This is to certify that the

dissertation entitled

On Interleaving Syntax and Semantics in Parsing

presented by

Dongyul Ra

has been accepted towards fulfillment
of the requirements for
PhD Computer Science
degree in

George C. Stockman
Major professor

17 Oct 1989
Date

ON INTERLEAVING SYNTAX AND SEMANTICS IN PARSING

By

Dongyul Ra

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1989

ABSTRACT

ON INTERLEAVING SYNTAX AND SEMANTICS IN PARSING

By

Dongyul Ra

It is agreed that natural language parsers should use both syntactic and semantic knowledge for better analysis. Several approaches for using these distinct knowledge have been proposed: non-integrated parsing and integrated parsing. Under non-integrated parsing, there are two kinds of approaches: the old extreme one which is represented by the serial stage parsing and the recent one called interleaved semantic processing. Under integrated parsing, there are also two views: the old extreme one represented by conceptual analyzers and the recent moderate one shown in Lytinen's MOPTRANS parser.

This thesis attempts to update the interleaved semantic processing to achieve the power of integrated parsing. An integrated parser, called SYNSEM, has been developed for this purpose. In SYNSEM, each alternative syntactic analysis becomes a branch of parsing. Multiple branches can run in parallel but they are filtered, as soon as possible, according to their semantic processing results at the periodic synchronization points. SYNSEM's approach is able to utilize semantics(as well as syntax) as much as the recent integrated parsing approach.

To provide the necessary semantics for comparing the branches of parsing, a knowledge base(KB) is built using the recent knowledge representation language, KODIAK. A marker passing paradigm is used to utilize the knowledge base. The semantic processing result for each branch is realized as the paths in the knowledge base. To

solve the hard problem of finding the best paths between two concepts, a mechanism called *secondary marker passing* has been developed which efficiently cuts down the number of paths to be considered. Several heuristics have been developed for selecting only the best paths. The comparison of branches to filter those with inferior semantic results is done by comparing the KB paths. A heuristic for this comparison is also developed.

The approach described so far aims at structural disambiguation. Word sense ambiguity is another big source of ambiguities in natural language. Several methods for word sense disambiguation are developed in SYNSEM. By manipulating a set of the KB paths, an easily conceived and simply implementable method for word sense disambiguation is introduced. The path adjustment operation after the concretion of a sense to the more specific sense is a new source of information utilized in the SYNSEM parser.

To my wife and daughter

for their love and support

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. George C. Stockman, for his guidance over the years at Michigan State University. Dr. Stockman advised me at every phase of my research. He showed a great deal of patience while I was going through the long period of reading literature and preparing for the research. He patiently listened to my ideas and plans and provided valuable comments and suggestions even when my ideas were sometimes vague or seemed strange. He also provided encouragement for me when I needed it to continue this arduous thesis work. Without that, my program might never have been completed.

I am grateful to my committee members, Dr. Anil K. Jain, Dr. Carl V. Page, Dr. Jon Sticklen, Dr. J. Kathryn Bock, and Dr. Barbara K. Abbott for giving me kindness, valuable suggestions and comments and for teaching me during my graduate program.

I am grateful to Barbara Czerny for reading the draft of this thesis and correcting numerous errors. I also would like to thank my friends I've made during my stay in this university for helping me in one or more ways: Chaur-Chin Chen, Sei-Wang Chen, Xian He Sun, Jayashree Ramanathan, Arun Nanda, Rob Jaksa, Mahmoud Pegah, John Kern, Joe Miller, Paul Wolberg and many others.

Finally I would like to thank my wife and my daughter for their constant support, patience, and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

For humans natural language is an easy, flexible, and powerful means of communication. The goal of natural language processing, an area of Artificial Intelligence, is to allow humans to use natural language to communicate with computers. One of the most important advantages of achieving this goal is that humans can use their language, i.e., natural language, to communicate with computers. They can issue commands to the computer with, for example, English and receive responses in English. Computers could read or write documents written in natural language.

The global goal of natural language parsing is to make computers understand input expressed in natural language. But this definition needs more elaboration because "understand" is not specific enough. The fact that "a man understood a sentence" means that he got the meaning of the sentence. This means: first he translated the input sentence into a chunk of some biological medium that is used by the brain processes for doing things such as thinking, storing, reasoning, speaking, etc.; second, he identified the status of the chunk by relating it to the existing knowledge (for example, it is true, false, new, etc.). An interesting thing is that these two steps of understanding are not separate steps. It seems that they are interwined and mixed. A similar analogy can be applied to the case of computers. The fact that a computer understands an input sentence means that the input sentence should be translated into some form of a representation language that can represent the meaning of the sentence. This representation language is usually called the internal representation language or knowledge representation language. This internal representation language is the universal medium that is used by the intelligent processes in the system. This might be used for further processing such as reasoning or

answering.

Building a natural language parser is a very complex and difficult task. The detailed processing principles and steps have been studied much but there is no unified theory yet. One of the difficulties of building a natural language parser is because natural language is full of ambiguities. To resolve the ambiguities, various kinds of knowledge should be extensively used. The use of syntactic knowledge usually enables the parser to understand the grammatical structure of a sentence. Syntactic analysis is one of the areas that is most clearly understood according to the research of linguists. But as we go from syntactic knowledge to the higher level knowledge, we do not have clear understanding of the exact way of representing the knowledge or using the knowledge during parsing. A big problem arises from the fact that many syntactic ambiguities require the use of higher level knowledge (let's call this semantic information broadly) for resolution.

In addition to this problem, there is not much agreement on the role that the syntactic and the semantic knowledge should play in parsing and there is not much agreement on what kinds of structures should be built during the course of parsing. Some argue that syntactic knowledge need not be extensively used and syntactic structure of the input sentence need not be completely built and used because the final goal of parsing is to compute its meaning (semantic interpretation) instead of its syntactic structure(Schank and Birnbaum, 1980). Some argued that the syntactic structure of the sentence should be fully computed first and then semantic information should be applied to resolve the ambiguities and compute the semantic interpretation(Woods, 1972). Others argued that the parsing should involve the use of syntactic and semantic information at the same time(Charniak, 1981c). The means of how to use semantic information is not completely known.

The research to be reported in this thesis investigates the use of semantics in parsing. What should the parsers' control strategy be for the efficient use of syntactic and

semantic information? What kind of method should be used to glean the semantic information that is necessary for the disambiguation? How are they applied to influence the computation of the syntactic structure? In other words, this thesis is dedicated to finding a parsing model that has a better answer for these questions. A parsing model that is new and advanced will be adopted and an actual parser based on this model will be developed. The problems and techniques that arise during the development of the parser will be examined in this thesis.

In this chapter, several parsing approaches that have been used in previous parsers will be introduced and then a new parsing approach will be introduced and motivated. This chapter will conclude with the outline of the thesis.

1.1 Serial Stage Parsing

In the serial stage parsing approach, the syntactic analysis and semantic analysis are done separately in two stages which are ordered serially. As shown in Figure 1.1, the syntactic analysis stage comes

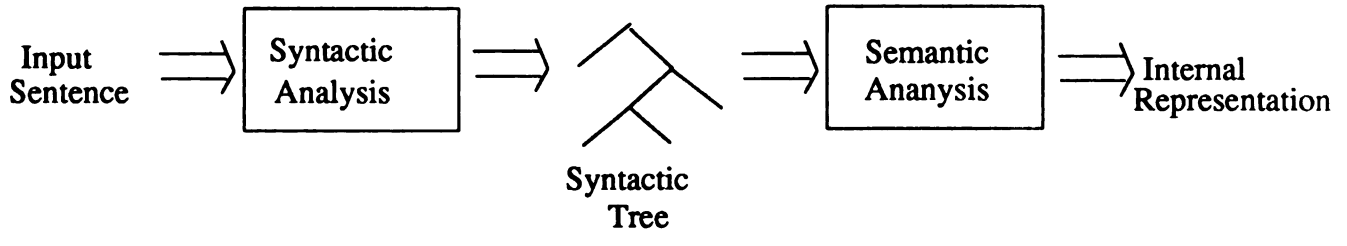


Figure 1.1 Stages of Parsing

first and then the semantic analysis stage follows. In the syntactic analysis stage, grammatical knowledge is used to build the syntactic tree for the input sentence. After the full syntactic tree for the whole input sentence has been constructed, the semantic analysis is done in the next stage. The syntactic tree from the first stage is input to the semantic analysis stage. Non-syntactic knowledge is the major information used in the semantic analysis stage. One of the advocates of this parsing approach was Woods(1972). He argued that the parsing will be more efficient and faster if syntactic analysis is done first

without worrying about the semantic content of the sentence and then semantic processing (which requires lots of computation) is based on the result of the syntactic analysis which is a more appropriate form to process. The following quotation from Woods(1972:145) shows his argument:

> We have not gotten any conclusive answers yet regarding the effectiveness of this type of semantic screening, but it looks as if it takes longer to do the parsing and semantic interpretation overall if the interpretation is done during the parsing than it does if the parsing is done first and the interpretation afterwords. This seems to be because the semantic interpretation process is at least as expensive as following out the syntactic consequences without semantic guidance, and the syntax is as likely to rule out an alternative and keep the interpreter from having to interpret it as the semantics is likely to rule out an alternative and keep the parser from having to operate further on it. Clearly there is no *a priori* reason to expect that semantic screening will save more of the parser's effort than syntactic screening saves of the interpreter's effort.

But this approach has been opposed by many researchers. For example, Charniak(1981c) refuted Woods by saying that the serial stage approach has no psychological reality and it is rather inefficient in the face of lots of ambiguities in the human language. The approach of serial stage parsing became an out-of-date parsing approach which is the typical example of non-integrated parsing. The non-integrated parsing approach is directly opposite to the integrated parsing approach that will be explained in the next section.

1.2 Integrated Parsing

In the integrated parsing approach, there is no separation between the syntactic and semantic processing stages as shown in Figure 1.2.

