MECHO [Mellish 1980] and has been designed to meet its specific needs.
We have now finished developing our model of the syntactic processor and have seen that it compares favourably with the related literature. In the two remaining chapters, we will look at several areas for future investigation. Our intention is to explore the theoretical implications of these problems and, except where noted, none of the techniques discussed here have been implemented.

We have seen how to handle many classes of ambiguity automatically, but there are some problems raised in the earlier sections that need to be rectified. In Chapter 5 we discussed the handling of 'have', but our solution required three buffers. Since we have said that we will use only two buffers this is not satisfactory. In this chapter, we will discuss global ambiguity and the method by which verb particles are handled as well.

9.1 Verb Particle Handling

The parser uses a fairly simple approach to parsing verb particles (prepositions used to modify the verb) and separating them from Prepositional Phrases. The approach outlined here is implemented and works well in the MECHO world, but is not generally adequate. As I explain the basic technique, I will explain its limitations and necessary improvements.

For each verb, there is a list of each particle that, if it occurs with the verb, must be used as a particle. For example:

[347] The robot picked up the block.

To handle this, there is a high priority rule in the packet SS-VP with the pattern, "[particle], sem chk (particle)". Before this rule can apply, the semantic check routine checks the obligatory list of particles associated with the semantics of the main verb. This
is implemented as a set of semantic markers. If the particle is on this list, then the rule
will match and attach the particle. Otherwise, the rule will not match and the particle will
be left in the buffer to be picked up by the PP rules.

This check is based only on a list of verb-particle pairs. To be correct, the list
should include the optional and obligatory prepositions that the verb can take and make a
more intelligent decision based on these possibilities. The current system of verb subcategor-
isation does not include this.

If the above rule does not match, then the particle will remain in the first buffer.
If the particle is followed by a word which can start a noun phrase, a PP would eventually
be built and handled in the normal way.

Finally, if there is no element of a PP after the particle, then it will still be in the
buffer. At this point, a low priority rule with the pattern [particle] will match and attach it
to the verb phrase as a particle. The same rule will pick up the ‘stranded’ particle if the
particle has been moved. No attempt has been made to ensure that this does not accept
ungrammatical stranded particles. This simple technique will resolve the
preposition/particle ambiguity in the following:

[348] He threw out the trash.
[349] He threw the trash out.
[350] He sat on the chair.
[351] *He sat the chair on.
[352] I want to throw the trash out.
[353] I want to throw the trash out the door.
[354] *I want to throw out the trash.
[355] *I want to throw out the trash the door.

There is one more aspect of particle handling we will consider briefly. This is
shown by the examples:

[356] He looked up the street.
[357] He looked up the address.
[358] He looked up the mountain.

In [356], two readings are possible and (supposedly) perceived. This is an exam-
ple of global ambiguity, which will be explained later. The important thing to note is that
the choice whether the word “up” is a preposition or a particle cannot be made until the fol-
lowing NP has been parsed. Even then, the decision must be made on a non-syntactic basis, as the syntax of both of these choices is the same. This decision must be based on selectional restrictions to be correct. If this analysis is to be accepted, it shows that any system starting a PP with a preposition and without building the NP first, as Kimball [Kimball 1973] has suggested, will be incorrect for these examples, or will need to backtrack if a distinction is to be made.

In order to parse verb particles correctly, better verb subcategorisation is needed as well as selectional restrictions on the verbs. Neither of these are more of a problem for a deterministic parsing than for a non-deterministic parser.

9.2 HAVE re-visited

In Chapter 5, Marcus's 'Have Diagnostic' was replaced by number and verb particle agreement. With this new method, the pattern for the rule YES-NO-QUESTION should be:

\[\text{[auxverb][np][verb], agree(auxverb, verb), agree(verb, np)}.\]

This means the rule YES-NO-QUESTION should only be run if the subject and verb agree in number and the auxiliary verb can be placed before the main verb in the verbal cluster. If this rule fails to match, then the rule IMPERATIVE will run instead and the sentence will be parsed as an imperative. In that chapter, I showed why this is needed and how it works.

In order to handle this rule, we need the pattern:

\[\text{[have][np][verb]}\]

This rule pattern presents two problems. Firstly the pattern uses three buffers in order to check the verb features as well as the NP features. Secondly, in order to have the NP built in the second buffer, we need to perform an Attention Shift.

The three buffers seem to be necessary in order to "see" the word 'Have' in the first buffer, the NP that it must agree with in the second buffer and the verb it must agree
with in the third buffer. This is a violation of the two buffer constraint. If this rule is correct, then we are unable to use only two buffers throughout the parser.

We have also eliminated the Attention Shift from the parser. In no way can the NP be built in the second buffer, given the current framework. Conversely, it seems to be necessary to build the entire NP before the decision is made, since we must know the number of the NP.

This is the only rule in ROBIE that seems to need three buffers and also the only rule that needs the Attention Shift! (Except, of course, conjunction.) In order to handle this one phenomenon two special mechanisms would have to be added to the parser. Clearly, either the rule is incorrect or the three buffers and the Attention Shift are necessary to the parser. Is it possible to find counter examples to our Have-Diagnostic? Consider:

[359] Have the boys taken the exam?
[360] Have the boys take the exam.
   (Our original sentences)
[361] Have the boys taken the exam.
   (Yes, an Imperative.)
[362] Have the eggs broken?
[363] Have the eggs broken.
   (Both are acceptable.)
[364] Have the students put it back?
[365] Have the students put it back.
   (Again both are acceptable.)

Sentence [361] can be paraphrased as: The speaker instructs the listener to make a third person deliver the exams to the boys. If the YES-NO-QUESTION rule were correct, none of the imperative sentences above would be possible. Although some are obscure, they are all clearly acceptable. It is certainly easy to produce counter examples to this rule. The imperative reading of [359] is clearly possible as [361] demonstrates. Even though [361] is acceptable, very few people notice this reading before it is explicitly pointed out. The rule YES-NO-QUESTION correctly predicts the preference for these sentences, but the rule does not say how to get the imperative reading.

Our re-formulation of Marcus's HAVE-DIAGNOSTIC into the YES-NO-QUESTION rule shares something with the original rule. It is not always correct. But what would the correct rule look like and where did this rule go wrong? Consider the following
sentences:

[366] Have the students who missed the exam take it.
[367] Have the students who missed the exam taken it?
[368] Have the students who missed the exam taken to the room.
[369] Have the students who missed the exam been taken to the room?
[370] Have the students who missed the exam taken in the back room.
[371] Have the students who missed the exam taken in the back room go home.
[372] Have the students who missed the exam taken in the back room gone home?
[373] Have the students who missed the exam taken in the back room finished yet?
[374] Have the students who missed the exam taken in the back room finished off.

Before continuing we must ask which, if any of these sentences caused us to garden path and where we had trouble when reading them. Finally, was there any trouble with the word “have”?

Each of these examples meets the criterion of a potential garden path. The distance from the ambiguous word (‘have’) to the disambiguating word is far more than our three buffers can encompass. If these are truly potential garden paths, the garden path effect should have been felt in at least one of each pair. However, many people reading these examples do not feel that they are garden paths.

These examples were tested in the second experiment to determine whether they caused the garden path effect. The sentences which were tested are as follows:

[375] Have the students take the exam.
[376] Have the students taken the exam?
[377] Make the students take the exam.
[378] Did the students take the exam?
[379] Have the students taken in the back room finished yet.
[380] Have the students taken in the back room finished off.

The results of this test will be discussed in two parts. First the simple sentences:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[375]</td>
<td>369</td>
<td>Imperative</td>
</tr>
<tr>
<td>[376]</td>
<td>268</td>
<td>Yes-No-Question</td>
</tr>
<tr>
<td>[377]</td>
<td>364</td>
<td>Imperative</td>
</tr>
<tr>
<td>[378]</td>
<td>279</td>
<td>Yes-No-Question</td>
</tr>
</tbody>
</table>
It was predicted that these examples would show that there was no significant difference in reaction time between reading 'have' as an imperative versus 'have' as a yes-no-question. There was no significant difference between [375] and [376]. There was a slight order effect in that the first sentence of each pair presented took longer than the second sentence. Sentences [377] and [378] showed the processing difference of imperatives versus yes-no-questions, (significantly different at the p< .05 level). In both sets, the Imperative was one second slower than the Yes-No-Question. This suggests that the 'have' examples were not different from the normal examples.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Mean Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[379]</td>
<td>832</td>
</tr>
<tr>
<td>[380]</td>
<td>930</td>
</tr>
</tbody>
</table>

There was a very definite order effect with these examples. The times given are those obtained when the example occurred first in the order. However, when the example occurred second, the times were 300 and 490 respectively. The difference between the sentences was not significant. It seems that when the subject saw the first sentence of the pair, irrespective of which sentence it was, they had a considerable amount of trouble. One can see that the time was almost exactly the same at the first exposure. These times showed that the subject had problems with these sentences the first time, but then learned most of the example, which assisted the understanding of the similar sentence the next time. It is not possible to tell whether the problems were due to the reduced. Unfortunately we have gained no useful data from this second set, but the results of the first set show that these do not cause the garden path effect.

The examples in this section show an extra complication with this diagnostic. The pattern [noun][verb,ed] must be present for the word 'have' to move between and have constituents to agree with. This is also the pattern for the reduced relative clause which we saw in Chapter 4 and which can also lead to a garden path. Therefore, this diagnostic must.
interact with the potential garden path, making the situation much more complex, but helping to explain why our rule cannot be correct.

Our simple formulation of the rule with number and verb agreement does not account for the data on this type of problem. Therefore, the old diagnostic must be incorrect. This diagnostic required two special mechanisms (Attention Shift and 3 Buffers) to be added to the parser. If we keep the parser as originally formulated, then all we lose is an incorrect rule. Henceforth, we will not use this and will not need the three buffers nor Attention Shift. To see how the parser should handle the word 'have', it is important to see how people handle it.

There does not appear to be any experimental data relevant to this question. Its implications are important and would make a subject for further research. The following are purely SPECULATIVE explanations of what may be nearly the correct approach. These accounts are given to show the important implications they could have.

There are several possible solutions to this problem.

1. Let the usage of 'have' as an imperative be illegal. In some cases of 'British English', this is true. Many of my British informants tell me 'have' is never used as an imperative. The use of 'have' as an imperative seems to be a rare problem. If it did not exist, then our theory would be simpler. One could say that it is not used in 'British English' because of processing difficulties. It seems the uses of this are mainly colloquial American, so this may have evolved in some way.

2. In some dialects, only a pronoun can exist between the word 'have' and the verb. This removes the need for the Attention Shift, as predicted, but again many people accept sentences with a longer NP in the middle.

3. Accept that some of these examples are garden paths and back up does occur. The disambiguating word is outside our limited lookahead and the syntax cannot decide between alternatives. Hence our Semantic Checking Hypothesis applies and non-syntactic information (intonation) is used to resolve the problem instead. This explanation fits the data, with only minor problems. At least part of these should cause the garden path effect.
In the next section, we will see why they may not. The data from the second experiment shows that these are not garden paths, but further experimentation needs to be done.

4. Chomsky's Auxiliary Inversion analysis is not quite correct. If the AUX is never "moved", then it should be possible to have a syntactic analysis that is the same for both alternatives until the point of removal of the ambiguity. At this time, only the features would need to be adjusted. This operation is very simple and would explain why the alternatives can be held open for so long and no conscious effort is needed for any of the examples. This proposal will be explained more clearly below. First let us see why Chomsky's Aux-Inversion is incompatible with the non-garden path account of these sentences.

We will use the following sentences for this discussion:

[381] Have the boys taken the exam?
[382] Have the boys take the exam.

For these sentences, the trees according to Chomsky would be:

One can see that these trees are quite different. Assuming it is not possible to postpone the decision on "have" until the verb is "seen" (as stated above), then the parser would have to commit itself to one tree or the other before the verb was seen. If the parser committed itself to the wrong analysis, when it discovered it is the wrong one, extensive changes of the tree would have to be made to correct the mistake.

As these trees show, if Aux-Inversion does indeed operate, the two examples have very different structures. Given this analysis, in no way can the two structures look the
same until the choice between imperative and yes-no-question is made. It seems that the only solution to this problem, given this analysis, is that the Attention Shift [Marcus 1980] is correct. With the Attention Shift and three buffers, the NP can be built before the auxiliary verb is moved. The parser can then examine all the features and decide whether the sentence is a yes-no-question or an imperative. If it did not examine all this information, it would make an error on half of these examples. Because of the extensive change that would be necessary, some conscious effort should be noticed. But I have already stated that no conscious effort seems to be involved here.

Since we do not accept the Attention Shift and the three buffer lookahead, the Aux-Inversion analysis is not possible using this parser. However, if the constraints we have assumed here are correct, then this analysis is wrong. Only if another viable analysis can be found to fit our constraints can this type of sentence be successfully parsed by ROBIE. Such an analysis could have a significant effect on current opinion concerning transformational versus non-transformational grammars.

Let us assume that a person creates the same structure for both possibilities until the disambiguation takes place. Then the only difference will be the features and it is simple to eliminate incorrect features. In dealing with ambiguity in Chapter 5, we gave each word all the syntactic features it could use. We then disambiguated it by removing all the incorrect features. The same method applies here. This would explain why the long 'have' examples do not cause garden paths, or indeed any trouble at all. It also provides an approach compatible with the two buffer constraint.

Many linguists have felt that a non-transformational approach to language is better than the transformational approach of Chomsky. A non-transformationalist believes that the inverted auxiliary verb remains in the front of the subject in the surface structure. We shall use Phrase Structure Grammar (PSG) [Gazdar 1979,1980,80b,80c] as an example of this school of thought. These two examples come under the following rules in this system:
The Imperative rule [Gazdar 1980a, p. 38]
< 16, [V= ⌈V V= ⌋ ... >
[+ IMP] [+ BSE]

The AUX Inversion Meta rule [ibid, p. 46]
SAI: < [V- V V- ] ....> = > < [V= V V= ] ....>[
[+ FIN] [a]
[+ INV ] [a]
[+ AUX]

Using these rules, the tree analysis for [360] and [377] are as below:

IMPERATIVE

= 

V
[+ IMP]
/ \=
V V
[+ IMP] [+ BSE]
| /
[ + BSE] have = 
N V
\ /
the boys take the exams

YA5-NO-QUESTION

= 

V
[+ INV]
/ \=
V V
[+ INV] [+ PSP]
| /
[ + PSP] have = 
N V
\ /
the boys taken the exams

Ignoring the features, these two trees are identical. This is exactly as suggested.

Given this analysis, a person could keep both features on the word 'have' and hence the tree nodes, as for ambiguous words, until the verb is found. This analysis seems to fit the psychological facts better. The same will be true for any analysis that does not move the Auxiliary verb in the surface structure.

These examples may provide an interesting area in which to investigate the psychological implications of Chomsky's Extended Standard Theory (EST) versus Gazdar's Phrase Structure Grammar. The results of the second experiment suggest that the proposals here may be correct. Before this can be confirmed however, data need to be collected on these examples to resolve the question of conscious effort. This investigation may show that one linguistic analysis is wrong.

To implement this in the current parser would be impossible, but a complete rewrite to PSG may make this possible.
In conclusion, we have seen that our reformulation of Marcus's Have-Diagnostic required two extra mechanisms and violated ROBIE's constraints. We also saw that it was not correct. In trying to reformulate the Diagnostic in a psychologically plausible way, we have seen that the Diagnostic was impossible to re-formulate using EST, but very elegantly done using PSG. Thus ROBIE has made an interesting and verifiable prediction and suggested an area for future collection of psycholinguistic data.

Finally, it should be noted that these examples overlap into the area of global ambiguity. The solution of global ambiguity presented below may apply to this case as well. The role of non-syntactic interaction described will satisfactorily account for these data.

9.3 Global ambiguity

Many sentences when viewed in isolation can have more than one possible and acceptable reading. For example:

- \[383\] They are flying planes.
- \[384\] I told the girl that I liked the story.

In both of these sentences at least two acceptable readings are possible. Without contextual information, it is not possible to tell which reading is desired. We will call these sentences "globally ambiguous". This is because there is nothing in the sentence itself to resolve the structural and part of speech ambiguity.

Since we have no knowledge of the method used by people to understand these sentences, it is difficult to produce a psychological model. It is necessary then to decide what a person does when one of these sentences is read. Are all possible meanings perceived; or only one meaning? If people seem to perceive both meanings at once, it may be necessary to expand the typical notion of deterministic parsing to allow this.

Unfortunately the answer to these questions is unknown. I feel that this area is a thesis by itself. The explanation offered here is purely SPECULATIVE. I will not offer a firm answer to these questions, but will give several possible solutions and my theoretical predictions.
As we are not sure exactly what people do when they read these sentences, let us look at all possible options. There are essentially three options: a) Return a preferred reading and then the other readings, b) Get both readings at once, c) Return one and only one reading. Let us look at these in turn.

9.3.1 Preferred Reading, then Other Readings

At first only one reading is perceived and then the other possible readings are found by backtracking. This would mean that the preferred reading is always found first. The parser 'knows' there are other possible readings as well and seeks them out. If the parser did not know that there are other possible readings, it would have to backtrack on all sentences even those with only one possible reading, in order to find all potential meanings. This seems to be incorrect. Most people believe that this does not occur and assuming that the parser does not backtrack on all examples, how does it 'know' when to backtrack and try to find a second reading whenever the first reading found is acceptable?

In an ATN, the preferred reading is usually expressed by ordering arcs. It is generally accepted that context can bias preferred reading. (We will return to this later). Does this mean that context will dynamically re-order the arcs to reflect the change in preference? If context does not cause some sort of re-ordering, how can the parser reflect the different preferences? Whilst it seems possible to design a system that is capable of handling dynamic re-ordering, I feel that an arc ordering approach to preference seems unable easily to reflect how the preference can change. We have met this problem before in Chapter 6. We will assume that this option is not correct.

9.3.2 All at Once

It may be that all possible readings are, somehow, produced at the same time. If the theory predicts that all meanings can be simultaneously perceived, then it must explain how this could occur and when each should be produced. Remember that we have forbidden
parallelism for our deterministic parser. If it can be shown that people produce multiple
readings in parallel, it may be necessary to expand the traditional notion of deterministic
parsing.

Martin's parser [Martin, Church and Patil 1981] shows that there are so many
possible readings for sentences, that, clearly, all possible readings can not be perceived. One
of his examples has 958 parses! Consider these two sentences:

[385] What are the production costs as a percentage of sales?
[386] List prices of single unit prices for both 72 and 73.
(from [Martin, Church and Patil 1981, p. 62])

For [385], Martin's parser finds 14 parses and for [386], it finds 85 parses!
Clearly a person does not perceive all 85 parses of [386] one at a time by backtracking. I do
not believe that, even without backtracking, as many meanings are found as a Chart parser
can produce in parallel. If each possible parse started a separate parallel process, there
would be a very large number of parallel processes. So we must assume that the use of
preference and non-syntactic information eliminates most of these possibilities.

Finding all meanings seems possible only by using some form of pseudo parallel-
ism. This could be done by:

9.3.2.1 Similar Trees

One possibility is that each reading has the same tree up to some point. This
would be an extension of the 'have' case we have just seen. For example in the sentence:

[387] Have the balls hit the wall.

In Gazdar's system, the word 'hit' would have both the feature [+PSP] and the
feature [+BSE]. This means that the trees would be identical and neither set of features
could be eliminated and hence both readings encoded into it. Unfortunately, from the
currently accepted linguistic analysis, most of these examples seem to have too great a varia-
tion for this to apply. Psycholinguistically, this might be the easiest and most elegant
method.
Here is another example of how this may work. Consider:

[388] He looked up the street.
[389] He looked up the address.
[390] He looked up the mountain.

In these examples, it is generally accepted that [388] has two possible readings; in [389], "up" is a particle and in [390], "up the mountain" is a prepositional phrase. Notice that the crossed meanings are all possible given the correct context. How can this be done? One approach is to interpret [388-390] as 'He (looked up) NP', that is, with the word "up" as a particle, and allow the non-syntactic processor to decide which reading relevant while it interprets the verb phrase.

But consider:

[391] *Up the address he looked.
[392] Up the mountain he looked.
[393] He looked the street up.
[394] *He looked the mountain up.

In [392], the PP can be fronted, but not in [391]. This data is used to show that, in one analysis the word "up" is a particle, whilst in the other it it a preposition.

The unacceptability of the starred examples above is not based only on syntax. Instead it could be because the selectional restrictions of the verb are violated and it is on this basis that these examples are rejected. Assuming they are different, before the ambiguity in these examples can be resolved, the semantic meaning of the NP must be decided. This means that the NP and all its modifiers must be fully parsed before the ambiguity of "up" is resolved. This can be done only by using the Attention Shift, which I no longer use. Clearly any non-backtracking approach that starts the PP as [Kimball 1973] has suggested: when the preposition arrives in the first buffer, cannot correctly handle these sentences given the current analysis.

On the other hand, if the (looked up) analysis above is accepted, then the "Function words start new nodes" principle of Kimball will still be valid and these examples would fit the current model.
Church [Church 1960] has introduced another method to accomplish the encoding of multiple meanings, which he called "pseudo attachment". Pseudo attachment is a method of encoding several structures in a single tree. This was originally developed for PP attachment and has not been tested on other types of ambiguities. Whether it will be applicable is hard to determine. Pseudo Attachment can be interpreted as "if the lower node may be attached to the upper node, it is attached". For example in:

[395] I saw the man on the hill with the telescope.

The PP "on the hill" will be pseudo attached to all the open nodes preceding it. This means that it will be attached to the NP, VP and S node. The PP "with the telescope" will then be attached again to all open nodes preceding it. These would be the NP, "the hill", the VP and the S and are exactly the items of the Active Node Stack, as was discussed in Section 7.1.2. This single structure then encodes all possible parses. Church used pseudo attachment to cover cases where a non-syntactic processor should decide the correct attachment, but as his parser has no semantic component, didn't decide.

If it is true that both meanings show up at once in all examples of global ambiguity, then this strategy may provide a solution. The analysis of "have" given earlier could be a case of this. For example, consider:

[396] They are flying planes.

The tree for this might be:

\[ S \]
\[ / \]
\[ NP \text{ AUX} \ [\text{VP}, \text{AP}] \]
\[ \uparrow \]
They are flying planes

Given that "flying planes" has the same internal structure when it is a VP as it does when it is a AP (which may be considered dubious), then the structure for [396] would be ambiguous. This would then give us the double meaning.

Each individual meaning could be produced in two ways. First if the number of the sentence allows only one of the two meanings, then the second feature can be deleted.
This could handle:

[397] Kissing aunts is boring.
[398] Kissing aunts are boring.

Secondly, the non-syntactic processor could resolve which of the two readings to throw away. Since the non-syntactic processor uses context to assist with its decision, it could decide which to keep. The Chart parser [Kay 1973] does not have this problem. It can get all meanings in parallel and the non-syntactic processor will choose which is preferred.

The disadvantage of the multiple encoding approach is that the normally accepted syntactic analysis for these sentences in most cases is too different to make this approach possible. Short of a major revolution in linguistics, it does not seem possible to have identical trees for all examples of global ambiguity until disambiguation can be made.

Conversely, this shows an area to investigate the psychological plausibility of various linguistic systems. The method used by people to handle these examples may help to guide linguists in the development of theories of Universal Grammar. This should make an interesting area for future investigation.

9.3.2.3 Produced in Parallel

Finally, it may be that the two readings are produced in parallel. When the ambiguous point is detected by two rules matching at once, then two processes are spawned. From this point both analyses are produced in parallel. Each possibility will then proceed as an independent process. We can detect the ambiguous point with the 'rule matcher'. In production systems, the parser runs the first rule to match. Often there is a conflict with a rule of a less constrained pattern and a rule with a more constrained pattern. By running the first rule to match, we resolve the conflict, but what happens if two rules of equal constraint match?

To handle this problem, both rules must be run. This would be easy to implement this on a parallel machine. We would need our system to find all rules of equal constraint that match and run them all in parallel. Each rule then calls the 'rule matcher'
recursively and independently. In this way both paths can be parsed at once. On a serial machine, we would have to simulate the parallelism in the traditional ways.

The non-syntactic processor can fail either path at any point. Remember that each rule makes a contribution to the semantic interpretation of the sentence and each path now builds its own semantic interpretation. If one of these paths fails to make semantic sense, then the non-syntactic processor for one of the rules will abort the parse and the path will die.

Context will then have its effect just after the process has spawned. This is the most simple method of accounting for global ambiguity for me.

There are however a few remaining problems. The effect on memory load is not clear. Several parallel processes may be in violation of a limited memory view. The biggest problem in the theory is one of determinism. This approach is exactly the approach of the Chart parser and as such is the non-deterministic approach we are trying to avoid. To accept this would change most of the theory. There is, however, one further option.

9.3.3 Only the Right One

We have now explored the possibility that a person produces a preferred reading, then other readings and that a person may somehow produce all possible meanings in parallel. The final possibility is that given a natural situation, only one meaning is perceived. That is, in context only one meaning is noticed. This means that context must feed directly into the handling of these examples so that it may influence the decision.

Let us again look at the “flying planes” examples. Tyler [Tyler 1977], shows that prior context can influence parsing. She was trying to test the on-line model of sentence processing proposed by [Marsten-Wilson and Tyler 1975]. This model predicts that the non-syntactic processor helps to make parsing decisions every step of the way. We have discussed this model in Chapter 6 and here only the data are presented.
This paper is convincing in that a 'true' result is found by the experiment, but Tyler is guilty of the "Across the Board Generalisation" [Clark 1973]. She tested the "flying planes'' construction only, but claimed that the result was applicable to all of Natural Language.

In her experiment, the subjects heard a sentence presented via earphones. When each control word was pronounced, a probe word was flashed on the screen in front of the subject. The subject was then timed while he named the probe. The subject then noted down whether the probe provided a good or bad, continuation of the sentence.

Examples were of the form:

[399] If you walk too near the runway, landing planes...
[400] If you've been trained as a pilot, landing planes...
[401] If you watch them as they swoop down for the kill, hunting eagles...
[402] Since it's forbidden by law, hunting eagles...

The probe words were in two groups, appropriate and inappropriate (For example "are", "is"). It was predicted that the appropriate probe would always be faster to name than the inappropriate one. The necessary control sentences were included and examples were properly mixed.

Tyler's [Tyler 1977] results show that naming the appropriate word was faster than naming the inappropriate word. The judgements written down by the subjects also showed agreement with the interpretation of the sentence. For ambiguous examples, as above, the naming latency for the appropriate word was 519 msecs while the inappropriate word took 555 msecs. For the unambiguous examples, the times were 554 msecs and 581 msecs respectively.

Tyler [Tyler 1977] then claims that the ambiguity of the fragment 'flying planes' is resolved with the aid of the prior context. We will accept this conclusion and see how it fits my theory.
When we defined global ambiguity, we said that, given the sentence in isolation, there is no way to tell which is the desired meaning. This is just another way of saying that syntax cannot decide which of several possible analyses to choose.

However, we have already decided on a method to use when there is a choice of alternatives between which the syntactic processor cannot choose. We let the non-syntactic processor decide. If discourse can affect these examples, then the non-syntactic processor is making the decision here too. Therefore, my theory of the "when and why" of non-syntactic interaction is applicable to this case.

For globally ambiguous sentences then, the parser must first detect that two rules of equal constraint can match. Rather than executing each in parallel, the non-syntactic processor decides which of the two it should run. This path then runs to completion.

This explains how people can always get a preferred reading and the preferred reading can change with time. Since the non-syntactic processor makes the decision, it will always pick the contextually appropriate reading. Therefore, these do not normally cause the garden path effect in people.