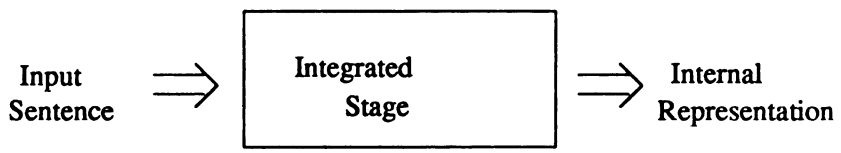


Figure 1.2 Integrated Stage of Parsing

In addition to the fact that two stages are integrated into one stage, this approach has the

following features: semantics is the major driving force of parsing; syntactic trees are not built, which results in using only local syntactic cues and not global syntactic information. Semantic representation of the input sentence is directly built. In this model, syntax is viewed as less important information for parsing. This view has been strongly advocated in Schank and Birnbaum(1980). A series of parsers have been developed following this approach: the Margie parser in Riesbeck(1975), ELI in Riesbeck and Schank(1976), IPP in Lebowitz(1980), and Conceptual Analyzer in Birnbaum and Selfridge(1981). Another parser that belongs to this model is the semantic grammar shown in Burton(1976).

But one recent parser attempts to provide more syntactic analysis to integrated parsing while it still tries to maintain the characteristics of integrated parsing. This parser is the MOPTRANS parser developed by Lytinen(1984). Instead of using the lexical request-based rules used in the conceptual analyzers, MOPTRANS uses *the generalized syntactic rules*. Figure 1.3 shows two generalized syntactic rules, "subject" rule and "object" rule.

```
Subject Rule
  Syntactic pattern:       NP, V (active)
  Additional restriction: NP is not attached syntactically
  Syntactic assignment:    NP is SUBJECT of V, V is a MAIN CLAUSE
  Semantic action:         NP is actor of V(or another slot,
                           if specified by V)
  Result:                  V (changed to S)

Object Rule
  Syntactic pattern:       S, NP
  Additional restriction: NP is not attached syntactically
  Syntactic assignment:    NP is (syntactic) OBJECT of S
  Semantic action:         NP is (semantic) OBJECT of S(or
                           another slot, if specified by S)
  Result:                  S, NP
```

Figure 1.3 Generalized Syntactic Rules in MOPTRANS

The generalized syntactic rules are indexed in a way that semantic knowledge is fully used before the use of syntactic knowledge. At each step of parsing, all possible semantic connections between the semantic objects in the memory are found. Then the

generalized syntactic rule which satisfies the constraint of its syntactic pattern and builds the best semantic connection is selected and executed. Thus semantic knowledge is used first and then syntactic knowledge is consulted at each step of rule indexing and execution. This strategy guarantees the utilization of the predictive power of semantics. The important point in the rule indexing strategy of MOPTRANS is that the parser will never consider a rule that results in incorrect semantic effect. This can be shown using the following two sentences:

(1-1)The man wrapped the present.

(1-2)The present wrapped by the man was expensive.

After "wrapped" is input in the first sentence, the memory contains two elements, HUMAN for "the man" and COVER for "wrapped". The possible semantic connections between these two memory elements are sought by the parser and it is found out that there is only one possible connection: HUMAN is connected to COVER via the agent relationship. Then any rule that matches the syntactic pattern and builds this semantic connection is sought. In this case, only one rule, the Subject rule, is found. Then this rule is executed and the parsing continues. Now, let's consider the situation after reading "wrapped" in sentence (1-2). The memory contains two elements: GIFT and COVER. Only one semantic connection is found between these two elements: GIFT is the (semantic) OBJECT of COVER. All syntactic rules which match the syntactic pattern and can build this semantic connection are found. Note that the Subject rule matches the syntactic pattern of the current situation but it can not build the required semantic connection. Thus the Subject rule is not found. The Object rule is not found because its syntactic pattern does not match the input pattern. But the Unmarked-Passive rule is found by the rule matching process.

```
Unmarked-Passive Rule
  Syntactic pattern:       NP, VPP
  Additional restriction: none
  Syntactic assignment:    NP is (syntactic) SUBJECT of VPP, VPP
                           is passive, VPP is a RELATIVE CLAUSE of NP
  Semantic action:         NP is (semantic) OBJECT of S(or another slot
                           if specified by VPP)
  Result:                  NP, VPP(changed to S)
```

This rule's syntactic pattern matches the pattern of the input. According to the semantic action of this rule, the connection of semantic OBJECT can be built by this rule. Therefore this rule is found and executed. The approach of MOPTRANS greatly improves on the weakness of the conceptual analyzers in the analysis of sentences that have complex syntactic structure. MOPTRANS used one stage of parsing but both syntactic and semantic knowledge is used in a more balanced way than the conceptual analyzers.

1.3 Interleaved Semantic Processing

It has been agreed that the serial stage parsing model is not the appropriate model for natural language parsing. An updated version of the serial approach has been popular. This is called interleaved semantic processing(Ritchie, 1983). In this approach the syntactic analysis is still the major driving force of parsing but semantic knowledge is consulted during syntactic analysis. Semantic processing is interleaved with the syntactic decision making points so that the decision can be made after consulting semantic information. Semantic interpretation is built in parallel with the construction of the syntactic representation. The possibility of doing the wrong syntactic analysis (which does not make sense semantically) for the whole sentence in the serial stage parsing model can be removed in the interleaved semantic processing model while the parser can still take advantage of the clear-cut information that can be provided by the syntactic structure of the sentence.

One of the typical parsers incorporating interleaved semantics is Psi-Klone(Bobrow and Webber, 1980). Psi-Klone's syntactic analyzer is based on the ATN parser. When it executes actions of an arc of a state of ATN, messages are passed to the semantic

component. The semantic component builds the semantic structure corresponding to the messages. If the attempt at building the semantic structure succeeds, the semantic component returns the signal of success to the ATN analyzer (otherwise, a signal of failure is returned). Let's consider Figure 1.4 which shows states of the ATN.

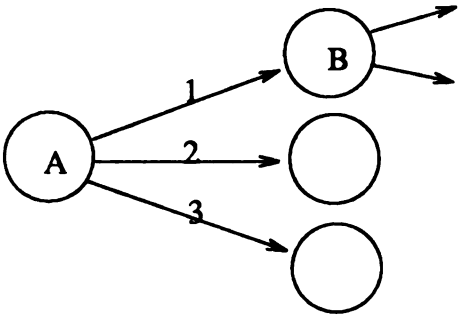


Figure 1.4 Arcs of an ATN State

At state A, arc 1 is tried first. If the semantic component returns the signal of success for the messages of this arc, the parser moves to state B without considering the other arcs(arc 2 and arc 3). But if the failure signal is returned, the parser tries the next arc, arc 2. This arrangement of control takes advantage of separating syntax and semantics but allows a large amount of semantic checking. This is a modular approach and the syntactic rules are independent of the domain.

The messages passed to the semantic component are of the form (M, L, C), where M(matrix) is some half-built constituent, C is a constituent of some kind and L is a label that indicates some syntactic relation. What (M, L, C) suggests is that C is attached to M by a link labeled with L. The semantic component checks the semantic constraint related to this attachment to decide if the attachment can be done. When the attachment is done, a symbolic name that points to the result of this structure building operation is returned to the parser. Upon the receipt of a message (M, L, C), the semantic component uses relation-mapping rules(RM rules) to determine into which semantic relation L can be mapped. The semantic relation here is something like cases or roles in other systems. The attachment is done based on this semantic relation. For each semantic

relation, there are interpretation rules(I rules) which are applied to convert the case-frame-like structure into a full semantic item for the updated matrix.

Note in Figure 1.4 that arcs are tried one after another, serially, until one succeeds. The first arc that gets the signal of success is the one that is selected by the parser to update the parsing. But the problem is that an arc ordered later may have better semantics. This arc with best semantics is missed without being detected because the arc appearing before it has just enough reasonable semantic effect to pass the semantic check. For example, the semantic effect related to arc 3 may be better than that of arc 1 but the arc 1 is reasonable enough to pass the semantic check. In this case, the parser never notice the fact that arc 3 is better than arc 1 and it just chooses arc 1 to update the parsing and move to the next state.

Another parser that is an example of the interleaved semantic processing approach is Paragram and Absity system(Charniak,1983b; Hirst, 1984, 1986). Paragram is based on the Marcus-style parser. At each stage of parsing, the state of stack and buffer is matched against the rules in the rule base (actually the rules in a packet of the rule base). A rule that is found is executed. During the execution of the rule, the corresponding semantic processing is done in parallel. The parser attaches one element to another element only when the semantic effect related to the attachement is proved to be good. Otherwise, another attachement in the rule will be tried. Because the parser is a deterministic parser, only one rule should be found by the pattern matcher. It is not clear what will happen when all attachment attempts in the rule fail because of semantic reasons. This problem is beyond the scope of Paragram. Thus the selection of one out of several possible attachments in the same rule is based on the semantic checking and this is the way of achieving "semantic interleaving" used in the Paragram and Absity parsing system.

Another early parsing system that used interleaved semantic processing is the SHRDLU parser developed by Winograd(1972). When this parser builds a semantic structure, a list of semantic markers is associated with the structure. The parser checks

the internal consistency of the semantic marker list when it builds a new structure and the checking is used as the feedback to the parser. This semantic feedback enables the parser to avoid the unnecessary construction of structures. The parser is based on procedures instead of declarative rules. But it is clear that the syntactic analysis routine consults the semantic component before it decides to take any action of syntactic analysis. At an ambiguous point, the first analysis with the reasonable semantic feedback is selected. The advocates of integrated parsing argued that the interleaved semantic processing approach also belongs to non-integrated parsing where the serial stage parsers are the most extreme examples. Lytinen(1984:4) described non-integrated parsing as follows:

> Syntax plays an important role in the process of understanding the meaning of a natural language text. Syntactic and semantic processing are largely separate, with syntactic processing performed first (although semantic processing can be interleaved with syntactic processing; i.e. once syntax has produced a partial analysis, semantic interpretation of that portion of the text can proceed before other portions of the text are syntactically analyzed). Syntax and semantics interact with each other in limited ways, if at all. Syntax might be allowed to ask certain types of questions of semantics at particular times, but communication between syntactic and semantic processing is not unlimited. Knowledge about syntax and semantics is also largely separate. Syntactic knowledge can be expressed without much reference to semantics.

1.4 Improving Interleaved Semantic Processing: a New Proposal

It cannot be denied that the popular approach to parsing is to make syntactic analysis rather than the semantic analysis the major driving force of parsing. The reason is that syntax can provide the first clear information about the structure of a sentence. This structural information is useful to understand the meaning that the sentence conveys. If the meaning can be understood easily without the use of this syntactic information as the integrated parsing advocates argued, why does the syntax exist? The following quotation from Charniak(1981c) argues for the whole idea behind interleaved semantic processing:

> In the scheme of this model(HEARSAY's blackboard model), the syntax and semantic components work in parallel, leaving their conclusions on a common blackboard. The idea is that even if the syntactic component fails, the semantic component will still succeed, at least if the sentence is comprehensible.
> That there are troubles with this model is suggested by the fact that HEARSAY itself did not really use it. While technically the two systems worked in parallel, in actual design the semantics would only work on the output of the syntactic component, making the model isomorphic to the semantic subroutine model (or the interleaved semantic processing model) proposed earlier. Nor is it hard to see why this should be the case. As we have already noted, syntax is the most obvious way to get a first cut at the logical structure of the sentence. In a real blackboard model the semantics would be forced to do without the logical structure established by the syntactic component. This would mean that it would have to re-establish the same information by other means. If it could do this, then why bother with syntax at all? In other words, a blackboard model will ultimately degenerate into either a "no syntax" model, or a combined semantics model.

The advocates of integrated parsing claimed that interleaved semantic processing does not use the full power of semantics and thus should be classified as a non-integrated parsing approach. The reason for this claim is that semantic processing is done only for the analysis that is suggested by the syntactic analyzer. For example, in Figure 1.4, the ATN parser at first suggests the semantic processing for arc 1 even if the other two arcs are possible ways of parsing in state 1. If arc 1 results in the semantic processing that is reasonably good, the possibilities of arc 2 and arc 3 are never considered. But it may be the case that the analysis represented by arc 3 may be the better analysis based on semantic processing. (Note that the order doesn't seem to be an important point here, just the severe control of semantics by syntax.) This means that syntactic analyzer may prevent the parser from taking the better analysis. This can be avoided in the integrated parsers.

In the case of Parsifal(Marcus, 1980) or Paragram, the problem gets worse than with Psi-Klone. Those parsers have a configuration where it is hard to apply semantic power to the action of the parser. For example, it is assumed that only one rule can be correctly found by looking ahead up to three buffer cells. This rule selection utilizes only syntactic information without any semantic information. Semantic checking can be used only for selecting one attachment action over the other in the action part of the

same rule. Thus the use of semantics for guiding the parser is limited compared to Psi-Klone. There is also an unsettled argument on the number of buffer cells to look ahead. For example, Milne used the lookahead of up to only two buffer cells, arguing that the two buffer lookahead is enough. Milne(1982) showed that a parsing decision should sometimes be made based on the semantic bias instead of using the lookahead of all three buffer cells.

The major goal of the research reported in this thesis is to design a parser that can fully utilize semantics but is still based on interleaved semantic processing. The previous interleaved semantic processing approach will be updated to achieve better use of semantics. Thus the major driving(controlling) force of the parser will be syntactic analysis but the method of utilizing the semantics will be modified. The parsing system that has been developed for the purpose is called SYNSEM. SYNSEM's syntactic analyzer is based on Parsifal. It has been significantly modified in several ways to achieve the aim of the thesis. The semantic processing is based on world knowledge that is written in the knowledge representation language called KODIAK developed by Wilensky(1987). This means that the internal representation language of SYNSEM is KODIAK.

The control of SYNSEM is designed in a way that the full use of semantics can be possible even if the parser is based on syntactic analysis (note that the base of interleaved semantic processing is syntactic analysis). This approach resulted in the partial-parallelism in the parser. The partial-parallelism means that the parser mostly runs in serial mode but it can run in parallel for the period when there exists ambiguity. The parser tries to reduce down the number of parallel branches to one as soon as possible using any information available (either semantics or syntax).

To use the knowledge base efficiently and conveniently, considering that the KODIAK knowledge base is a kind of semantic network, the marker passing paradigm has been adopted as the major means that the semantic component uses to obtain

semantic information. This strategy allows easy use of the knowledge base. Thus the framework of the parser is to use a rule-based syntactic analyzer as the syntactic analysis component and use the marker passing paradigm as the major means of getting semantic information. Some schemes are developed that make this framework achieve the global aim of designing a parser that utilizes semantics as fully as possible.

One of the most important problems related to marker passing is how to find the best paths between two concepts. This problem has been extensively studied in this research. The parser is designed in a way that the goodness of a syntactic analysis can be represented by the goodness of the paths related to the analysis. This creates a problem of comparing the paths that represent the possible analyses that compete. The path comparison problem is also addressed in this research.

The word sense disambiguation problem is one of the problems that any significant natural language understanding system should address because it is another major source of ambiguity. The marker passing-based parsers have some advantages for handling this problem(Charniak,1986). Computational techniques on how to handle this problem under the framework of SYNSEM have also been studied.

1.5 Outline of the Thesis

In Chapter 2, the parsing models in psycholinguistics are reviewed for the purpose of providing some support to the framework of SYNSEM. The partial-parallel approach used by SYNSEM is supported by some recent parsing models. The overall framework of SYNSEM will be designed in Chapter 3. The discussion in that chapter is mostly related to the issue of the control strategy of parsing. The partial-parallelism is adopted as a means of improving interleaved semantic processing to use semantics as fully as possible. Chapter 4 is related to the description of the syntactic component of SYNSEM. The rule-based Marcus-style syntactic analyzer is introduced. Several new features of SYNSEM are explained such as the format of rules, rule-indexing, and the

relation between ambiguity and rules. How the analyzer works is explained using some detailed examples. Chapter 5 is a prelude to the description of the semantic processing component. First KODIAK knowledge representation language is explained in detail but only as much as necessary in this research. Secondly, the marker passing method used in SYNSEM is described in detail. How the markers are passed and how the collisions are detected will be explained. The core problems of the semantic processing component are touched on in Chapter 6. The two problems addressed in that chapter are: how the best path(s) are searched between two concepts in the knowledge base; how the branches that run in parallel are compared based on the semantic processing result and how the best ones are filtered to reduce the ambiguity. In Chapter 7, word sense disambiguation is attacked. The major weapon for this comes from the framework of SYNSEM that uses the semantic network and the marker passing paradigm. The use of syntactic information for this problem is also considered. Chapter 8 concludes the thesis by discussing why SYNSEM can utilize semantics as much as integrated parsers and summarizes the research results and introduces the future research problems.

The brief summary of the accomplishments in this thesis is as follows. A new control strategy for using syntactic and semantic information in parsing has been proposed. The syntactic and semantic analysis components to achieve this control strategy have been developed. For the syntactic analysis component, a rule-based analyzer with new features has been developed. To provide semantics to the parser based on the marker passing paradigm, several heuristics for finding the best path between two concept nodes have been introduced along with the powerful mechanism called secondary marker passing. We introduce a heuristic that is used for comparing the semantic interpretation results of the competing branches of parsing. The word sense disambiguation problem has been studied and partly solved in the context of the parser developed.

CHAPTER 2

PARSING MODELS IN PSYCHOLINGUISTICS

There has been a great deal of research in psycholinguistics for parsing models for the human language processing system. It is worthwhile to review these parsing models and compare them because this can shed some light on the design of natural language processing systems. The human parsing system shows mysterious high performance in spite of the abundant ambiguities in natural language. It does not necessarily mean that the human parsing system has the most efficient processing mechanism. But, no natural language parsing systems that have been developed so far can compete with humans. Thus there might be a good chance that following the strategy of the human parsing system may result in the design of a good automatic natural language parser.

A parsing model can be characterized by considering the following two important questions: (1) whether all possibilities of an ambiguity are considered in serial or in parallel; (2) what kind of information is used to select one alternative out of the multiple analyses of the ambiguity. Some important parsing models will be reviewed with these points in mind.

2.1 Serial Models

A parsing model that belongs to this group considers only one possible analysis at a time when an ambiguity is encountered. After selecting one possibility according to some selection criteria, the parsing continues as if there has been no ambiguity. If this selection proves to be wrong by some disambiguating material appearing later in the sentence, the parser goes back to the ambiguous point and the next possibility is tried. Thus the alternatives of an ambiguity are tried in serial as illustrated in Figure 2.1.

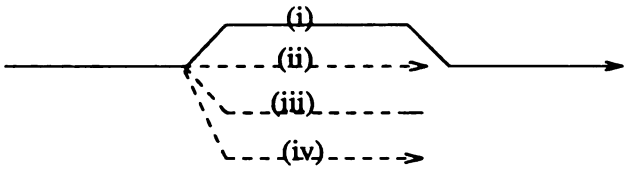


Figure 2.1 Alternative Analyses Tried Serially

At first, alternative (i) is tried. If the parser encounters a dead end later, the next alternative, (ii), is tried. Then if it fails again, the third alternative, (iii), is tried, and so on. When a parser encounters a dead end, the parser should retreat to a previous point of the sentence to reanalyze it. There hasn't been much research on this backtrack and reanalysis strategy. It has been known that there can be three kinds of reanalysis strategy. The first is to start the reanalysis from the first word of the sentence. The second is the blind chronological backtrack, which means that the parser backs up to the most recent ambiguous point and the next alternative is used to start the analysis from that point. The last is the one which is called selective reanalysis. In this strategy, the parser backs up to the most appropriate ambiguous point in the sentence which is determined by considering all information available to the parser. Let's consider some serial models which are classified according to what kind of information is used to determine the order of considering the alternatives of ambiguities.