9.4 Why are These not Garden Paths?

We have defined a garden path sentence as one in which the non-syntactic processor makes a decision; this decision was incorrect and the parser failed to complete the parse. In a garden path sentence, if the wrong path is taken, it is impossible to finish the parse. In an example of global ambiguity however, there are two possible paths to a successful parse. The non-syntactic processor merely decides which path to pursue. Either is guaranteed to be successful; so it is not possible to make the wrong choice.

The approach to global ambiguity which is the most compatible with the work in this thesis is: The parser first finds the potential globally ambiguous fragment. This could be done by having two rules match at once in the production system grammar. When both match, the non-syntactic processor decides which rule to run. This is how context effects
these examples and is the same as the theory of non-syntactic interaction presented in Chapter 4.
Using Semantic Information for PP Attachment and VP Parsing

In the previous chapter, we have seen suggestions for future investigation into the implications of the word 'have' and global ambiguity. In this chapter, we will look at areas for further investigation involving non-syntactic information.

I have alluded to the non-syntactic check that is made before a PP is attached, for complex head nouns and reduced relative clauses. In this chapter we will look at how non-syntactic information is used in the "MECHO world" for making decisions in PP attachment and how non-syntactic information could be used in parsing verb phrases. Most of the semantics of PP attachment is implemented as described, but the VP parsing discussed is not implemented.

10.1 Semantic Handling of PPs

In this section, the method by which non-syntactic information in the form of semantic markers is used to make PP attachment decisions in ROBIE for the MECHO world will be explained. This explanation is to assist the reader in understanding how this may work. No particular commitment is felt to the method presented. It is simply adequate to handle PP attachment problems in the MECHO world. A better method could probably be devised quite easily.

The PP semantic check is made each time one of the rules attaching a PP tries to match. Before the pattern matches and the rule runs, the semantic check has to agree that this attachment is desirable. This check tests the bottom node of the stack (the current active NP) and uses information from the non-syntactic interpretation to make its decision. This semantic check question can be thought of as asking "Shall I attach this PP to the bottom of the Active Node Stack?" If the semantic check says "Yes", the PP will be attached to
the bottom of the Active Node Stack. If the semantic check says 'No', then the attachment will not take place, and a later rule will attach the PP to a higher constituent.

The semantic test can return only a yes or a no answer. It cannot recommend to the rule a better place to put the PP. The semantic check can look at the properties of the PP including the preposition and the properties of its NP. In order to handle all examples of PP attachment ambiguity, it should also consider the properties of the item that is at the bottom of the Active Node Stack, the properties of the main verb, and intonation and discourse information. These last two items of information are not used in the current system.

The following example will be used to explain how this is done:

[403] The particle hit the wall with velocity 12 ft/sec.

In this sentence we are concerned with the PP (with velocity 12 ft/sec) and the NP (the wall). The semantic check first isolates the preposition and the two NPs involved. In this example it will use: (with) (the wall) and (velocity 12 ft/sec). ROBIE uses a simple set of semantic markers to make its decision.

For this example, it will use a set of semantic markers called 'can have'. For each word, there is a list of markers reflecting dimensions that it can have. For example:

```plaintext
can_have(wall, [height, length, width, mass]).
can_have(particle, [mass, velocity]).
can_have(spring, [length, mass, constant, elasticity]).
```

These markers state that a wall can have height, length, width and mass, but cannot have velocity. Similarly a particle does not have length or width. The semantic check then sees if the wall 'can have' the headnoun of the NP, 'velocity' in this case. It cannot, so the PP attachment does not take place.

[404] A mass is hung from the spring with a constant of 8 lbs/ft.

In [404], springs can have constants so the PP would be attached to the spring. This particular test does not consider the main verb of the sentence nor the preposition associated with the PP.
This is only one of several strategies ROBIE uses for PP attachment. Some of the other strategies used include always attaching "of" PPs and if an Adjectival Phrase is the head of the PP and it tries to attach to a NP, then it is always attached. This would occur in a sentence such as:

\[ A \text{ stone is dropped from a cliff } 100 \text{ meters above the sea.} \]

Even though this method looks very simple, it makes the correct decision for all the PP attachment problems in Appendix B.

The majority of the PP attachment strategies were developed jointly with McKay [McKay 1981]. This approach was two fold. Every word has a "semantic definition" consisting of semantic markers. The non-syntactic processor checks to see if one head noun is a semantic feature of the other. This is an extended version of 'can/have' described above. The second technique is to have a semantic "specialist" for each preposition. For example, a PP with the preposition "at" will be attached to an NP if the NP is a physical object and the NP of the PP is a location.

These techniques are very simple and counter examples to these strategies are not hard to find. None the less, they produce the correct result for the MECHO problems in Appendix B. A proper PP attachment scheme should take into account the main verb and what types of PP modifiers it needs. It should also take into account the other possible places to attach the PP and make a best fit according to all factors. Woods used a very good method in Lunar [Woods 1973] for this, Selective Modifier Placement (SMP).

The SMP facility computed the possible positions of a modifier. For example, if one has to attach a PP, then the SMP will compute all the possible attachment points. Woods does this by a special form of POP arc called SPOP. The SMP looks up the push-down stack of the ATN and returns all the items on this stack. It then asks which of the items on the stack could take the modifier to be attached. For each of these stack elements, the SMP computes how much it needs the modifier. This measurement was based on a technique similar to the "paraplates" of [Wilks 1975]. The preferred alternative then is "the closest item that needs the modifier the most".
In ROBIE, the elements of the Active Node Stack are items under construction, but work on them has been suspended until a node further to the right is built. When construction begins on a new node, it is pushed onto the Active Node Stack and work is stopped on the old node. This operation has the same effect on the stack as a PUSH. So the alternatives found by the SMP will be the same as the items in the Active Node Stack. If a mechanism was added to ROBIE to assess how much each node needed a modifier, we would have an equivalent of the SMP.

But this would involve looking at the Active Node Stack, which we have stated we will not do. Our point is merely that the SMP is similar to the approach outlined in this section, if this restriction was lifted. However, it is not clear that lifting this restriction is necessary in order to resolve the PP attachment problems.

In Church's YAP [Church 1980], he did not use semantic checking. Instead he "pseudo attached" the modifier to all its possible attachment points (See Chapter 9). These possible attachment points are the same as the elements of the Active Node Stack above. This approach provides parallel attachment of all possibilities and has been discussed in Section 9.3. While this approach gave all possible alternatives at once, I feel that non-syntactic information should be used to decide one unique placement, if one exists.

One further note on the implementation of semantic checks. In Chapter 3, I showed the "plural garden paths" are resolved on the basis of a non-syntactic preference for complex headnouns. The current semantic interpretation for ROBIE is not sufficient to model this completely. In an early version of the parser, this was modeled by having explicit lists of preferred pairs. Preferences were changed by altering this list.

In the current parser, this is avoided by a syntactic heuristic. This heuristic was written only to make these examples easier to model and is called whenever ROBIE does not have a semantic marker pair for the current words. Using the data collected in the pre-test for the first experiment, (Section 3.4.1), the two buffer lookahead of ROBIE is used to resolve this ambiguity.
The heuristic is as follows:

[noun/verb][ngstart or prep or adverb or pronoun] -> verb
[noun/verb][of] -> noun
no headnoun or singular -> noun
[t] -> noun

The last three rules, which will use the word as a noun are really handled as a default. They are inserted only to make these cases explicit.

If there is no headnoun for the NP so far, or the determiner was singular, or if an "of" is next indicating a PP follows, then the word is made a noun. If the word in the second buffer is: "ngstart", "prep", "adverb" or "pronoun", then it will be used as a verb. Otherwise, it will be used as a noun. This heuristic is very accurate and is used only because the current non-syntactic interpretation is insufficient to model this phenomenon effectively.

In Chapter 4, we saw that the reduced relative garden path case is resolved by a semantic test. In the current parser, the non-syntactic interpretation is only the minimum necessary for the MECHO world [Bundy, et al. 1979b]. Hence it is not sufficiently powerful to make the decisions that Crain and Coker have proposed for this. Instead the parser uses a heuristic which says: "one may only have a reduced relative if the sentence has a main verb". This is exactly the strategy of Bever [Bever 1970] as was explained in Section 3.1.

10.2 Parsing VPs with Semantic Information

We can say:

[406] I told a story.
[407] I told Bill a story.
[408] I told Bill to kiss Mary.
[409] I told Bill that Tom hit Mary.
but not:
[410] *I told a rock.
[411] *I told a story Bill
[412] *I told Bill the block.
[413] *I told a story to kiss Mary.
[414] *I told the block that Tom hit Mary.

The first set of sentences are acceptable because "told" is subcategorised for the following list of NPs and complements:
This subcategorisation is very important for several reasons. It can be used to reject ungrammatical sentences, help to find movement gaps, and assist with handling ambiguity.

If a verb does not accept two objects ("kiss" for example), but the sentence has two NPs after the verb, then we know that the sentence would be ungrammatical if the second NP was an indirect object. This is one reason the following sentence is unacceptable:

[415] *The boy kissed the girl a check.

Verb subcategorisation can also assist with finding gaps. If the proper number of obligatory objects is not present and one has a WH-comp that has not found a home, then it can take the place of the missing constituent.

[416] What did Rob give Val?

'Give' requires two objects. Since only one object is present and there is a WH-comp being moved, it can be placed in the gap. This is a simplification of the problem but clearly this subcategorisation is very useful for detecting gaps. It is essential for distinguishing between various types of verbs.

Finally subcategorisation is useful for resolving ambiguities. We have already used it this way in our earlier discussions. We have seen that if a sentence is not subcategorised for a VP and the word 'to' occurs after the verb, it cannot be an auxiliary verb. Similarly for 'that'. If it occurs after a NP in the VP and the verb will not accept 'that' complements, we know it is not a complementiser.

We can use subcategorisation for detecting relative clauses. If the sentence is:

[417] I picked up the coin the boy dropped.

We know that 'picked up' does not take an indirect object in this case, so 'the boy' must be part of a relative clause.
It is also believed [Pulman 1980] that subcategorisation can account for many of the syntactic constraints that have been proposed. It has been suggested, [Pulman ibid], [Gazdar 1980] that the Specified Subject and Tensed S constraints are merely side effects of proper verb subcategorisation. Clearly verb subcategorisation is very important and can assist parsing. So far we have only subcategorised verbs on the basis of syntactic constituents.

This is not the whole story. [410-414] are ungrammatical sentences but meet the subcategorisation that we have given. All of these examples are unacceptable for semantic reasons. In [407], the first NP must be a listener and the second NP must be something to be related, such as a story. The reason each of these is ungrammatical is that the NP must also meet certain semantic restrictions. In other words, verbs should be subcategorised semantically as well as syntactically. This is commonly known as 'selectional restrictions'.

The current parser does not make extensive use of selectional restrictions, which constitutes a major defect. A truly psychologically plausible parser should have selectional restrictions in the rule patterns. In the current parser, these sentences are rejected by the semantic analysis as soon as the rule is run. This gives the same result as having the restriction in the pattern; i.e., the parse fails.

In other words, the pattern for 'told' could be:

\[
\text{told} \ [\text{NP, listener}] \\
\text{told} \ [\text{NP, listener}] \ [\text{NP, story}] \\
\text{told} \ [\text{NP, animate}] \ [\text{VP, action}] \\
\text{told} \ [\text{NP, listener}] \ [\text{that, fact}]
\]

Investigation of this area is beyond the scope of this thesis, but there is other work in this area that I feel should be added directly onto the current parser to remedy this deficiency.

10.3 Preference Semantics

10.3.1 Wilks
Wilks [Wilks 1972, 75a, 75b, 78] has proposed a very good system for semantically parsing sentences based on preference.

Wilks's system was aimed at machine translation and designed explicitly to handle the problems of word sense. The system did not perform a syntactic analysis, even though it did use some syntactic information. Instead it was based on the concept of "preference semantics", where "preference" is distinguished from "strict requirement".

Wilks used several levels of structure, the lowest being "semantic primitives". He had 80-100 primitives, which formed the basic building blocks of the "semantic formulas". Each formula expressed the meaning of the word with which it was associated. Every sense of each word has a separate semantic formula. These formulas had a rigorously defined syntax.

The next level was the "bare template". Each bare template represented a specific "underlying message", having ACTOR, AGENT, OBJECT, etc positions. These represented skeletal propositions. These bare templates were then expanded to "full templates" by filling these positions in the template with the semantic formulae of the corresponding words. These templates corresponded roughly to simple sentences.

When Wilks's program analysed a sentence, the initial word string was first fragmented into template boundaries. The fragments of the surface text were then expanded into full templates, these gave the possible combinations between the word-sense formulae and the surface strings. Links were then established between the main elements of the templates and their possible dependents.

For example in "John drinks water", "drink" expects an animate subject and a fluid object. Likewise for "sly fox": "sly" can modify only animate entries. The bare template for "drinks" would be [*pot cause *ent]. (*pot is potential actor and *ent is an entity.) This bare template would match the above sentence. The template having the most links is the preferred interpretation.

For example [Wilks 1975]:

227
The formula for "interrogate" defined the preference \textit{\*pot force \*ent}. There were two possible templates (\textit{MAN FORCE MAN}) and (\textit{MAN FORCE THING}) produced by this sentence. The primitive "\*ent" could be associated with "MAN", but not "THING", so the template (\textit{MAN FORCE THING}) was eliminated. In this way, the meaning of "crook" as a "walking stick" was eliminated. It can be seen that this is very similar to selectional restrictions.

It is this part of Wilks's work that is of interest here. This approach gives a semantic representation for the sentence and, at the same time enforces selectional restriction and handles word sense ambiguity.

Wilks used syntactic information only in his fragmentation routine to split the fragments. I suggest that ROBIE could find the fragments and words so that a modified version of his templates could then be used to remove extra word senses and handle selectional restrictions.

10.3.2 Boguraev

Boguraev [Boguraev 1979] used Wilks's theories as the starting point in his work, but added syntax and several other changes. His system was designed to paraphrase English sentences to demonstrate its understanding of word sense ambiguity. In this section, we are only concerned with the analysis part of his program.