2.1.1 Structural Preference Model

This model is also known as the local and minimal attachment principle which was advocated by Frazier and Fodor(1978). They proposed that the human parsing model consists of two stages. The first stage is called the Preliminary Phrase Packager(PPP) and the second stage the Sentence Structure Supervisor(SSS). The PPP assigns lexical and phrasal nodes to groups of words within the lexical string that is received. The output of the PPP is shunted to the SSS which combines these structural phrases into a complete phrase marker by adding higher nonterminal nodes. The PPP is a shortsighted device which looks at the input string using a narrow window (of the size of 6 words). Thus

the PPP analyzes the sentence locally and the global processing is the job of the SSS. It is not clear how the PPP and the SSS are synchronized in detail except that the PPP goes ahead of the SSS. From the fact that the PPP is shortsighted, there comes the local attachment principle which can be described as:

> *An attachment can not be made to a structure that is out of sight of the PPP. If there are more than one possible attachment points inside the sight of the PPP, then they are equally fine as the attachment point.*

We can show the example of local attachment in sentence (2-1). In (2-1),

(2-1) John read the postcard, the memo, and the letter to Mary.

(2-2) John read the letter to Mary.

"to Mary" is preferably attached to "the letter" because another possible attachment point "read" is out of sight of the PPP. But in (2-2), both "read" and "the letter" are within the window of the PPP. Thus "read" and "the letter" are equally fine as the attachment point of "to Mary" (but actually "read" is preferred by the minimal attachment principle which will be explained below). Another example of the local attachment principle can be found in (2-3) and (2-4):

(2-3) Ellen got the idea that the coast guard was going to send a life raft across.

(2-4) Though Martha claimed that she would be the first woman president yesterday she announced that she'd rather be an astronaut.

In (2-3), "across" is associated with "send" instead of "got" because "got" is too far away. In (2-4), "yesterday" is associated with "announced" because "claimed" is not available within the window of the PPP when "yesterday" is being processed.

After the local attachment principle is applied, the minimal attachment principle is used if there is more than one possible attachment that is still left. The minimal attachment principle can be described as follows:

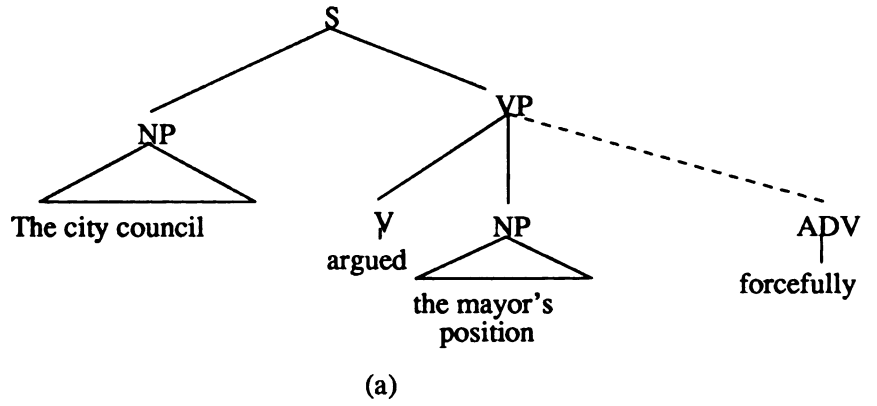
Attach the incoming material into the phrase-marker being constructed using the fewest number of nonterminal nodes.

This principle can be illustrated by looking at the processing of (2-5) and (2-6):

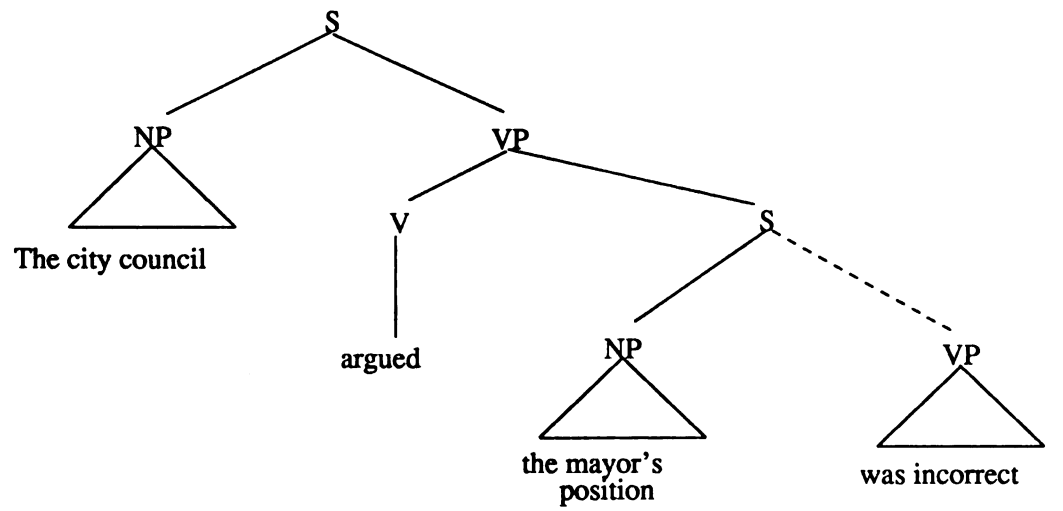
(2-5) The city council argued the mayor's position forcefully.

(2-6) The city council argued the mayor's position was incorrect.

When the noun phrase "the mayor's position" is analyzed, there are two possible attachment points as shown in Figure 2.2. It is clear that



(a)



(b)

Figure 2.2 Minimal Attachment

the attachment shown in (a) of the figure uses a smaller number of nonterminal nodes than that in (b). According to the minimal attachment principle, the attachment in (a) is chosen first by the parser. In the processing of (2-5), this decision will prove to be correct by the input that follows. But "was incorrect" in (2-6) will prove that the decision of the minimal attachment principle is wrong. Then the parser should attempt reanalyzing the sentence by trying the other attachment.

It was said that the local attachment principle provides no preference in (2-2). Then the minimal attachment principle is used to attach "to Mary" to "read" rather than to "the letter", which can be illustrated using Figure 2.3.

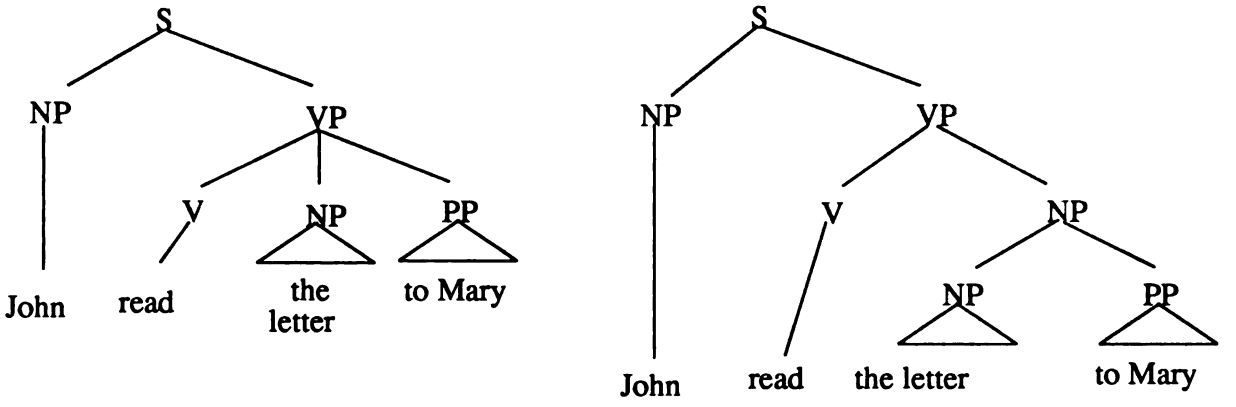


Figure 2.3 Failure of Local Attachment

Thus the local attachment principle is used first and then the minimal attachment principle is used if there is still more than one possible attachment point.

In (Frazier and Raynor,1982), some experimental data was reported which further strengthens their argument. In their experiment, the eye movements were recorded as the subject read a sentence containing the attachment ambiguity. They found that shorter reading time is taken for the sentences whose ambiguity resolution conforms to the strategy of the local and minimal attachment principle than for the sentences whose ambiguity resolution does not conform to this strategy.

They extend the concept of garden-path by incorporating into the group of garden-path sentences the sentences whose temporary ambiguity is resolved into the wrong

analysis and then revised later by the disambiguating material. In other words, the sentences containing temporary ambiguities which require reanalysis but do not require conscious effort are also considered to be garden-path sentences. Other significant data they provided is that the reading time and fixation duration in area (b) in Figure 2.4 (the region where the ambiguity remains) is not longer than (a), and the difference in reading times is associated with the disambiguating region (c).

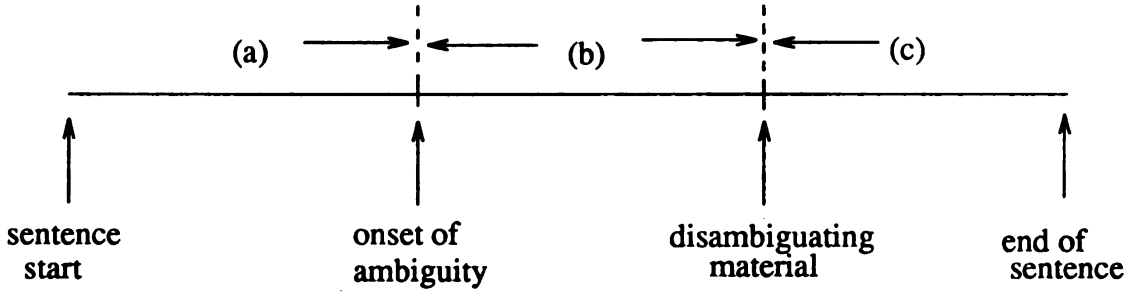


Figure 2.4 Ambiguous Region

This data supports the view that the existence of the ambiguity is not detected if it does not garden-path (even if there is ambiguity) and the view of the serial model in which only one analysis of the ambiguous material is computed at first.