Boguraev's semantics look very similar to Wilks's semantics and many of the same levels of semantic structure are identifiable. Examples of Boguraev's definitions for "grasp" and "crook" are [Boguraev 1979, p. 3.3]
An explanation of the details of this diagram would be extremely complex. For each possible word sense, Boguraev had a "semantic formula" like those above. The process of disambiguation was one of choosing the correct dictionary entry.

We will use an example from [Boguraev 1979, p. 324] to illustrate his method of disambiguation. Consider the sentences:

[419] John asked Mary a question.
[420] John asked Mary to come with him.

The dictionary entries for "ask" are:

ask1 (inquire):
   ((man subj) (**ani obje ask))
ask2 (want):
   ((man subj)
    ((act obje)
     (((man (please feel)) cause) goal) ask))
ask3 (request):
   ((man subj)
    ((**ent obje) (**hum from want))

It can be seen that each sentence above fits a different definition of "ask". By matching the bare templates of the words with the dictionary definition of the verb, it is possible to arrive at a single meaning of the sentence. If two meanings are possible, then both templates would be matched and both meanings returned.

Boguraev used an ATN-style parser to recognise syntactically well-formed constituents and coordinate the semantic routines which constructed semantic representations for the semantically valid ones. The major addition to Wilks's approach is the addition of "contextual verb frames". This is a semantic pattern that operates on the semantically represented constituents that have been recognised by the ATN. These use patterns as below
and, based on the constituents, helped to guide the parse and remove possible templates. For example the frames for "ask" would be: [Boguraev 1979, p. 3.34]

frame1:
*hum ASK (*hum) (@sign) (ABOUT *ent)

frame2:
*hum ASK *hum TO *do (@act)

frame3:
*hum ASK (*hum) FOR *ent

Here upper case words are keywords ('words as they appear in the text') and brackets denote optionality. These frames are the central part of the parser and guide the parse.

These frames correspond very closely to verb subcategorisation, but with selectional restrictions on the NPs. As stated in the previous section, this is desirable for VPs. Instead of subcategorising verbs as I have done in ROBIE, I feel that it would be beneficial if the system described here were fitted onto it. The rejection of selectional restriction violations could happen just as it does in Boguraev's system.

Word sense definitions could be resolved at the semantic level, as Boguraev does in his system. The syntactic portions would remove the part of speech ambiguity and the semantic routine could decide on the sense of the word. Whilst I am not committed to any particular semantic system, the addition of a system such as this shows that it is possible to build appropriate semantic analysis programs.
Conclusion

This work was an investigation into part of the human sentence parsing mechanism (HSPM), where parsing involves syntactic and non-syntactic analysis. It was proposed that the HSPM consists of at least two processors. We called the first processor the syntactic processor, and the second the non-syntactic processor. The syntactic processor is unconscious, deterministic and fast, but limited. The resolution of lexical ambiguity was used as a vehicle to investigate this hypothesis. We then saw that the two processors could work in parallel during the processing of a normal sentence, with the non-syntactic processor "listening" to the syntactic processor. During processing of some sentences, the syntactic processor, at key points, could ask the non-syntactic processor to make a decision in order to resolve an ambiguity. These key points occur whenever a situation arose in which the syntactic processor could no longer guarantee a correct analysis. A major focus of this research was the identification of those situations in which people use the non-syntactic processor to assist with the resolution of ambiguity. It was shown that these situations can be correctly predicted using the two buffer lookahead of the syntactic processor.

Only the syntactic processor has been investigated. A major test of the psychological validity of this model was that it failed on precisely those sentences that humans find to be garden paths. As a starting point, we used Marcus's work on deterministic parsing. The advances reported here were:

- Reaction time experiments were used to provide a non-subjective classification of sentences as garden paths or not. Using this classification, it was shown that Marcus's parser would succeed on some garden path sentences and fail on some non-garden path sentences.

- This deficiency can be corrected by the use of non-syntactic information for ambiguities which may lead to a garden path.
- Non-syntactic information is to be used to help resolve an ambiguity when the syntactic processor can no longer guarantee a correct analysis. All other ambiguities are to be resolved on the basis of syntactic information.

- An amended parser, ROBIE was presented which incorporates these conclusions. It was shown that the situations in which non-syntactic information is to be used can be accurately predicted by ROBIE's two buffer lookahead. ROBIE was shown to be compatible with the psychological evidence currently available on human sentence comprehension.

- ROBIE is computationally and conceptually simpler than Marcus's parser.

Perhaps the most significant result of this thesis is the identification of the situations in which the syntactic and non-syntactic processors interact. These occur whenever an ambiguity arises which the syntactic processor cannot guarantee to resolve correctly, because of its limitations. This is contrasted with a theory which might suggests that the non-syntactic processor is used only when the syntactic processor has been led astray. Rather than using the non-syntactic processor only when an ambiguity has taken the syntactic processor astray, the non-syntactic processor is used when an ambiguity arises which might lead the syntactic processor astray.

11.1 Summary

We first looked at cases of lexical ambiguity that can lead to garden paths. In the first few chapters, we investigated the garden path prediction of Marcus's PARSIFAL. We saw that it was inadequate and so a new theory of garden path sentences based on non-syntactic interaction was presented. This theory made several predictions which were tested and confirmed using a reaction time experiment. We then investigated WHEN does the non-syntactic processor make a decision and WHY at those times.

It was shown that the syntactic processor does not have enough information in its limited lookahead always to resolve some ambiguities correctly. When a situation such as
this arises, the non-syntactic processor is called upon to choose one alternative, the syntactic processor making no attempt to guess the correct alternative. This theory provided a better prediction and explanation of garden path sentences. It also allowed us to explain how preferred readings of ambiguous sentences can change and to develop a hypothesis of how people resolve global ambiguity.

In the next sections, we investigated cases of lexical ambiguity that do not lead to garden paths. We saw that it was possible to handle most of these cases of part of speech ambiguity without any special rules or mechanisms. This was accomplished by enforcing agreement in ROBIE, to promote the rejection of ungrammatical sentences. We saw that by exploiting agreement and well-formedness, we could handle, in a simple way, many ambiguities that Marcus required special mechanisms to handle.

Marcus’s parser was modified and restricted to accommodate the changes above. We saw that it was possible to parse English deterministically, ignoring conjunction and unbounded movement, with only two adjacent buffers for rule matching and no Attention Shift. This resulted in a computationally and conceptually more simple parser.

Finally, the model, as developed to handle lexical ambiguity, was compared with the relevant psycholinguistic literature. The parser here presented was shown to have many advantages over current proposals for the HSPM. We also saw that Gazdar’s Phrase Structure Grammar has several advantages over Chomsky’s Extended Standard Theory for the issues discussed here.

11.2 The Parser vs. The Grammar

This thesis has presented some linguistic, psychological and computational advances over other models of the HSPM. One important question is: which of these advances are a function of the grammar and which a function of the parsing mechanism? That is, which results depend on the grammar that is used and which results depend only on the parsing mechanism?
I feel that the results presented here are all a function of the grammar, but that the parser's grammar is itself a function of the parsing mechanism. The main influence of the parsing mechanism is to put several constraints on the grammar.

Firstly, the grammar must be organised in such a way that it can be interpreted deterministically. Secondly, the rules of the grammar must be constrained such that they match only two buffers. Finally, it is assumed that it is only possible to produce grammar rules that can be interpreted by the parser. This puts a heavy constraint on the type of grammars that are possible. For the HSPM this means that some possible rules are not learnable. For a computer program, this means that some rules are not implementable.

We have seen several examples of these constraints in this thesis. Because ROBIE cannot backtrack, it was not possible to implement a strategy based on backtracking to handle a particular ambiguity. Instead the grammar rules for each ambiguity were written so that backtracking was not necessary.

In Section 5.2.3, we discussed the EST use of 'for' as a complementiser. In that section we saw that it was not possible to write a grammar rule using only two buffer lookahead to distinguish this use from the preposition use. We also saw that it seemed possible to implement the PSG analysis of those examples with two buffer lookahead.

In Section 9.2 we discussed several problems with the EST analysis of 'have' as an imperative versus a yes-no-question. We again saw that it was impossible to formulate the EST analysis in a two buffer deterministic parser. We discovered that the PSG analysis was compatible with our two buffer lookahead.

These two problems illustrate how the parser mechanism can constrain the grammar. For both of these examples, the EST analysis could not be implemented in a two buffer deterministic parser. These considerations provide a new ground on which to evaluate grammar theories. The type of parsing mechanism needed to parse a particular grammar is very important. In this thesis it is suggested that some grammars can be parsed with a two buffer deterministic parser, whilst others cannot.
Once the grammar is written, the parser is only a simple device that interprets it. Assuming a parser contains only legal rules, then violations of the principles presented here are not possible. The parsing mechanisms ensures that there are only legal rules. Hence, whilst the parsing mechanism influences the grammar, the grammar is everything.

11.3 Areas for Future Study

There are still many important problems that need to be investigated and areas that should be fertile for future research.

This thesis has considered only the English language. A very interesting question is: "Is it possible to parse all other languages deterministically?" The answer is not unknown. It would be very interesting to try to write a grammar using this deterministic parser for several languages and observe how well the principles will hold true.

One must remember that for many languages the linguistic analysis is not necessarily agreed upon. This paper does not consider the exact linguistic analysis of other languages. There are three key points, discussed in this thesis, which may be valid in other languages.

1) the production system type grammar and some form of top-down parsing sub-categorisation.

2) the parser structure, that is a stack of partially built constituents and the two lookahead buffers.

3) The timing of non-syntactic interaction at those situations in which the syntactic processor cannot choose the correct alternative because of its limited lookahead.

It would be interesting to design parsers for several languages, whilst obeying these three points.

In Chapter 5, it was suggested that the amount of ambiguity in a language is inversely related to the strictness of word order and complexity of inflections in that language at both the clause and constituent level. This interesting hypothesis could also be
It would be interesting to write a grammar based on the Phrase Structure Grammar of [Gazdar, Pullman and Sag 1980c]. This may reveal several things. First, whether it is possible to write a more elegant grammar in this way? Second, by using the category features, handling conjunction and WH movement could be much easier. Thirdly, could the packets be eliminated or reduced by the category grammar?

The problems of global ambiguity also need further research and more data needs to be collected. It is clear that context can affect how people deal with global ambiguity, but do people really come up with two readings in a situation where the non-syntactic processor has no preference and do these situations cause any garden path effect? It was proposed that the non-syntactic interaction theory should apply in this case. This hypothesis needs to be tested as do the 'have' theories and the predictions in Section 6.7.

In Section 6.7.2, an experiment was suggested which would test the prediction of the two buffer lookahead as it relates to lexical access. In order to establish the validity of the two buffers, this and similar experiments should be performed.

An adequate technique for WH movement and gapping in a deterministic parser needs to be found. Perhaps this could be tied in with using Phrase Structure Grammar. It is often simple to know that movement has taken place, but what is the best way to move the trace and what is the best way to detect gaps?

In Section 4.6, a method to recover from a garden path situation was outlined. The ideas presented there need to be developed further and the nature of the error recovery component needs to be investigated. Deterministic parsing depends on the existence of some form of error component, making this topic especially important.

Finally, any psychologically plausible system needs to make some comments about handling fragments and ill-formed utterances. I have relied heavily in this paper, on the use of grammaticality and rejecting ungrammatical utterances, but people can understand ungrammatical utterances. Little is known about this area and data needs to be collected.
Appendix A: An Annotated Example

To help the reader understand how the parser works, I will go step by step through the parse of "The shy boy has kissed Mary." I will show a "snapshot" of the parser before each rule is applied. These snapshots are taken from an actual parse. In this form of tracing, the features of the words and nodes are not printed, as they would clutter the snapshots.

Each snapshot is taken just before the parser runs the rule mentioned. These show the Active Node Stack, with "1:" being the bottom, or the Current Active Node. Beneath it are the two buffers, noted by B1: and B2:. The item < open> indicates that the node with which it is associated is still "open," that is, it may not have all its daughters. To the right of each node on the Active Node Stack are the packets that are active when that node is the Current Active Node. On the right of each rule name is the pattern for that rule.

To parse a sentence the user first types "go." The system will then prompt with "Sentence:" The user then types in the sentence as seen below. The packet and rule names have been copied from [Marcus 1980]. These rules are very similar to his original rules.

```
| go.
Sentence: The shy boy has kissed Mary.
```

Packet: CPOOL pattern: [ngstart], agree(det)
Rule about to run: MARKED-STARTNP
Active Node Stack:
1: < open> S [SS-START,CPOOL]
B1: the
B2: shy

Initially the parser generates an S node and activates the packets SS-START and CPOOL. The buffers are the next two nodes from the bottom of the Active Node Stack, in this case the next two words. The first rule to match is the rule MARKED-STARTNP. This matches the feature "determiner" in the first buffer and will create a new NP node.

```
Packet: PARSE-DET
Rule about to run: DETERMINER pattern: [det]
Active Node Stack:
2: < open> S [SS-START,CPOOL]
1: < open> NP [PARSE-DET,NPOOL]
B1: the
B2: shy
```

One can now see that the NP node is at the bottom of the Active Node Stack. The rule MARKED-STARTNP created the NP node. The rule about to run will attach the word "the" as the determiner. The packet NPOOL contains rules to locate number phrases inside the NP.
Packet: PARSE-QP-2  
Rule about to run: DETERMINER-DONE  
Active Node Stack:
2: <open> S  
1: <open> NP det-the  
  [SS-START,CPOOL]  
  [PARSE-QP-2,NPOOL]  
B1: shy  
B2: boy  
The parser now deactivates the packet PARSE-QP-2 which contains rules for 
varying pre-noun modifiers. Since there are none, a default rule runs, deactivating this 
packet and activating the packet PARSE-ADJ.

Packet: PARSE-ADJ  
Rule about to run: ADJECTIVE  
Active Node Stack:
2: <open> S  
1: <open> NP det-the  
  [SS-START,CPOOL]  
  [PARSE-QP-2,NPOOL]  
B1: shy  
B2: boy  
The rule ADJECTIVE in the packet PARSE-ADJ has now matched, and will 
attach the word 'shy' to the NP being built.

Packet: PARSE-ADJ  
Rule about to run: ADJ-DONE  
Active Node Stack:
2: <open> S  
1: <open> NP det-the  
  [SS-START,CPOOL]  
  [PARSE-QP-2,NPOOL]  
B1: boy  
B2: has  
The adjective has been attached and the default rule in PARSE-ADJ will deac-
tivate the packet PARSE-ADJ and activate the packet for parsing nouns. If there were 
more adjectives, they would have matched the above state since the rule was still active.

Packet: PARSE-NOUN  
Rule about to run: NOUN  
Active Node Stack:
2: <open> S  
1: <open> NP det-the  
  [SS-START,CPOOL]  
  [PARSE-QP-2,NPOOL]  
B1: boy  
B2: has  
The rule NOUN in PARSE-NOUN has now matched the [noun] feature of 'boy' 
and it will be attached to the NP. If the noun could be any other parts of speech, they would 
no longer be carried forward.
At this point the NP is completed, and the packet NP-COMPLETE is active. This contains rules to find any post-modifiers for the NP such as a PP or a relative clause. Since there are no modifiers, the packet is deactivated by the rule NP-DONE and the finished NP is dropped into the first buffer.

The S node is now the Current Active Node, with the NP in the first buffer. The next rule to match has the pattern [np][verb]. The parser can now decide that this is a declarative sentence. It adds the features "ded-s, major" to the S node, deactivates the SS-START packet and activates the packet to parse the subject.

The packet PARSE-SUBJ will now attach the NP as the subject of the current S node and activate the packet for parsing the auxiliary.

The rule START-AUX in PARSE-AUX will match with the feature, [verb], and start a new AUX node. This also sets the tense of the sentence and will automatically transfer the tense and number from the auxiliary verb to the AUX node. The S node is no longer the Current Active Node, so the rules in the packets associated with it cannot match.
PackEt: BUILD-AUX
Rule about to run: PERFECTIVE   pattern: [have][en]
Active Node Stack:
2:  <open>  S NP det-the   [PARSE-AUX,CPOOL]
     adj-shy
     noun-boy
1:  <open>  AUX            [BUILD-AUX]
B1:  has
B2:  kissed

The rule PERFECTIVE now runs. This has the pattern, [have][en], which
matches the current state of the two buffers. It will then attach 'has' to the AUX node.

Packet: BUILD-AUX
Rule about to run: AUX-COMPLETE   pattern: []
Active Node Stack:
2:  <open>  S NP det-the   [PARSE-AUX,CPOOL]
     adj-shy
     noun-boy
1:  <open>  AUX            [BUILD-AUX]
B1:  kissed
B2:  Mary

The AUX node is now completed, and no other rule in BUILD-AUX will match,
so the default rule, AUX-COMPLETE matches. This rule will 'move' the AUX node from
the bottom of the Active Node Stack to the first buffer, signifying that it is finished.

Packet: PARSE-AUX
Rule about to run: AUX-ATTACH   pattern: [aux]
Active Node Stack:
1:  <open>  S NP det-the   [PARSE-AUX,CPOOL]
     adj-shy
     noun-boy
B1:<open>  AUX has
B2:  kissed

One can see that the buffers have automatically shifted so that they are the first
two items after the bottom of the Active Node Stack. The rule AUX-ATTACH will now
attach the AUX node to the S node which is active. Note that the AUX node was closed
when it was attached. The rule also activates the packet to start the verb phrase.

Packet: PARSE-VP
Rule about to run: MAIN-VERB   pattern: [verb]
Active Node Stack:
1:  <open>  S NP det-the    [PARSE-VP,CPOOL]
     adj-shy
     noun-boy
     AUX-has
B1:  kissed
B2:  Mary

The rule MAIN-VERB will now match the pattern, [verb], in the first buffer.
This will start a new VP node, activate the appropriate packets and attach the verb as the
main verb. The packets activated are for finding various complements such as a toVP or an
S-.
Packet: CPOOL
Rule about to run: PROPN AM
Active Node Stack:
2: <open> S NP det-the [SS-FINAL,CPOOL]
adj-shy
noun-boy
AUX-has
1: <open> VP verb-kissed [SS-VP,CPOOL]
B1: Mary
B2: 

At this time, the feature [name] will match the rule PROPN AM in the packet CPOOL. This packet has been active all the time and handles clause level items. It is very important to have several packets active at once in this way. This rule will make “Mary” into a NP in the first buffer.

Packet: SS-VP
Rule about to run: OBJECTS pattern: [np]
Active Node Stack:
2: <open> S NP det-the [SS-FINAL,CPOOL]
adj-shy
noun-boy
AUX-has
1: <open> VP verb-kissed [SS-VP,CPOOL]
B1: <open> NP name-Mary
B2: .

The rule OBJECTS will now match the NP in B1 and attach it as the object of the VP.

Packet: SS-VP
Rule about to run: VP-DONE pattern: []
Active Node Stack:
2: <open> S NP det-the [SS-FINAL,CPOOL]
adj-shy
noun-boy
AUX-has
1: <open> VP verb-kissed [SS-VP,CPOOL]
NP name-Mary
B1: .
B2: <null>

No rule in SS-VP or CPOOL matches on the final punctuation, so the default rule VP-DONE will run and drop the VP into the first buffer.

Packet: CPOOL
Rule about to run: VP-ATTACH pattern: [vp]
Active Node Stack:
1: <open> S NP det-the [SS-FINAL,CPOOL]
adj-shy
noun-boy
AUX-has
B1: <open> VP verb-kissed
NP name-Mary
B2: .

241
The rule VP-ATTACH will attach the VP to the S node, which is again the active node, and the parser will be in the state below.

Packet: SS-FINAL
Rule about to run: S-DONE pattern: [finalpunc]
Active Node Stack:
1: <open> S NP det- the [SS-FINAL,CPOOL]
   adj- shy
   noun-boy
   AUX-has
   VP verb-kissed
   NP name-M ary
B1: .
B2: <null>

The rule S-DONE in SS-FINAL now matches the final punctuation and terminates the parse. The parse tree is then printed:

S-1 [s,major,decl]
   NP-1 [np,def,ns,n3p]
      DET
      THE [det,def,ns,n3p]
      ADJ
      SHY [adj]
      NOUN
      BOY [noun,ns,n3p]
   AUX-1 [aux,past,v3s]
      AUXVERB
      HAS [auxverb,past,v3s,verb]
   VP-1
      VERB
      KISSED [verb,en,past,vspl]
   NP-2 [np,name,ns,n3p]
      NAME
      MARY [name,propnoun,ns,n3p]
Appendix B: Some Example Sentences

The following examples show some of the linguistic coverage of the parser. All of these can be successfully parsed by the parser. In addition, the sentences in Section 4.8 illustrate further examples that can be successfully parsed by the parser.

The movement stack, as described in Section 8.5.3, is used with each sentence that involves auxiliary inversion or relative clauses with moved objects. The sentences with conjunction are the only sentences which require three buffers to process. Sentences which require these extensions are given in the second half of each section. This division is based on the actual execution of the full parser. All other sentences can be parsed using only the two buffers and no additional mechanisms.

Examples by Milne

Tom found her.
Tom found her dog.
The block will block.
What block hit her.
What hit her.
Which boy kissed her.
Which man kissed her?
The boy which kissed her kissed Mary.

In the park, tom hit mary.
I believe you want to kiss me.
I want you.
I want you to leave.
I want you to kiss me.
I want for you to leave.
I saw tom.
I saw tom hit mary.

The trash can be taken out.
The trash can was taken out.
The paper will be destroyed.
The paper will was destroyed.

Jack is 4 times heavier than Mary.
If they gain 20 pounds Jack will be 3 times heavier than Mary.

Examples by Milne with Movement

The boys will be leaving next week.
Was the boy's mother's block broken by Tom?
Have the boys taken the exam?
Is the last boy running down the street?
The boy which I saw kissed her.
Who kissed Rob?

Tom hit the boy and the girl.
Tom hit Mary and kissed Sue.
Rob hit the girl on the hill and in the park.

Jack is 10 years old.
Jill is 2 years younger than Jack.
How old is Jill?

Jack is 4 times older than Mary.
In 5 years Jack will be 3 times older than Mary.
How old is Jack?

PP ATTACHMENT using Semantic Markers

A mass is connected to a spring with a constant of 5 lbs/ft.
A mass is connected to a spring with a light string.
The particle hit the ball with velocity 5 ft/sec.
The particle hit the wall with velocity 5 ft/sec.
The boy in the park on the hill saw Mary.
The mass of the particle of mass 3 lbs is falling.

13.1 Examples from Marcus

The following examples are from [Marcus 1980]. These are the examples his parser could do which ROBIE can as well.

I told that boy that boys should do it.
The pencil seems broken.
There seems to be a pencil broken.
I wanted John to do it.
I want to do it.
I persuaded John to do it.
Schedule a meeting for Friday.
A meeting seems to have been scheduled for Friday.
I told the boy that I saw sue.
I told sue you would schedule a meeting.
I told the girl that you would schedule the meeting.
The boy who wanted to meet you scheduled the meeting.
The boy who met you scheduled a meeting.
I promised John to do it.
You promised to give the book to John.

Examples from Marcus involving Movement.

The boy who you met scheduled the meeting.
Who did John see?
Who broke the pencil?
What did Rob give Sue?
Who did Rob give the book?
What did Rob give Sue?
What did Rob give to Sue?
There seems to have been a meeting scheduled for Friday.
Is there a meeting scheduled for Friday?
Does there seem to be a meeting scheduled for Friday?
Who did you say that Tom told?
What did you give Sue yesterday?
What did you give the book yesterday?

13.2 Examples from Church

These examples are from [Church 1980] and show which of the examples his parser can parse which ROBIE can parse as well.

It seems likely that John would be sitting.
There seems to be a block in the table.
That I might take a ball seems likely.
For me to take a ball seems nice.
To take a ball seems nice.

I wonder what to do?
I wonder what I should do?
I wonder what should have been done?

I know a man that was nice.
I know that was nice.
I know that that was nice.
I know that boys are nice.
I know that boy is nice.
I know that he is nice.

That he is nice is a fact.
That that boy is nice is a fact.
That that is nice is a fact.
That that block was red surprised me.

Examples from Church involving Movement.

Which boys have the girls taken to the ball?
Which boys were the girls taking to the ball.
Which boys have the girls take the pencils?
What will he do to you?
What to do?
Which boy did you see?
Which is the ball?

Which boys and girls went?
Which boys and which girls went.
Which boys went to the ball and took the pencil?

Who do you believe that was?
Who do you believe that that was?
Did you believe that?
Did you believe that that was him?
Did you believe that was him?
Did you believe he did that?

Did you say that?
Did you say that to him?
Did you say that Tom left?
The boy that hit ran away.
The boy that hit Mary ran away.
A hammer of mass 2 kg travelling at 15 ms\(^{-1}\) is brought to rest when it strikes a nail. What impulse acts on the hammer?

(from Bostock and Chandler 1975)

A small object of weight 10 N rests in equilibrium on a rough plane inclined at 30 degrees to the horizontal. Calculate the magnitude of the frictional force.

(from Bostock and Chandler 1975)

A stone is dropped from a cliff 100 m above the sea. Find the speed with which it hits the sea.

(from Bostock and Chandler 1975)

A ball is thrown vertically upward to a height of 10 m. Find the time taken to reach this height and the initial speed of the ball.

(from Bostock and Chandler 1975)

A stone is projected vertically upward with a speed of 21 m\(s^{-1}\). Find the distance travelled by the stone in the first 3 s of its motion.

(from Bostock and Chandler 1975)

A ball is thrown vertically upward with a speed of 15 m\(s^{-1}\) from a point which is 1 m above ground level. Find the speed with which the ball hits the ground.

(from Bostock and Chandler 1975)
A stone is dropped from the top of a tower.
In the last second of its motion it falls through a distance
which is $\frac{1}{5}$ of the height of the tower.
Find the height of the tower.

(from Bostock and Chandler 1975)

A stone is dropped from the top of a building 20 m high.
A second stone is dropped from a point half-way up the same building.
Find the time that should elapse between the release of the two
stones if they are to reach the ground at the same time.

(from Bostock and Chandler 1975)

A particle which is moving in a straight line with constant
acceleration takes 3 s and 5 s to cover two successive distances
of 1 m.
Find the acceleration.

(from Bostock and Chandler 1975)

A lever 10 ft long is pinned at its left end.
The lever is supported by a spring with a constant
of 40 lb/ft.
The spring is attached 6 ft from the left end of the lever.
A weight of 20 lb is attached at the other end of the lever.
The weight of the lever is 8 lb.
How much is the spring stretched?

Where must a weight be hung on a pole, of negligible weight,
so that the boy at one end supports $\frac{1}{3}$ as much as the man at
the other end?

(from Novak 1976)

A scaffold 10 ft long is supported by ropes attached at each end.
The scaffold weighs 100 lb.
One painter weighing 150 lb stands on the scaffold 4 ft from
one end, while a second painter weighing 175 lb stands on
the scaffold 2 ft from the other end.
What is the tension on each of the ropes supporting the scaffold?

(from Novak 1976)
A horizontal uniform bar 10 m long is supported by two ropes attached at its ends. The rope on the left end makes an angle of 45 degrees with the horizontal, while the rope on the right end makes an angle of 60 degrees with the horizontal. A weight of 100 nt is attached 2 m from the right end of the bar. What is the weight of the bar?

A uniform scaffold 12 ft long and weighing 100 lb is supported horizontally by two vertical ropes hung from its ends. Find the tension in each rope when a 180 lb painter stands 4 ft from one end.

(from Novak 1976)

A uniform bar B-C is 100 cm long and weighs 50 lb. The bar is to be supported at ends B and C. An upward force of 40 lb is applied 80 cm from B. Compute the forces on the supports.