2.1.2 Lexical Preference Model

Ford, Bresnan and Kaplan(1982) (henceforth FBK) introduced a theory which differs from the structural preference model in the previous section in how an ambiguity is resolved. FBK argued that the lexical property of the head of a phrase is what determines the order of choosing the alternatives of an ambiguity. The sentences from (2-7) to (2-9) can be used to illustrate this lexical preference theory :

(2-7a)The woman wanted the dress on that rack.

(2-7b)The woman positioned the dress on that rack.

(2-8a)The tourists objected to the guide that they couldn't hear.

(2-8b)The tourists signaled to the guide that they couldn't hear.

(2-9a)Joe included the package for Susan.

(2-9b)Joe carried the package for Susan.

In (2-7a), the PP, "on that rack", can be attached to either the VP headed by "wanted" or to the NP headed by "the dress". (2-7b) has ambiguity similar to (2-7a). The only difference in (2-7a) and (2-7b) is in the main verb. But the attachment to the NP is preferred in (2-7a) while the VP is the preferred attachment point in (2-7b). A similar situation can be seen in (2-8). The only difference in (2-8a) and (2-8b) is also the main verb. In (2-8a), the preference is that the phrase "that they couldn't hear" is analyzed as the relative clause which is attached to the NP "the guide". But the secondary clause "that they couldn't hear" is analyzed as the complement clause of the VP headed by "signaled" in (2-8b). (2-9) shows a similar ambiguity resolution as in (2-7). The ambiguity resolution phenomenon shown in (2-7) to (2-9) can not be explained using the parsing model of Frazier and Fodor because the sentences have the same syntactic structure but the ambiguity is resolved in different ways.

FBK sought the explanation of these phenomenon from the lexical component of the grammatical system in the human sentence processing system. They argued that each verb has various lexical forms which have different strengths and the strongest forms determine the preferred syntactic analysis. Figure 2.5 shows the lexical form of the verbs appearing in the above sentences. The first form is stronger than the second form in each verb. They did not explain what determines the strength of the lexical forms but they conjectured that the frequency of usage might contribute in part. But this theory can not be applied if the ambiguity is not related to a verb as in (2-10):

(2-10)That silly old-fashioned .....

```
want:      < (SUBJ), (OBJ) >
           < (SUBJ), (OBJ), (PCOMP) >
position:  < (SUBJ), (OBJ), (PCOMP) >
           < (SUBJ), (OBJ) >
object:    < (SUBJ), (OBJ) >
           < (SUBJ), (OBJ), (COMP) >
signal:    < (SUBJ), (OBJ), (COMP) >
           < (SUBJ), (OBJ) >
include:   < (SUBJ), (OBJ) >
           < (SUBJ), (OBJ), (PCOMP) >
carry:     < (SUBJ), (OBJ), (PCOMP) >
           < (SUBJ), (OBJ) >
```

Figure 2.5 Lexical Forms of Verbs

FBK used the "syntactic preference" principle to solve this problem. It was argued that the alternative phrase structure categories in the expansion of a phrase structure have ordered priorities which determine the preference of the analyses. In (2-10), the subject NP analysis as shown in "That silly old-fashioned joke is told too often" has the higher priority than the sentential subject analysis as shown in "That silly old-fashioned jokes are told too often is well-known." Therefore the NP analysis is preferred to the S analysis. The following sentence (2-11) provides another example related to the current discussion:

(2-11)We discussed running.

"running" in (2-11) can be analyzed as either a noun as in (2-12) or a verb as shown in (2-13).

(2-12)We discussed the excessive running of races.

(2-13)We discussed running races excessively.

Because the noun category is ranked higher than the VP category, "running" in (2-11) is first analyzed as a noun.

Based on the two principles discussed so far (i.e. the lexical preference principle and the syntactic category preference principle), FBK formulated the final argument principle that is related to the closure problem as follows:

> **final argument principle** (Ford, Bresnan and Kaplan,1982): Give low priority to attaching to a phrase the final argument of the strongest lexical form of that phrase and to attaching any elements subsequent to the final argument. Low priority is defined here with respect to other options that arise at the end position of the element whose attachment is to be delayed.

Let's explain this principle using (2-7a) and (2-7b): In (2-7a), "the dress" is analyzed as the NP and it corresponds to the final argument of the strongest lexical form <(subj), (obj)> of "want". So the attachment of "the dress" to the VP "want" is delayed. Rather the other option that the NP2, "the dress", is attached to the partial complex NP1 is given the higher priority as shown in Figure 2.6.

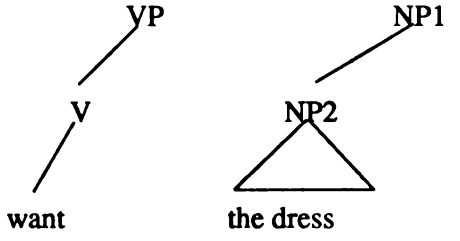


Figure 2.6 Example of Final Argument Principle

(The reason for this analysis is that if the NP, "the dress", is attached to the VP, then it will be closed and the coming PP, "on that rack", can not be attached to the NP headed by "the dress".) Then the PP "on that rack" is attached to the NP1 and the NP1 is attached to the VP. In the case of (2-7b), the final argument principle can not come into play to delay the attachment of the NP "the dress" to the VP because the object NP is not the final argument of the strongest lexical form of "position". In this situation, the NP "the dress" is attached to the VP rather than to a complex NP (this tendency was called *the invoked attachment principle*). Then the PP, "on that rack", is analyzed as a final

argument of the strongest lexical form. Thus its attachment to the VP is delayed. There are no more words following it and the attachment of the PP to the VP is the only choice. The model described in this section can be considered to be a serial model because the first alternative that conforms to the theory is chosen but the other alternatives are never considered.

2.2 Parallel Models

Parallel parsing models, which are directly opposite to the serial parsing models, also received much attention in psycholinguistics. One of the oldest parallel parsing model seems to be the model proposed by Fodor, Bever, and Garrett(1974) who suggested that all possible analyses are computed when an ambiguity is encountered. All possible analyses are carried out in parallel. An analysis that is inconsistent with the following input is dropped. When a clause boundary is reached, one analysis out of a set of the parallel analyses is selected based on some decision principles.

Forster's(1979) autonomous parsing model can also be considered to be a parallel model. In this model, phonological, syntactic, and message-level processors compose the language processing system. Each processor operates autonomously but the input into it is the information that is available in its level and the information generated from the next lower level. This definition implies that the syntactic processor computes all possible syntactic analyses as far as it can and the result is shunted to the next upper level (which is the message-level processor). The analyses in a level can be dropped or accepted by the next level processor. This model can explain the dumb, fast, automatic process of each linguistic processing level such as the lexical access. Recently there has been serious research on parallel models that challenge serial models that have until now prevailed. These parallel models will be explained in the following sections.

2.2.1 Discourse Model

Crain and Steedman(1985) proposed a model that directly argued against the structural preference and lexical preference models(Frazier and Fodor, 1978; Ford, Bresnan and Kaplan, 1982). This model is called the discourse model (or context model) because it is argued that the access to the contextual information is what disambiguates the local syntactic ambiguity. It is obvious that Crain and Steedman's original idea is that the plausibility based on the broad sense of semantics is the thing that works for the disambiguation. This broad sense of semantics includes both the general world knowledge and contextual information. But their paper concentrated on the explanation of using only the contextual information. They mostly argued for the influence of specific conversational context upon the syntactic disambiguation.

Crain and Steedman described two possible interactions between syntax and semantics: the weak and strong interaction. In weak interaction, the syntactic component from time to time proposes several alternatives of syntactic analysis to the semantic component either serially or in parallel. Then the semantic component selects one alternative by comparing the evaluations or referents of the alternatives. In this interaction the syntactic processing level independently proposes the alternatives of the local structural ambiguity, while the semantic level disposes among these alternatives. On the other hand, in strong interaction, semantics and context influence which alternatives are actually proposed by the syntactic level in the first place. Semantics and context can either set the order of alternatives being proposed serially or make some alternatives entirely unavailable.

Crain and Steedman's choice out of the two interactions was the weak interaction. They also argued that the alternatives should be proposed in parallel. They opted for a very intimate interaction between syntax and semantics, for example, an interaction for every word. This means that syntax appeals to semantics frequently and immediately. They called this interaction *the radical weak interaction*. One important point here is

that the alternatives are proposed in parallel in contrast to the serial models such as minimal attachment. The reason they went for the parallel model is that their experiment did not show any residual effect of the structure when semantics and context support the two alternatives equally. The most general principle they proposed is described as follows:

> **The principle of a priori plausibility** (Crain and Steedman,1985): If a reading is more plausible in terms either of general knowledge about the world, or of specific knowledge about the universe of discourse, then other things being equal, it will be favored over one that is not.

One specific knowledge they most specifically pointed out was the knowledge about the referents that are present in the listener's mental model of the universe of discourse. Consider (2-14a) and (2-14b) to see this point:

(2-14a) Put [the block in the box] [on the table].

(2-14b) Put [the block] [in the box on the table].

If there is actually a block that is in the box or that kind of block has been mentioned, then (2-14a) would be the preferred analysis, because there is a referent in the mental model. But if the (2-14a) analysis fails to produce a referent, but (2-14b) succeeds in finding a referent for "the block", then (2-14b) will be the analysis that is chosen. This heuristic was stated as follows:

> **The principle of referential success** (Crain and Steedman,1985): If there is a reading that succeeds in referring to an entity already established in the hearer's mental model of the domain of the discourse, then it is favored over one that does not.

This principle is a special case of the principle of a priori plausibility. In Crain and Steedman's model, the source of semantics is the mental representation of a specific conversational context including things that have been mentioned and things that have been implied in the conversation rather than the world itself. In the cooperative

conversation, the hearer will introduce the referents into his or her mental model as well as recognizing the reference to items already introduced in the conversation. The reason for this can be explained by the cooperativeness between the conversants(Grice, 1975). In other words, the hearer will add the referents for the presuppositions of the speaker that are not satisfied with his mental model. Based on this consideration, another heuristic is suggested which is a more generalized version of the principle of referential success:

> **The principle of parsimony**(Crain and Steedman,1985): If there is a reading that carries fewer unsatisfied but consistent presuppositions or entailments than any other, then, other criteria of plausibility being equal, that reading will be adopted as the most plausible by the hearer, and the presuppositions in question will be incorporated in his or her mental model.

This principle is still a special case of the principle of a priori plausibility. The principle of parsimony can be used to explain the garden path effect in (2-15):

(2-15)The horse raced past the barn fell.

In (2-15), local syntactic ambiguity exists in the main-verb analysis or the reduced relative clause analysis of "raced". In the case of reduced relative clause analysis, the following presuppositions are required in the so-called null context: (1) the set of horses (2) some horses were raced past the barn. But the main verb analysis does not need the presupposition of (2). Therefore it is argued that the main verb analysis is chosen over the reduced relative clause analysis and the garden path occurs.

(2-16)A horse raced past the barn fell.

In (2-16), no presuppositions such as (1) and (2) above are needed and thus it is expected that the possibility of the occurrence of garden pathing will be less than (2-15). This argument was supported by their experiment.

One important point that was argued by Crain and Steedman is that there can not be the null context in the sense of "no horses has been mentioned before (2-15) or (2-16) is told to the hearer". The reason they provided was that a null context like this can not be neutral to the two possible analyses because the two readings may differ in the ease with which the hearer may have to set up the referents because of the different number of presuppositions the readings invoke. They said that the argument for the structural preference model is actually based on the null context which is not a real neutral context and thus the experiment data provided by the structural proponents lose credibility. They argued that the result that the structural proponents obtained is actually due to the effect of context.

Crain and Steedman also argued that the source of semantics in the analysis of (2-15) is not the prior plausibility based on the general world knowledge but the presuppositions in the context of discourse. According to their theory, the control of the context can not only induce the garden path effects but also overcome them. Consider (2-17):

(2-17)It frightened the child that John wanted to visit the lab.

The usual preference is that "that .... lab" is analyzed as a complement clause rather than as a relative clause attached to "the child". But if the statement, "There was an explosion", was mentioned just before (2-17), it was found that the preference switched. In (2-18), the context even induced the garden path effect:

(2-18)Context: Several children heard an explosion.

It frightened the child that John wanted him to visit the lab.

2.2.2 Ranked Parallel Model

Gorell(1987) attempted an experiment to investigate the issue of serial/parallel model related to the resolution of the local syntactic ambiguity. He used an experiment called *the syntactic priming paradigm*. He argued that the experiment used by the serial model proponents(i.e. the acceptability judgement task or the eye movement

experiment) is not appropriate for investigating the issue. In the case of the experiment which used the acceptability judgement task reported in Frazier(1978), the acceptability judgement is made at the end of the sentence, which makes it hard to examine what is actually going on during the time between the onset of ambiguity and the appearance of the disambiguating material. Frazier argued that the poor and slow performance for the non-preferred reading of an ambiguity is due to the reanalysis of the input after the first analysis has failed. But Gorell proposed another possible explanation that can replace Frazier's claim. He argued that the poor and slow performance might be due to the complexity of the structure that exists in the sentences in which non-preferred reading is the correct reading.

(2-19)John knew the very old woman was ill.

In (2-19), the appearance of "was" makes the parser construct a clause which is a complex task. According to Gorrell, this complexity causes the parser to become slow. He attributed the poor performance to the processing load due to the complexity of the structure rather than the requirement for reanalysis.

Frazier and Raynor(1982) reported an experiment that provided a stronger support for the serial model. The experiment measured the eye fixation and movement during the reading of the sentences. Even for this data, Gorell suggested that the data can be accounted for by the increased processing load due to the structural complexity. Gorell suggested an experiment in which "knew" in (2-19) is replaced by "thought" as shown in (2-20):

(2-20)John thought the very old woman was ill.

(2-20) is an unambiguous sentence having the same syntactic structure as (2-19). Gorell argued that if (2-20) shows the similar poor performance as shown in (2-19) this shows that his argument is correct. If the serial model is assumed, (2-20) should show better performance than (2-19) because (2-20) does not require reanalysis. But unfortunately there is no data available on (2-20).

Gorell's experiment which used *the syntactic priming paradigm* is similar to the experiment for the problem of lexical access done by Swinney(1979). Swinney showed that a semantically ambiguous word can prime a target related to only one of its various meanings in any biasing contexts. This experiment made parallelism of lexical access widely accepted. Gorell argued that the syntactic category can prime a target that is associated to this category. In his experiment, a target word is presented during the reading of a sentence with a local syntactic ambiguity before the disambiguating words appear. The subjects are asked to decide if the target is an English word or not immediately after the appearance of the target word. He said that a significant priming effect was observed for targets belonging to the categories predicted by the structure associated with the non-preferred reading of the ambiguity. He interpreted the data in this way: the non-preferred reading is actually computed and proposed in parallel with the preferred reading. He also used unambiguous counter parts of the ambiguous sentences in the experiment and found that the response time and correct ratio is almost the same as the ambiguous counter parts with non-preferred reading. The use of unambiguous control like this excluded the possibility that the priming effect for ambiguous contexts was the result of the rapid reanalysis.

Using the result of his experiment, Gorell proposed a parsing model dubbed *the ranked parallel* model. In this model, all possible alternative analyses of a structural ambiguity are computed and proposed in parallel to the next processing level. To reach a reconciliation with the serial model proponents, he suggested that the multiple alternatives are proposed in parallel but they are rank-ordered according to the complexity of the structure and they are considered by the next level in that order. The major point is that syntactic level computes and proposes the alternatives in parallel but they are examined serially in the next level.

2.2.3 A Parallel Model with Conceptual Selection Hypothesis

Kurtzman(1985) made an attempt which argues for the parallel parsing model. Instead of the structural preference (i.e. the local and minimal attachment principles), lexical preference, and discourse preference used in other models, Kurtzman suggested that expectation for the conceptual information is what determines the resolution of the ambiguity. This was called the *conceptual selection hypothesis*. Consider a situation in sentence (2-21) where someone is "keeping" something:

(2-21)The woman kept the dogs on the beach.

(2-22)The woman discussed the dogs on the beach.

Kurtzman argued that the conceptual information expressed by "kept" expects to have some conceptual information about "where" (something is kept) in addition to the information about "who" and "what". Because of this expectation, "on the beach" is associated with "kept" instead of with "the dogs". This seems to be a reasonable argument. It is interesting to see how he handled the case in (2-22). In the situation expressed by "discussed", the conceptual information for the location where the discussion occurred is not so strong as in the situation of "keeping" in (2-21). Rather the more elaborated information of the object of discussion is what is expected conceptually. He argued that this different expectation is what causes "on the beach" to be attached to "the dogs" instead of "discussed".

The information sought by the expectations that affect the resolution of ambiguity in Kurtzman's model is of a fully conceptual sort at the level of real world knowledge. It is quite specific information which can not be mapped one to one onto some type of structural semantic representation. One of the parsing approaches that utilizes conceptual information is the conceptual analyzer developed by Schank's group (for example, Riesbeck and Schank, 1978; Birnbaum and Selfridge, 1981). But it seems that their representation language, the Conceptual Dependency, is too limited to be used for representing conceptual information discussed in Kurtzman's model.

The use of conceptual information in Kurtzman's model seems to be similar to the use of mental representation in Crain and Steedman's discourse model but the former model emphasizes the conceptual expectation as it can be shown in the sentences (2-23a) - (2-23c):

(2-23a) The horse raced past the barn fell.

(2-23b) A horse raced past the barn fell.

(2-23c) Horses raced past the barn fell.

In (2-23a), "the horse" has the determiner and it is already in focus. Thus "the horse" expresses old information. The participants in the dialogue have some information about this horse. But "a horse" in (2-23b) or "horses" in (2-23c) expresses new information whose nature is not known yet. It seems that there will be more expectation of new information about an entity that is not known well. For this reason, we can say that "raced past the barn" is more likely to modify the subject in (2-23b) and (2-23c) rather than in (2-23a).

But this conceptual selection hypothesis is not out of trouble. Consider an example in (2-24) and (2-25):

(2-24)We discussed running.

(2-25)We considered running.

In (2-24), it is preferred that "running" is analyzed as the simple noun (running as a sport) instead of as a VP dominating a simple verb (engaging in running). But in (2-25), the preferred resolution is the VP. As Kurtzman acknowledged, the resolution does not seem to be based on the expectations for particular conceptual information in these sentences, because we can discuss "running" as a sport as much as "running" as "engaging in running". What matters in this case is the subtle difference in relative plausibility. It seems to be more plausible to discuss a sport of running than to discuss engaging in running. But Kurtzman argued that adding the general plausibility as a source of

disambiguation does not undermine his conceptual selection hypothesis model because there are cases which the general plausibility can not handle as shown in (2-26) and (2-27). In (2-26) "discussing on the beach" is as plausible as "dogs on the beach", which is true in (2-27), too.

(2-26)The woman discussed dogs on the beach.

(2-27)The woman kept the dogs on the beach.

Another difficulty for the conceptual selection hypothesis can be found in (2-10). The general preference is that "that" in (2-10) is analyzed as a determiner rather than as a complementizer. His explanation for this case is based on the following observation: Analyzing "that" as a determiner (which begins a main clause's subject NP) invokes an expectation for a term referent but analyzing it as a complementizer (which begins a complement clause) invokes an expectation that the topic of the sentence is a proposition. He argued that the better conceptual expectation is to take a simple object or idea rather than a whole proposition as the topic of a sentence. This explanation seems to be at least as good as that given by other parsing models introduced earlier.