(from Novak 1976)

A uniform pole 20 ft long and weighing 30 lb is supported by a boy 3 ft from one end and a man 6 ft from the other end. At what point must a 150 lb weight be attached so that the man supports twice as much as the boy?

(from Novak 1976)

The hinges of a door weighing 20 lb are 12 ft apart, and the door is 3 ft wide. The weight of the door is supported by the upper hinge. Determine the forces exerted on the door at the hinges.

(from Novak 1976)

A bridge is 80 ft long. What force must the pier at each end of the bridge exert to support an automobile weighing 2 tons which is 30 ft from one end of the bridge?

(from Novak 1976)

A gun has a maximum range of 200 m on the horizontal. Find the velocity of a shell as it leaves the muzzle of the gun.

(from Bostock and Chandler 1975)
The greatest range of a particle, with a given velocity of projection, on a horizontal plane is 3000 metres. Find the greatest range up a plane inclined at 30 degrees to the horizontal.

(adapted from Humphrey 1930)

Two particles of mass B and C are connected by a light string passing over a smooth pulley. Find the acceleration of the particle of mass B.

A particle of mass 4 kg rests on a smooth horizontal table. It is connected by a light inextensible string passing over a smooth pulley at the edge of the table to a particle of mass 2 kg, which is hanging freely. Find the acceleration of the system and the tension in the string.

(from Bostock and Chandler 1975)

A particle of mass 5 kg rests on a rough horizontal table. It is connected by a light inextensible string passing over a smooth pulley at the edge of the table to a particle of mass of 6 kg, which is hanging freely. The coefficient of friction between the 5 kg mass and the table is 1/3. Find the acceleration of the system and the tension in the string.

(from Bostock and Chandler 1975)

Two particles of mass 3 kg and 4 kg are connected by a light inextensible string passing over a smooth fixed pulley. The system is released from rest with the string taut and both particles at a height of 2 m above the ground. Find the velocity of the 3 kg mass when the 4 kg mass reaches the ground.

(from Bostock and Chandler 1975)

Two particles of mass 3 kg and 5 kg are connected by a light inextensible string passing over a smooth pulley which is fixed to the ceiling of a lift. Find the tension in the string when the system is moving freely, and the lift has a downward acceleration G ms⁻².

(from Bostock and Chandler 1975)
The driver of a car travelling due East on a straight road at 40 kmh-1 is watching a train moving due North at 75 kmh. What is the apparent speed and direction of motion of the train?

(from Bostock and Chandler 1975)

A particle of mass \( M_1 \) is suspended from the end of a spring of length \( L_1 \) and elasticity \( E \).
A second spring with length \( L_2 \) and elasticity \( 2E \) is attached to the first particle, and another particle of mass \( M_2 \) is suspended from the second spring.
Find the extension of each spring.

A light elastic string of unstretched length \( A \) and modulus of elasticity \( W \), is fixed at one end to a point on the ceiling of a room.
To the other end of the string is attached a particle of weight \( W \).
A horizontal force \( P \) is applied to the particle and in equilibrium it is found that the string is stretched to three times its natural length.
Calculate the angle the string makes with the horizontal, and the value of \( P \) in terms of \( W \).

(A-level exam (part): U of L)

A block of mass 500 kg is raised a height of 10 m by a crane.
Find the work done by the crane against gravity.

(from Bostock and Chandler 1975)

A train has a maximum speed of 80 kmh-1 on the level against resistance of 50000 N.
Find the power of the engine.

(from Bostock and Chandler 1975)

A car has a maximum speed of 100 kmh-1 on the level with the engine working at 50 kW.
Find the resistance to motion.

(from Bostock and Chandler 1975)

A lorry of mass 10000 kg has a maximum speed of 24 km/h up a slope of 1 in 10 against a resistance of 1200 newtons.
Find the effective power of the engine in kilowatts.

(A-level exam (part): U of L)
A train x1 moves from a station x2 to a station x3 with a uniform acceleration of 5 ft/sec. x1 is 2000 meters from x2. What is the final velocity of x1?
Appendix C: Free Text Analysis

To check the coverage of the ambiguity handling techniques I have explained in this thesis, several free text simulations were done. These include, the local newspaper, 'The Scotsman', [Scotsman 1981] an international magazine 'TIME', [TIME 1978, 81, 81b], the 100 MECHO problems (see Appendix B) and the ASHOK corpus from [Martin, Church, and Patil 1981]. This analysis showed no violations of importance to these techniques.

As the grammar and dictionaries are not large enough to handle all the sentences found in these texts, I have hand-simulated the rules. This hand simulation is very accurate as careful thought goes into all marginal cases.

For each word I discussed in Chapter 5, I will list the number of occurrences in the texts and any problems. This includes: 'to', 'for', 'have', 'what', 'which', and 'that'.

These texts are all fairly large. There are 276 sentence in the MECHO problems with an average number of words per sentence of 14. This makes about 4000 words total. The ASHOK corpus is larger than this. The first TIME articles was about 5000 words and the second TIME article contained 4000 words.

14.1 HAVE

The five sources were checked for violations of our 'have' as a yes-no-question versus an imperative rule. There are no sentence initial examples of 'have' in the MECHO sentences. There are three sentence initial examples of 'have' in the ASHOK corpus. All three are yes-no-questions. In the Scotsman and both TIME articles, there were no sentence initial usages of 'have'. One can see that the use of 'have' for yes-no-questions is rare and the use of it as an imperative is extremely rare. It seems that the problem we have concerned ourselves with does not occur in free text.

14.2 TO

The five sources were checked for conflicts between 'to' as an auxverb and 'to' as a preposition. In over 300 occurrences of the word 'to', there were no violations. The following table shows the occurrences of the word 'to'.

<table>
<thead>
<tr>
<th>Source</th>
<th>Number of Occurrences</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scotsman</td>
<td>96</td>
<td>none</td>
</tr>
<tr>
<td>ASHOK</td>
<td>34</td>
<td>none</td>
</tr>
<tr>
<td>TIME [81]</td>
<td>55</td>
<td>none</td>
</tr>
<tr>
<td>TIME2 [81b]</td>
<td>102</td>
<td>none</td>
</tr>
<tr>
<td>MECHO</td>
<td>80</td>
<td>none</td>
</tr>
</tbody>
</table>
14.3 FOR

The sources were checked for conflicts between "for" as a preposition and "for" as a complementiser. All occurrences of "for" in the M.ECHO corpus were as a preposition. The Scotsman had 16 occurrences of "for". The one possible exception is:

"An act was hurried through Parliament suspending a 10 day rule for cases to be brought to trial."

But this is a nominal modifier problem. There were a surprising 224 occurrences of "for" in the ASHOK corpus. All were used as a preposition. The one interesting one is:

"List actual and budgeted unit costs for:
product 1 for 63 to 73."

A rule that said [forPP]to -> S would be wrong here. TIME contained 20 occurrences of "for", all were disambiguated correctly. In TIME2, there were 33 occurrences of "for". Several of these had the pattern "for" ... "to", but 'for' was not a complementiser. This includes the following examples:

- the Spacehab, a self-contained scientific compartment for up to four ...
- placed atop a modified Boeing 747 for a slow return ...
  to Cape Canaveral...
- So far the space agency has been unable to scratch up the money for the opportunity to intercept this visitor from the deep space...
- is a bit too risky for most corporate chiefs to contemplate.

Like the 'have' problem, the issue we have discussed rarely arises in free text. There were more occurrences of the pattern [for][np][to] where it was not to be an embedded sentence, than where it was supposed to be.

14.4 WHICH

The sources were checked for conflicts between "which" as a determiner and as a relative pronoun. In the Scotsman, there were two occurrences of "which" both of which were used to start relative clauses and hence disambiguated correctly. All occurrences of "which" in the M.ECHO sentences were as determiners. In the ASHOK corpus, there were 11 occurrences of "which". Eight of these were sentence initial. There were no errors among all of these, including:

"At plant 2, which product accounted for the lowest percentage of total sales in dollars?" and

"In 1972, which product or products has largest variances?"

In TIME and TIME2 there were no violations.
14.5 WHAT

The sources were checked for "what" as a determiner versus a relative pronoun. There were no occurrences of "what" in the Scotsman. In the MECHO sentences there were no violations. All occurrences were of the form: what force... or what is ... All occurrences in the ASHOK corpus were correct, as were all occurrences in TIME and TIME2.

14.6 THAT

The word 'that' can be used as four parts of speech, determiner, complementiser, pronoun and relative pronoun. All the sources were checked for mistakes in handling this. In the ASHOK corpus there were surprisingly only 3 occurrences. None of these lead to an error. In the MECHO sentences, there were 16 occurrences of 'that'. It was used as a complementiser with 'so that' 9 times and as a complementiser 5 other times. The most difficult sentence:

Find the time that should elapse between the 
release of the two strings if they are to hit 
the ground at the same time.

In TIME, there were 28 occurrences of 'that'. Three of these were as determiners and 1 as a pronoun. These were handled correctly. There were 13 uses as a complementiser and 11 uses as a relative pronoun. In all the relative pronoun cases, the subject was missing from the relative clause.

In TIME2 there were 33 occurrences. There was only one sentence initial example, and it was used as a pronoun and correctly handled by the rules. The word 'that' was used as a determiner 4 times, and as a pronoun 4 times. These uses were resolved correctly. It was used as a complementiser 10 times. None of these uses were after a headnoun, so no errors resulted. Finally, 'that' was used as a relative pronoun 15 times. All these uses came after a headnoun.

It is very significant that each relative clause had the subject missing. There were no occurrences with the object of the relative clause missing. This means that the hard problem of telling a complementiser from a relative pronoun did not happen at all!

14.7 Noun/Verb Ambiguity

To test the coverage of the rules presented in this thesis for disambiguating words which can be both a noun and a verb, I checked [TIME 1978]. In the lead article, there was 315 sentences, containing 213 examples of this ambiguity. There was only one error on these examples: 'Kids his friend'. The use of noun/verb words as verbs was almost always determined by the morphology. The next most common use of noun/verb words was as nouns, and many of these were done by word order. This leaves only a small number of cases where the word is a verb with no morphological changes, but these were also handled correctly.
The following is the grammar for ROBIE. The details of PROLOG have been left out and instead, for each rule, the pattern which must be present before the rule can run and the state of the parser after the rule has run will be given. The reader may find it useful to compare the rules presented here with those used in Appendix A. This appendix reflects the grammar as of May 1981.

A few notes on the notation used in this appendix. In the rule patterns, a '&' means "and", a '#' means "or". [X] means a buffer containing X. [X-Y] means Y is attached to X. This corresponds to the results of the command "attach Y to X." [[X(a,b)]] means a buffer with the features a and b. This corresponds to the commands "add the features "a" and "b" to X. "next" means the buffers are shifted to the next word. Remember the buffers are always the next two items after the Active Node Stack.

We have seen the rule DETERMINER in Chapter 8. Before we see the grammar as a whole, the correspondence between the rule and the diagrams used in this appendix will be explained.

Rule DETERMINER in packet PARSE-DET:
To analyse a det, if you have the feature "det"
in the first buffer then:-
1) attach the first Buffer to The Bottom of the
   Active Node Stack as a determiner;
2) tell the semantic processor you have a determiner;
3) deactivate the packet containing this rule, PARSE-DET
4) activate the packet PARSE-QP-2
   Recursively call the rule matcher.

This rule corresponds to the following diagram:

Rule: determinant    Priority: 10
Active Node Stack:   Active Node Stack:
[NP] {PARSE_DET,X} = > [NP-det] {PARSE_QP_2,X}
Buffers:             Buffers:
[det] next

The first buffer contains the feature "det", indicated by the symbol: [det]. C, the bottom of the Active Node Stack, is an NP node. The result of the attach function is indicated by the symbols: [NP-det], meaning the determiner has been attached to the NP. The packet PARSE_DET has been replaced with the packet PARSE_QP_2. The call to the semantic processor will not be illustrated in these diagrams but happens in every rule.

PACKET: SS_START

This packet is active at the very start of the sentence and deactivated once the subject has been located. This packet finds sentence initial modifiers and determines the type of the sentence. This packet assumes that C (The Current Active Node) is a S.
BEFORE:

Rule: `iLwbat`  
Priority: 5

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S1-binder] {SS_START,X}  
[S]  {CPOOL,SS_START}  
```

 Buffers:  
```
[binder]  
```

Active N  

Buffers:  
```
next  
```

This rule is used to parse sentences of the form: If the mass is 3 lbs, what is the acceleration?

Rule: `wb__quest`  
Priority: 10

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S(major,wh__quest) {PARSE_SUBJ}  
```

 Buffers:  
```
[wh &(np#pp# ap)]  
```

Buffers:  
```
next  
```

This rule handles wh questions and may insert a trace if B2 contains a verb.