So far we examined the conceptual selection hypothesis. Now it is necessary to consider how the conceptual information level works with the syntactic analysis level in Kurtzman's parsing model. In the Schankian conceptual analyzers, syntactic analysis is held to a minimum and conceptual analysis is given the major role. But Kurtzman proposed a model similar to the autonomous model. All possible syntactic analyses are computed on the syntactic level and this result is input into the level that is in charge of handling conceptual information. Then the conceptual level chooses one analysis which is most appropriate based on the conceptual expectation. It was argued that it would be hard to conceive a model in which the syntactic processing is interrupted by the conceptual level so that the parser can consider only one syntactic analysis (out of several) that is most appropriate conceptually. The reason for this is that there are no systematic relations between the conceptual information and the syntactic information which could be

used in such a direct conceptual determination of the parsing process. Kurtzman also objected to the scheme in which one analysis is proposed by the parser and the conceptual mechanism either rejects or accepts it (in the case of rejection, the parser proposes the next possible analysis, and so on. Note that this is the approach used by the parsers of interleaved semantic processing). His reason for the objection is that there can be so many single analyses of the unambiguous and generally plausible sentences that fail to express the conceptual expectation.

The architecture of the model he proposed is the so called, *Immediate parallel analysis(IPA) with strong parallelism.* "Immediate parallel analysis" means that all possible analyses are computed from the onset of the ambiguity instead of delaying the analysis (which is called the delayed parallel analysis as shown in (Marcus,1980)). The strong parallelism means that each of the possible analyses is carried on until it is removed based on further input material. The opposite of the strong parallelism is called momentary parallelism. In momentary parallelism, the ambiguity is resolved at the onset of the ambiguity. In other words, one analysis is chosen immediately out of the multiple possible analyses based on some selection criteria.

Kurtzman's parsing model can be described as follows. At an ambiguous point, all possible analyses are computed. Out of these alternatives, those that are best according to conceptual expectation are retained and the others are removed. If the confident decision can not be made because of the lack of conceptual information, the decision is delayed until it is possible to. The analyses that have not been removed stay active as the parsing is going on.

The reason that strong parallelism is adopted instead of the momentary parallelism can be explained using sentences (2-28) and (2-29):

(2-28)The executives discussed the problems in the factory.

(2-29)The executives discussed the problems in the restaurant.

In the case of the momentary parallelism, the attachment point of the PP (even if it is not complete yet) should be decided as soon as "in" is read. But this is not what actually happens. In (2-28), the PP is attached to the NP "the problems", while it is attached to the VP in (2-29). In other words, the attachment point is determined by the content of the PP, which implies that the ambiguity can not be resolved at "in" but after completing the PP. This example shows that strong parallelism should be adopted instead of the momentary parallelism. The advocates of the serial model (for example, the Frazier and Fodor model) might argue that a single analysis is chosen first and backtrack is done if it proves that the selection was wrong. But note that neither (2-28) nor (2-29) are garden-path sentences which require backtracking (but the matter becomes complicated if we consider that the slight garden-path sentences do not require the conscious backtracking).

The important question that needs to be investigated under strong parallelism is the point at which the ambiguity is resolved (note that the source of disambiguation in Kurtzman's model is conceptual expectation). Kurtzman conjectures that the point of disambiguation might be when the conceptual mechanism can confidently evaluate which analysis is an appropriate one. The resolution point can vary according to the lexical content of the following input material. One example for this phenomenon can be found in the following sentences.

(2-30)The horse raced past the barn fell.

(2-31)The aluminum strengthened to a great degree.

(2-32)The children observed here our plant fields.

In each of the above sentences, the first verb can be analyzed as either the main verb of the sentence or the past participle leading the reduced relative clause. After reading this verb, the experimental data reveals the following result: in (2-30), the main verb analysis is the prefered analysis and garden pathing occurs; No garden-path effect occurs in (2-31); But in (2-32), garden-pathing occurs in favor of the reduced relative analysis.

This data suggests that the decision can be made prior to the reception of the disambiguating material.

2.3 Summary

In this chapter, the important parsing models proposed in the psycholinguistic literature have been studied. The two major features that distinguish the models are serial/parallel computation of the syntactic alternatives and the type of information that determines the preferred analysis. In the serial model, the parser computes and proposes the syntactic alternatives serially. In the parallel model, the alternatives are computed and proposed in parallel and it is up to the next processing level to choose one out of the multiple analyses of an ambiguity.

Information that is used to select an alternative is important in defining a parsing model. The complexity of the structure is used in Frazier's minimal attachment, the lexical preference is used in Ford, Bresnan and Kaplan's model, discourse information is used in Crain and Steedman's discourse model, and conceptual expectation is used in Kurtzman's parallel model. Note that the models that use semantics to resolve local syntactic ambiguity go for parallelism. (Semantics here does not include the lexical information used in the lexical preference model). One of the reasons for this tendency is the autonomy of language processing. The input into the syntactic level consists of the output of the next lower level(lexical level) and the syntactic knowledge available in the syntactic level. The syntactic level does whatever processing it can and the output is input into the next higher level (let us assume this is the semantic level). Because the information available to the syntactic level and the semantic level is so distinct it is not likely that the syntactic level's processing is interrupted by the semantic level.

Gorell used an on-line experimental technique to show that parallel processing is actually going on at the syntactic level. But it is too early to decide that one model is to be chosen over another model.

CHAPTER 3

FRAMEWORK OF THE NEW APPROACH

We will describe the overall framework of the SYNSEM parser which was proposed in Section 1.4. The objective of the parser is to update the approach of interleaved semantic processing to achieve integrated parsing by approaching from syntax-oriented parsing. The major attraction that integrated parsing has is to use semantic information as much as possible. The previous integrated parsing advocates argued that the full use of semantics can be achieved only by making the semantic analysis the major analysis and restricting the amount of syntactic analysis. Integrated parsing could not be considered to be an approach that uses both full-fledged syntax and semantics. The fact that the integrated parsers are biased too much toward semantics was corrected by Lytinen's new integrated parsing approach which was shown in the MOPTRANS parser (Lytinen, 1984). But his approach still puts more emphasis on semantics and uses syntax only as the filter that applies after the semantic analysis.

We take the position that the syntactic processing level is lower than the semantic processing level and is an autonomous processing level which runs more or less independently. We are not aware of any psycholinguistic research that claims that the semantic processing is on the lower level than the syntactic processing. Thus it seems to be natural to start the design of a parser from the syntax-oriented parser and then add more on-line semantic processing to it. This approach resulted in the interleaved semantic processing approach. But the use of semantics in interleaved semantic processing is not enough to satisfy the advocates of integrated parsing.

Kurtzman's conceptual selection hypothesis and the immediate parallel analysis with strong parallelism provides the basic framework for the new parser that will be developed in this thesis.

3.1 General Framework of the Control

The starting point to approach the desired parser is interleaved semantic processing. The control loop for interleaved semantic processing is shown in Figure 3.1 (let's call this model ISP1).

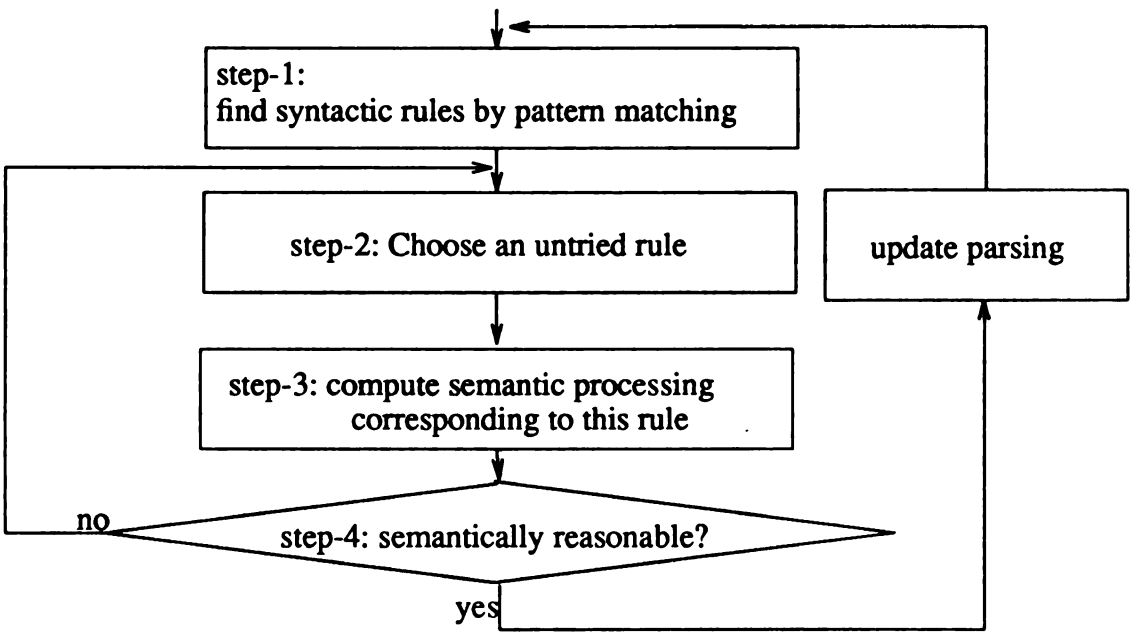


Figure 3.1 Interleaved Semantic Processing - 1
(Loop terminated at end of sentence)

It is clear from this figure that the processing is compatible with the serial models discussed in the previous chapter. If the rules found at step-1 are sorted and tried in the order of the simplicity of the structure at step-2 , then the approach shown in this figure is the same as the structural preference model such as minimal attachment. If the rules are tried in the order of lexical preference in step-2, this approach would be the same as the lexical preference model. It is also conceivable that a parser adopts no specific selection strategy in step-2 but rather a next rule is chosen randomly from the rules that have

not been tried yet. This is the method that was usually used in natural language parsers that used interleaved semantic processing.