Rule: `majoc_daLs`  
Priority: 10

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S(ded,major) {PARSE_SUBJ}  
```

 Buffers:  
```
[np][verb]  
```

Rule: `adverb`  
Priority: 10

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S-adverb] {SS_START,X}  
```

 Buffers:  
```
[adverb][ngstart]  
```

Rule: `aux..invcrt.`  
Priority: 10

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S(yinqufSt,major)] {PARSE_SUBJ}  
```

 Buffers:  
```
[auxverb][ngstart]  
```

Buffers:  
```
[ngstart]next  
```

This rule replaces the old and incorrect yes-no-question rule. The auxverb is pushed onto the movement stack and will be placed at the start of the auxiliary. The movement stack is explained in Section 8.5.3.

Rule: `np..pp_.default`  
Priority: 10

Active Node Stack:  
```
[S]  {SS_START,X}  =>  [S] {SS_START,X}  
```

 Buffers:  
```
[np][pp]  
```

Buffers:  
```
[np-pp]next  
```

257
This rule handles clause initial PPs by attaching them to an initial NP.

**Rule: np utterance**  Priority: 10
Active Node Stack:  
[S]  {SS_START,X}  =>  [S(utterance,major)-np-finalpunc]
Buffers:  
[np][finalpunc]

This rule ensures that wh question words are dominated by a NP node.

**Rule: imperative**  Priority: 10
Active Node Stack:  
[S]  {SS_START,X}  =>  [S(imperative,major)]
Buffers:  
[tenseless]

**Rule: fronted_pp**  Priority: 10
Active Node Stack:  
[S]  {SS_START,X}  =>  [S-pp][SS_START,X]
Buffers:  
[pp]  next

**Rule: wh_np**  Priority: 10
Active Node Stack:  
[S]  {SS_START,X}  =>  [S][SS_START,X]
Buffers:  
[wh]

This packet is active whenever C is an S node. This packet contains the rules to start constituents such as NPs and PPs and handles clause level modifiers.

**Packet: CPOOL**

**Rule: X_and_X**  Priority: 5
Active Node Stack:  
[S]  {CPOOL,X}  =>  [S][CPOOL,X]
Buffers:  
[X][conj][X]

---

258
This rule builds conjoined items into a complex node. The pattern also contains a check to ensure that the Xs are syntactically and sometimes semantically the same.

Rule: poss_det  
Priority: 5  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[pos-poss-np], agree(det)  
[det-poss-np]

This rule degrades possessive NPs to determiners.

Rule: so-that  
Priority: 10  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[comp-adv][that]  
[comp-adv-that][.]

This rule is to handle phrases such as "so that" and "such that". The comma is picked up by a rule is SSL_FINAL.

Rule: propname  
Priority: 10  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[name, not(np)]

Rule: propnoun  
Priority: 10  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[prop-noun]

Rule: pp  
Priority: 10  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[prep][ngstart]

Rule: marked_startnp  
Priority: 10  
Active Node Stack:  
[S] {CPOOL,X}  
Buffers:  
[parse_det,N_POOL]
Rule: startnp    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S] {CPOOL,X}
[NP] {PARSE_QP_1,NPOOL}
Buffers:  
[ngstart\&not(pronoun/\& det)] [ngstart]

This rule starts NP which do not have a determiner. This pattern is needed for historical reasons.

Rule: comparative    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S] {CPOOL,X}
Buffers:  
[than_comp][np] [than_comp-np]
This rule handles phrases such as "3 ft longer than the rod."

Rule: and    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S] {CPOOL,X}
Buffers:  
[conj] next

Rule: comp_to_np    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S] {CPOOL,X}
Buffers:  
[comp_s] [NP-comp_s]

Rule: np_pp    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S-pp] {CPOOL,X}
Buffers:  
[pp] sem chk(pp) next

Rule: pronoun    Priority: 10
Active Node Stack:  
[S] {CPOOL,X} => [S] {CPOOL,X}
Buffers:  
[pronoun] [NP-pronoun]
Rule: vp_attach  Priority: 10
Active Node Stack:
[S] \{CPOOL,X\} => [S-vp] \{CPOOL,X\}
Buffers:Buffers:
[vp]
next

Rule: poss_np  Priority: 15
Active Node Stack:
[S] \{CPOOL,X\} => [S] \{CPOOL,X\}
Buffers:Buffers:
[[t]-possessive] [[t]-possessive]

PACKET: NPOOL

This packet is active whenever C is a NP and contains rules to handle number phrases and other pre-nominal modifiers.

BEFORE:

AFTER:

Rule: qp_and_quant  Priority: 5
Active Node Stack:
[NP] \{NPOOL,X\} => [NP] \{NPOOL,X\}
[QP-qp -conj]
Buffers:Buffers:
[qp][conj][quant] [quant]

This is a very old rule and handles phrases such as '3 lbs and 5 lbs' in a not very elegant way.

Rule: longer_than  Priority: 10
Active Node Stack:
[NP] \{NPOOL,X\} => [NP] \{NPOOL,X\}
[than_name]
Buffers:Buffers:
[than][name] next

For phrases such as '3 years older than Tom'.

Rule: 3_ft/sec  Priority: 10
Active Node Stack:
[NP] \{NPOOL,X\} => [NP] \{NPOOL,X\}
Buffers:Buffers:
[qp][units] [qp-units]
Rule: fl-long Priority: 10
Active Node Stack:
\[ [\text{NP}] \, \{\text{NPOOL}X \} \rightarrow [\text{NP}] \, \{\text{NPOOL}X \} \]
Buffers:
\[ [\text{qp}][\text{adj}] \]

Rule: noun_qp Priority: 10
Active Node Stack:
\[ [\text{NP}] \, \{\text{NPOOL}X \} \rightarrow [\text{NP}] \, \{\text{NPOOL}X \} \]
Buffers:
\[ [\text{qpquant}] \]

Rule: ap_attach Priority: 10
Active Node Stack:
\[ [\text{NP}] \, \{\text{NPOOL}X \} \rightarrow [\text{NP} ap] \, \{\text{NPOOL}X \} \]
Buffers:
\[ [\text{ap}] \]

Rule: qp_attach Priority: 10
Active Node Stack:
\[ [\text{NP}] \, \{\text{NPOOL}X \} \rightarrow [\text{NP} qp] \, \{\text{NPOOL}X \} \]
Buffers:
\[ [\text{qp}] \]

PACKET: PARSE_DET

This packet is active at the start of a NP when a determiner is present. (see rule "marked_start_np in CPOOL").

BEFORE: \quad AFTER:

Rule: determiner Priority: 10
Active Node Stack:
\[ [\text{NP}] \, \{\text{PARSEDET}X \} \rightarrow [\text{NP-det}] \{\text{PARSEQP2}X \} \]
Buffers:
\[ [\text{det}] \]

PACKET: PARSEQP1

This packet is active when a NP is started which does not have a determiner. (see rule start_np in CPOOL)

BEFORE: \quad AFTER:
This packet is used when C is a NP and it has a determiner attached.
Rule: det_quant  Priority: 10
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_QP,2,X}  =>  [NP]  {PARSE_QP,2,X}
Buffers: Buffers:
[quant][adj# noun]       [qp-quant][adj# noun]

Rule: ordinal  Priority: 10
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_QP,2,X}  =>  [NP]  {PARSE_ADJ,X}
Buffers: Buffers:
[ord]       [ap-ord]

Rule: det_quant_done  Priority: 15
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_QP,2,X}  =>  [NP]  {PARSE_ADJ,X}
Buffers: Buffers:
[t]       [t]

PACKET: PARSE_ADJ

This packet is active when C is a NP and after the PARSE_QP,X packets, i.e. the adjective is now expected.

BEFORE:         AFTER:

Rule: adj_group  Priority: 10
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_ADJ,X}  =>  [NP-adj]  {PARSE_ADJ,X}
Buffers: Buffers:
[adj][adj# noun# dim]       [adj# noun# dim]

Rule: adj_np  Priority: 10
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_ADJ,X}  =>  [NP(ap)-adj][NP_COMPLETE]
Buffers: Buffers:
[adj]       next
run np_done next.

This handles NPs which are really only an adjective, for example as in: The ball is red.

Rule: adj  Priority: 15
Active Node Stack:       Active Node Stack:
[NP]  {PARSE_ADJ,X}  =>  [NP]  {PARSE_NOUN,X}
Buffers: Buffers:
[t]       [t]
This is active when C is an NP and the head noun is now expected (i.e. after all the adjectives have been found.)

**BEFORE:** ___ **AFTER:** ___

**Rule: complex_noun**  
 Priority: 10  
 Active Node Stack:  
 [NP] {PARSE_NOUN,X} => [NP-noun] {NP_COMPLETE,X}  
 Buffers:  
 [noun,noun] sem_chk(complex_noun)  

This agreement check worries about noun/modal ambiguity as well as semantic considerations.

**Rule: nouns**  
 Priority: 10  
 Active Node Stack:  
 [NP] {PARSE_NOUN,X} => [NP-noun] {NP_COMPLETE,X}  
 Buffers:  
 [noun,noun] sem_chk(nouns)  

This rule handles the case of "building blocks" and the plural garden paths.

**Rule: np_built**  
 Priority: 15  
 Active Node Stack:  
 [NP] {PARSE_NOUN,X} => [NP] {NP_COMPLETE,X}  
 Buffers:  
 [t]  

**PACKET: NP_COMPLETE**  

This packet is active after the NP has a head noun, and is to find various NP modifiers.

**BEFORE:** ___ **AFTER:** ___

**Rule: qp_pp**  
 Priority: 10  
 Active Node Stack:  
 [NP] {NP_COMPLETE,X} => [NP] {NP_COMPLETE,X}  
 Buffers:  
 [qp][prep]  

**PACKET: PARSE_NOUN**  

This packet is active when C is an NP and the head noun is now expected (i.e. after all the adjectives have been found.)

**BEFORE:** ___ **AFTER:** ___

**Rule: complex_noun**  
 Priority: 10  
 Active Node Stack:  
 [NP] {PARSE_NOUN,X} => [NP-noun] {PARSE_NOUN,X}  
 Buffers:  
 [noun][noun] agree(complex_noun)  

This agreement check worries about noun/modal ambiguity as well as semantic considerations.

**Rule: nouns**  
 Priority: 10  
 Active Node Stack:  
 [NP] {PARSE_NOUN,X} => [NP-noun] {NP_COMPLETE,X}  
 Buffers:  
 [noun,noun] sem_chk(nouns)  

This rule handles the case of "building blocks" and the plural garden paths.
This rule handles phrases such as "3 ft above the sea" by building a compound preposition.

Rule: prep_start  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP]  {NP_COMPLETE,X}
[PP]  {PARSE_PP,CPOOL}
Buffers:  Buffers:
[prep]ngstart]  [prep]ngstart]

Rule: reduced_relative  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP]  {NP_COMPLETE,X}
Buffers:  Buffers:
[verb,ing] or  [wh,verb]
[verb,ed] sem_chk(red_re)

This rule is for reduced relative clauses and garden paths. The "ing" case is handles slightly different to match the MECHO world.

Rule: rel_attach  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP-relative] {NP_COMPLETE}
Buffers:  Buffers:
[relative]  next

Attaches relative clauses to NPs.

Rule: relpron_np  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP]  {NP_COMPLETE,X}
Buffers:  Buffers:
[relpron]  [np(relpron_np)-relpron]

Rule: wh_relative_clause  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP]  {NP_COMPLETE,X}
[\(S(\text{sec,relative})\)]
-\(\text{relpron_np}\)
\{CPOOL,\text{PARSE}\_\text{SUBJ}\}
Buffers:  Buffers:
[relpron_np]  next or [trace]
may insert a trace if B2 is a verb.

Rule: np_pp  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  {NP_COMPLETE,X}  =>  [NP-pp]  {NP_COMPLETE,X}
Buffers:  Buffers:
[pp]  sem_chk(pp)  next
Rule: and  Priority: 10  
Active Node Stack:  

[NP]  \{NP\_COMPLETE,X\} = >  [NP]  \{NP\_COMPLETE,X\}  
[conj\(\text{andc}\)]  
\{CPOOL,PARSE\_VP,PARSE\_CONJ\}  

Buffers:  
[conj]  

Rule: comma  Priority: 10  
Active Node Stack:  

[NP]  \{NP\_COMPLETE,X\} = >  [NP]  \{NP\_COMPLETE,X\}  
Buffers:  
[comma]  

run np\_done next  

Rule: insert\_WH  Priority: 15  
Active Node Stack:  

[NP]  \{NP\_COMPLETE,X\} = >  [NP]  \{NP\_COMPLETE,X\}  
Buffers:  
[det\#$\text{ngstart}]  [wh][det\#$\text{ngstart}]  

This rule is to handle phrases such as 'the boy the man knows'. It has not been carefully thought out.  

Rule: of\_pp  Priority: 15  
Active Node Stack:  

[NP]  \{NP\_COMPLETE,X\} = >  [NP]  \{NP\_COMPLETE,X\}  
Buffers:  
[of\[noun\]]  [of\[noun(\text{ngstart})\]]  

Rule: np\_done  Priority: 15  
Active Node Stack:  

[NEXT]  \{X\}  
[NP]  \{NP\_COMPLETE,Y\} = >  [NEXT]  \{X\}  
Buffers:  
[l]  [NP]  

PACKET: PARSE\_PP  

This packet is to parse PPs and is used whenever C is a PP node.  

BEFORE:  

AFTER:  

Rule: attach\_prep  Priority: 10  
Active Node Stack:  

[PP]  \{PARSE\_PP,X\} = >  [PP\_prep]  \{PARSE\_PP,X\}  
Buffers:  
[prep]  

next  

---
Rule: attach_np  Priority: 10
Active Node Stack:

\[ \text{[NEXT]} \{X\} \]
\[ \{\text{PARSE}_{-}\text{PP},Y\} \rightarrow \{X\} \]
Buffers:

\[ \{\text{np}\} \]
\[ \{\text{PP-np}\} \]

Rule: with_which  Priority: 10
Active Node Stack:

\[ \{\text{PARSE}_{-}\text{PP},X\} \rightarrow \{\text{PP-relpron}_{-}\text{np}\} \{\text{PARSE}_{-}\text{PP},X\} \]
Buffers:

\[ \{\text{wh}\} \]
\[ \{\text{np-wh}\} \]

PACKET: PARSE_SUBJ

This packet is used to attach the subject of a sentence. C is an S expecting the subject.

BEFORE:  
AFTER:

Rule: unmarked_order  Priority: 10
Active Node Stack:

\[ \{\text{PARSE}_{-}\text{SUBJ},X\} \rightarrow \{\text{S-np}\} \{\text{PARSE}_{-}\text{AUX},X\} \]
Buffers:

\[ \{\text{np}[/\text{verb}] \text{agree(subj)}\} \]
\[ \{\text{verb}\} \]

Rule: aux_inversion  Priority: 10
Active Node Stack:

\[ \{\text{PARSE}_{-}\text{SUBJ},X\} \rightarrow \{\text{S}\} \{\text{PARSE}_{-}\text{SUBJ},X\} \]
Buffers:

\[ \{\text{auxverb}[\text{np# ngstart}]\} \]
\[ \{\text{np# ngstart}\} \]

The auxverb is moved by techniques not explained in the thesis.

PACKET: BUILD_AUX

This packet builds the auxiliary. It assumes C is the AUX node.

BEFORE:  
AFTER:

Rule: modal  Priority: 10
Active Node Stack:

\[ \{\text{BUILD}_{-}\text{AUX},X\} \rightarrow \{\text{AUX-modal}\} \{\text{BUILD}_{-}\text{AUX},X\} \]
Buffers:

\[ \{\text{modal}[\text{tenseless}]\} \]
\[ \{\text{tenseless}\} \]
Rule: perfective  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-have]  {BUILD_AUX,X}
Buffers:                      Buffers:
[have][en]                  [en]

Rule: passive_aux  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-be]  {BUILD_AUX,X}
Buffers:                      Buffers:
[be][en]                     [en]

Rule: progressive  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-be]  {BUILD_AUX,X}
Buffers:                      Buffers:
[be][ing]                    [ing]

Rule: do_support  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-do]  {BUILD_AUX,X}
Buffers:                      Buffers:
[do][tenseless]               [tenseless]

Rule: be_pred  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-be]  {BUILD_AUX,X}
Buffers:                      Buffers:
[be][prep# adj]               [prep# adj]

Rule: negative  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-neg]  {BUILD_AUX,X}
Buffers:                      Buffers:
[neg]                         next

Rule: aux_adverb  Priority: 10
Active Node Stack:          Active Node Stack:
[AUX]  {BUILD_AUX,X}  = >  [AUX-adverb]  {BUILD_AUX,X}
Buffers:                      Buffers:
[adverb]                      next

Rule: aux_complete  Priority: 15
Active Node Stack:          Active Node Stack:
[NEXT]  {X}        Active Node Stack:
[AUX]  {BUILD_AUX,Y}  = >  [NEXT]  {X}
Buffers:                      Buffers:
[t]  [AUX]
PACKET: PARSE_AUX

This packet is active after the subject has been attached and when the AUX is expected next. Assumes that C is the S node. It both starts and attaches the AUX node.

BEFORE:  
AFTER:

Rule: to_infinitive  Priority: 10
Active Node Stack:  
[S] {PARSE_AUX,X} = > [S] {PARSE_AUX,X}  
Buffers:  
[to][tenseless]  
Buffers:

Rule: start_aux  Priority: 10
Active Node Stack:  
[S] {PARSE_AUX,X} = > [S] {PARSE_AUX,X}  
Buffers:  
[verb]  
Buffers:

Rule: attach_aux  Priority: 10
Active Node Stack:  
[S] {PARSE_AUX,X} = > [S-aux] {X,PARSE_VP}  
Buffers:  
[aux]  
Buffers:  
[next]

PACKET: PARSE_VP

This packet is active after the AUX has been attached and the MAIN verb is expected. C is the S node.

BEFORE:  
AFTER:

Rule: prep  Priority: 10
Active Node Stack:  
[S] {PARSE_VP,X} = > [S-pp(prep)] {SSL_FINAL,X}  
Buffers:  
[pp# ap]  
Buffers:  
[next]

Rule: main_verb  Priority: 10
Active Node Stack:  
[S] {PARSE_VP,X} = > [S] {SSL_FINAL}  
Buffers:  
[verb]  
Buffers:  
[next]
C is the VP, and the verb is passive. As per Marcus’s rule, it inserts a trace into the first buffer.

BEFORE:  

AFTER:

Rule: passive  
Priority: 5  
Active Node Stack:  
[VP] {PASSIVE,X} = > [VP] {X}
Buffers:  
[t]  

PACKET: SS_vp

This is the main packet to parse a VP. This packet collects the various VP modifiers, etc.

BEFORE:  

AFTER:

Rule: particle  
Priority: 5  
Active Node Stack:  
[VP] {SS_vp,X} = > [VP-particle] {SS_vp,X}
Buffers:  
[particle] som_chk(particle) next

Rule: adverb_group  
Priority: 10  
Active Node Stack:  
[VP] {SS_vp,X} = > [VP] {SS_vp,X}
Buffers:  
[adverb][adverb]  

Rule: adverb  
Priority: 10  
Active Node Stack:  
[VP] {SS_vp,X} = > [VP-adverb] {SS_vp,X}
Buffers:  
[adverb] next

Rule: pp_under_vp  
Priority: 10  
Active Node Stack:  
[VP] {SS_vp,X} = > [VP-pp] {SS_vp,X}
Buffers:  
[pp] next
Rule: particle.2   Priority: 15
Active Node Stack:   Active Node Stack: 
[VP] {SS_VP,X} => [VP-particle] {SS_VP,X} 
Buffers:       Buffers: 
{particle}     next

Rule: vp_done   Priority: 15
Active Node Stack:   Active Node Stack: 
[NEXT] {X}  [NEXT] {X} 
[VP] {SS_VP,Y} => [VP] {t} 
Buffers:       Buffers: 
{t}        next

PACKET: OBJECT

Obviously, this packet finds the object. C is the VP, needing an object.

BEFORE:         AFTER:

Rule: object   Priority: 10
Active Node Stack:   Active Node Stack: 
[VP] {OBJECT,X} => [VP-np] {X} 
Buffers:       Buffers: 
{np}        next

PACKET: TWO_OBJ

For verbs with two objects (such as give), this packet find the first object, and then activates the packet for the other object. C is a VP needing two objects.

BEFORE:         AFTER:

Rule: first_object   Priority: 10
Active Node Stack:   Active Node Stack: 
[VP] {TWO_OBJ,X} => [VP-np] {X} 
Buffers:       Buffers: 
{np}        next

PACKET: NO_SUBJ

C is a VP and the verb is 'want'.

BEFORE:         AFTER:
Rule: create_delta_subj  Priority: 10
Active Node Stack:  Active Node Stack:
[VP] {NO_SUBJ,X}  = >  [VP] {X}
Buffers:  Buffers:
[to][tenseless]  [trace][to]

PACKET: THAT_COMP

C is a VP and the verb takes a 'that complement'.

BEFORE:  AFTER:

Rule: that_s_start  Priority: 5
Active Node Stack:  Active Node Stack:
[VP] {THAT_COMP,X}  = >  [VP] {THAT_COMP,X}
[S-(sec,comp_s)
  -np]
{CPOOL,PARSE_AUX}
Buffers:  Buffers:
[np][verb]  [verb]

For the unmarked case.

Rule: that_s  Priority: 10
Active Node Stack:  Active Node Stack:
[VP] {THAT_COMP,X}  = >  [VP] {THAT_COMP,X}
[S-(sec,comp_s)
  -that]
{CPOOL,PARSE_SUBJ}
Buffers:  Buffers:
[that][ingstart]  [ingstart]

PACKET: INF_COMP

C is a VP and the verb takes an infinitive complement.

BEFORE:  AFTER:

Rule: inf_s_start  Priority: 5
Active Node Stack:  Active Node Stack:
[VP] {INF_COMP,X}  = >  [VP] {INF_COMP,X}
[S-(sec,comp_s)
  -np]
{CPOOL,PARSE_AUX}
Buffers:  Buffers:
[np][to][tenseless]  [to][tenseless]

This rule has been re-formulated with two buffers, but retains the three buffers for compatibility with MECHO semantics.
PACKET: TO_LESS_INF_COMP

C is a VP and the verb is "see" or "saw".

BEFORE: 

AFTER:

Rule: unmarked_s  Priority: 10
Active Node Stack:  Active Node Stack:
[VP] {TO_LESS_INF_COMP},X = > [VP] {TO_LESS_INF_COMP,X}
[S(see,comp_s)
- np]

Buffers: Buffers:
[np][tnsless]

PACKET: TO_BE_LESS_INF_COMP

C is a VP and the verb is seem. This changes "you seem happy" to "you seem to be happy".

BEFORE: 

AFTER:

Rule: insert_to_be  Priority: 10
Active Node Stack:  Active Node Stack:
[VP] {TO_BE_LESS_INF_COMP},X = > [VP] {TO_BE_LESS_INF_COMP,X}
Buffers: Buffers:
[en or adj] [to][be]

PACKET: EMBEDDED_S_FINAL

C is an embedded S that has a VP attached to it.

BEFORE: 

AFTER:

Rule: pp_under_s  Priority: 10
Active Node Stack:  Active Node Stack:
[S-] {EMBEDDED_S_FINAL,X} = > [S-pp] {EMBEDDED_S_FINAL,X}
Buffers: Buffers:
[pp] next

Rule: s_done  Priority: 15
Active Node Stack:  Active Node Stack:
[REST] {X}
[S-] {EMBEDDED_S_FINAL,Y} = > [REST] {X}
Buffers: Buffers:
[t] [S-t]
PACKET: BUILD_NAME

C is a NP that will be a name.

BEFORE: ATER:

Rule: name  Priority: 10
Active Node Stack:  Active Node Stack:
[NP]  [BUILD_NAME,X] = > [NP-name]  {BUILD_NAME,X}
Buffers:  Buffers:
[name]  next

Rule: name_done  Priority: 15
Active Node Stack:  Active Node Stack:
[NP]  {BUILD_NAME,X} = > [NP]  {NP_COMPLETE,X}
Buffers:  Buffers:
[t]  [t]
run the rule np_done.

PACKET: PARSE_CONJ

C is the word "and" and a conjunction is being processed.

BEFORE: ATER:

Rule: drop_and  Priority: 5
Active Node Stack:  Active Node Stack:
[REST]  {X}
[and]  {PARSE_CONJ,Y} = > [REST]  {X}
Buffers:  Buffers:
[vp]  [and][vp]

Rule: drop_and  Priority: 15
Active Node Stack:  Active Node Stack:
[REST]  {X}
[and]  {PARSE_CONJ,Y} = > [REST]  {X}
Buffers:  Buffers:
[t]  [and][t]

PACKET: SS_FINAL

C is the major S with everything attached to it.
Rule: pp_under_s  Priority: 10
Active Node Stack:  
[S] {SSL_FINAL,X} => [S-pp] {SSL_FINAL,X}
Buffers: Buffers:
[pp] next

Rule: s_done  Priority: 10
Active Node Stack:  
[S] {SSL_FINAL,X} => [S-finalpunc]
Buffers: Buffers:
[finalpunc] The parse is finished.

Rule: init_s_bar  Priority: 10
Active Node Stack:  
[S] {SSL_FINAL,X} => [S1(major)][CPOOL,PARSE_SUBJ]
Buffers: Buffers:
[sent_subj] [S(comp_s)][sent_subj]

Downgrades sentential subjects. This is for sentences such as: 'What little fish eat is worms'.

Rule: conjoined_s  Priority: 10
Active Node Stack:  
[S] {SSL_FINAL,X} => [S1-S-conj] {SSL_FINAL,X}
[S2] {CPOOL,SS_START}
Buffers: Buffers:
[comma][conj#binder] next

This rule handles conjoined sentences in a simple, but effective way. It is not adequate in general.

Rule: hypo_s  Priority: 10
Active Node Stack:  
[S1] [S1-S2]
[S2] {SSL_FINAL,X} => [S3] {CPOOL,SS_START}
Buffers: Buffers:
[comma] next
For sentences of the form: If sentence, wh question?
BIBLIOGRAPHY

ing, vol 1, Prentice-Hall, Englewood Cliffs, N. J.

Akmajian, A. and F. Heny [1975] An Introduction to the Principles of Transfor-

Psychology, University of Warwick.

Bach, E. and Hord, G. [1976] "Remarks on 'Conditions on Transformations'", in
Linguistic Inquiry, 7, p. 256-299.

Bach, E. [1977] 'Comments on a paper by Chomsky', in [P. Culicover, T. Wason,
A. Akmajian 1977].

Berwick, R. [1979] 'Learning Structural Descriptions of Grammar Rules From
Examples', IJCAI-79 Conference Proceedings, also MIT
AI-TR 578, 1980.

Berwick, R. [1979] 'Computational Analogues of Constraints on Grammars',
Proceedings of the 18th Annual Meeting of the Association
for Computational Linguistics.

Berwick, R. [1981] 'A Model of Syntactic Acquisition' in the University of Mas-

Bever, T. [1970] 'The Cognitive Basis for Linguistic Structures' in Cognition and
the Development of Language, Hayes, J. (ed), John Wiley

and Ambiguous Sentences', in Memory and Cognition, vol.
1 no. 3.

Report no. 11 (Thesis), Computer Laboratory, University of
Cambridge.


Bole, L. [1978b] Natural Language Communication with Computers, Springer-
Verlag, Berlin.

Bostock, L. and Chandler, S. [1975] Applied Mechanics, Stanley Thomes (Pub-
lisher), Ltd., Bath.

Bresnan, J. [1973] 'Syntax of the Comparative Clause Construction in English',

277


Gersham, A. [1979] 'Knowledge Based Parsing', Research Report no. 156, Yale University, Department of Computer Science.


Halle, Bresnan, and Miller (eds) [1978] Linguistic Structure and Psychological Reality, MIT Press.


Marcus, M. [1975] 'Diagnosis as a Notion of Grammar', in Theoretical Issues in Natural Language Processing, R. Schank and B. Nash-


Martin, W., Church, K., Patil, R. [1961] 'Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results', MIT AI Lab, presented at 'Modeling Human Parsing Strategies Symposium', Austin, Texas.


McKay, K. [1981], 'Parsing Prepositional Phrases', A12 Project, Department of Artificial Intelligence, Edinburgh, University.


Pereira, F. [1980b] "Extraposition Grammars", Department of Artificial Intelligence, Edinburgh, University, Research Paper no. 146,
Presented at the 'Logic Programming Workshop', Debrecen, Hungary.


TIME [9 Jan. 1978], "Good ole Burt; Cool-eyed Clint."

TIME [16 February, 1981], "Embattled but Unbowed."

TIME [27 April 1981], "Touchdown Columbia!".


Winograd, T. [1972] Understanding Natural Language, Edinburgh University


Abbreviations Used:
IJCAI - Conference Proceedings of the International Joint Conference on Artificial Intelligence.
AISB - Conference Proceedings of the Association for Artificial Intelligence and Simulation of Behavior.
MIT - The Massachusetts Institute of Technology, Cambridge, Mass.