The parallel parsing models suggested that all rules are proposed and processed in parallel. We can take this suggestion as one way of changing the control strategy of interleaved semantic processing. The control strategy updated according to this suggestion is shown in figure 3.2 (let's call this model ISP2).

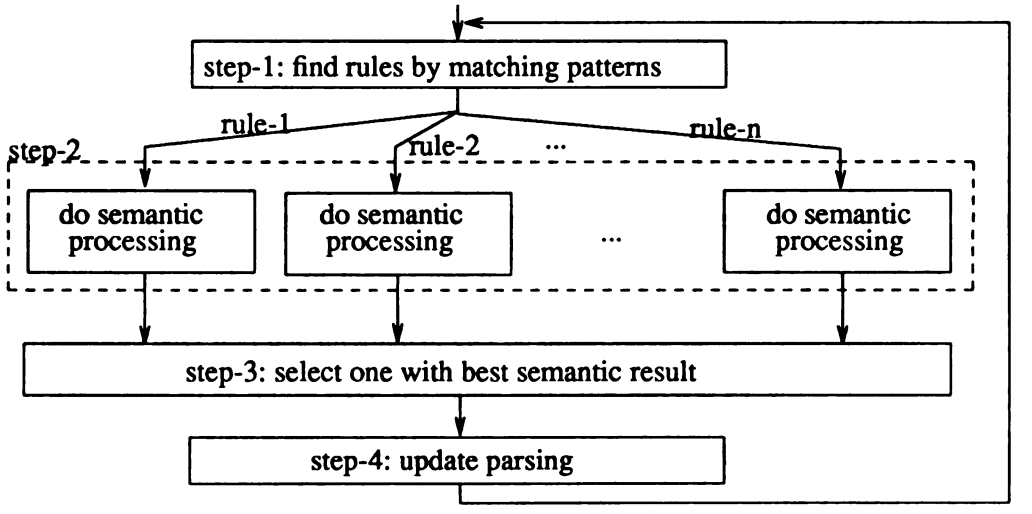


Figure 3.2 Interleaved Semantic Processing - 2

All rules (or syntactic alternatives) found by the syntactic processor at step-1 are provided to the next higher level at step-2 and these alternatives are processed and compared in parallel according to the information available in this level. This information will be the discourse context in the discourse model and conceptual information in Kurtzman's model. In the case of Gorell's model, this information would be the structural complexity (provided by the lower level, the syntactic level) rather than the sort of semantic information. Based on the processing at step-2, one rule that has obtained the best score at step-2 will be chosen at step-3. This rule will be used to update the parsing at step-4 and the parsing goes to the next round of the loop.

The improvement in going from ISP1 to ISP2 is that the alternative with the best semantic result can be selected. Thus the probability of making an error in selecting one alternative can be reduced compared with ISP1. ISP1 has the step of checking the semantic effect of the chosen alternative but it is not an appropriate architecture for selecting the best alternative. As far as the speed of the parsing is concerned, it is not clear that ISP2 is better than ISP1. It is usually acknowledged that the semantic processing takes more computation than the lower level processing such as the lexical or syntactic processing. But ISP2 encourages more semantic processing than ISP1.

According to Kurtzman's IPA with strong parallelism, the point of disambiguation depends on the time that a confident decision can be made based on conceptual information. This means that a decision may not be made just after finding the rules and considering the semantic effects of these rules (at step-2 and step-3 in Figure 3.2). It may be several words later that one alternative can be determined to be better than the other. In some cases two rules might have the same semantic processing result or in other cases some rules might never invoke semantic processing but only the build-up of the syntactic structure. These considerations indicate that the output of step-3 in Figure 3.2 may have to be more than one rule. This implies that there might be more than one way of parsing (called the branches of parsing) active at the same time. The control flow shown in Figure 3.3 (named to be ISP3) takes these considerations into account. Note that there can be more than one branch at the beginning of the control loop. Finding more than one rule at the pattern matching step(step-1) means that the number of branches will be increased by that amount (if only one rule is found, the number of branches would not change).

But there is a problem that needs more consideration in ISP3. The problem is related to the question of when the branches are compared and filtered. According to Figure 3.3, the point of comparing and filtering the branches comes always right after the semantic processing is done for the rules found during the pattern matching step.

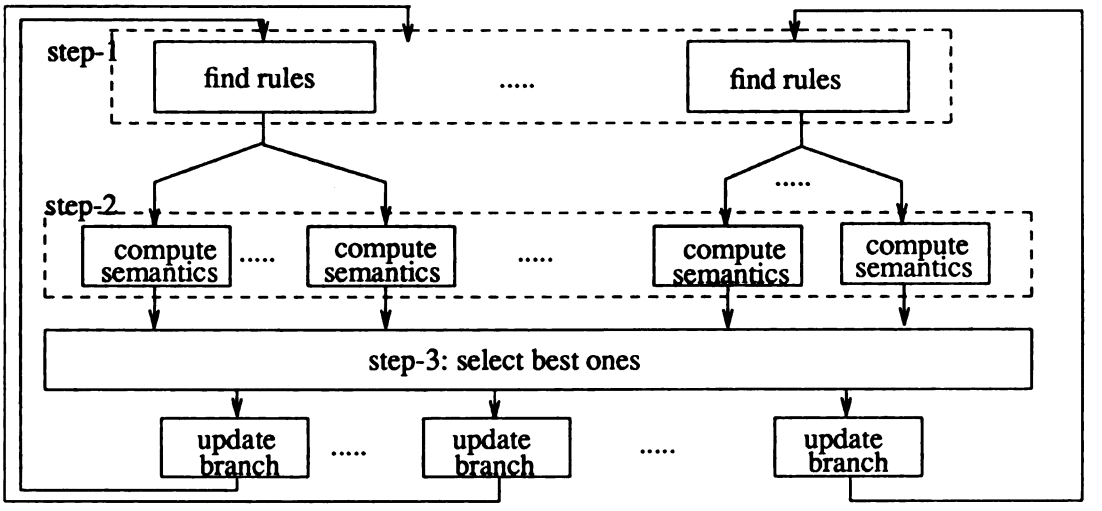


Figure 3.3 Interleaved Semantic Processing - 3

This might cause the system to compare the branches even if insufficient semantic processing results have been collected, which also means that the branches might be compared too frequently. Based on this consideration, the control is modified as shown in Figure 3.4 (this is named to be ISP4).

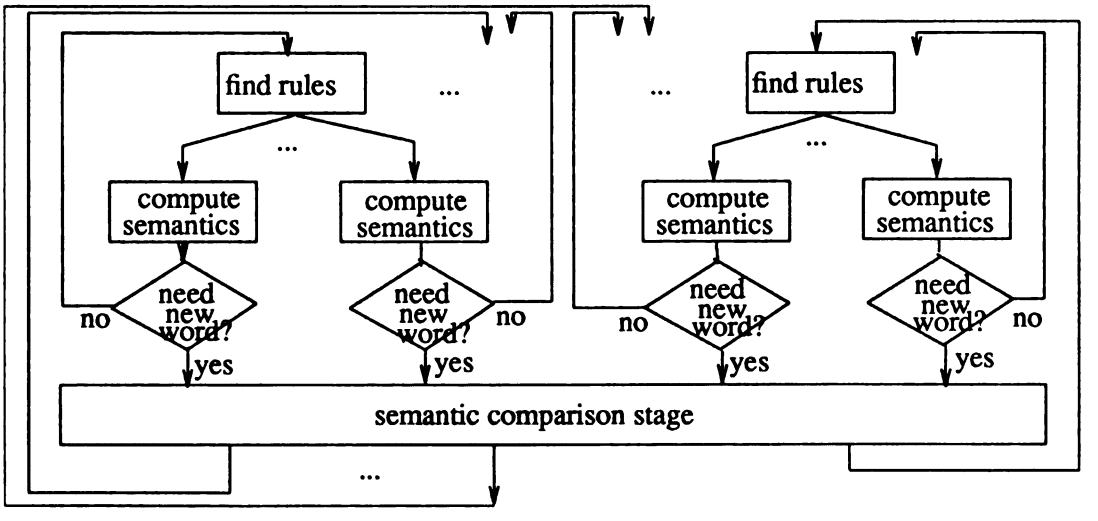


Figure 3.4 Interleaved Semantic Processing - 4

Looking at the figure, we can notice that another loop is added in the control. In the

inner loop, each branch goes through the following steps: (1) find all rules whose patterns match the working memory (called the pattern matching step); (2) for each rule found in (1), a new branch is created and the semantic effect of the rule is computed and the parsing state of the branch is updated; (3) check to see if the branch needs to read a *new input word* and go into the semantic comparison stage if it does (if the branch does not need a new word, the control returns to the beginning of the inner loop as shown in the figure). The branches that reach the semantic comparison stage wait until all the branches reach this stage. When all branches in the system have reached the comparison stage, the actual processing of the stage begins. During this stage, the branches are compared using the semantic processing result collected after the last semantic comparison stage. Only those branches with the best semantic processing result are kept and others are thrown away (they are actually stored in the backup stack). If there is a branch which has no semantic processing result collected since the last semantic comparison stage, this branch is also kept as an active branch. The detailed operations and implementation of ISP4 will be explained throughout the thesis.

3.2 Configuration of the Parser

Figure 3.5 shows the configuration of the SYNSEM parser. The program modules are indicated by the ellipses and the data structures are enclosed in the boxes. The regular arrows indicate the flow of control and the access of data structures is shown with the dotted arrows. The monitor(MON) is in charge of reading each input word and calls the other program modules to process the input. It supervises the control of the parser in a synchronized way to successfully achieve the parsing.

The syntactic analysis component(SYN) is the program module that does all the processing related to syntax. For example, matching the patterns of the syntactic rules (henceforth, "rules" means the syntactic rules) against the state of parsing to find the rules to be triggered and building the syntactic trees are handled by SYN. The semantic

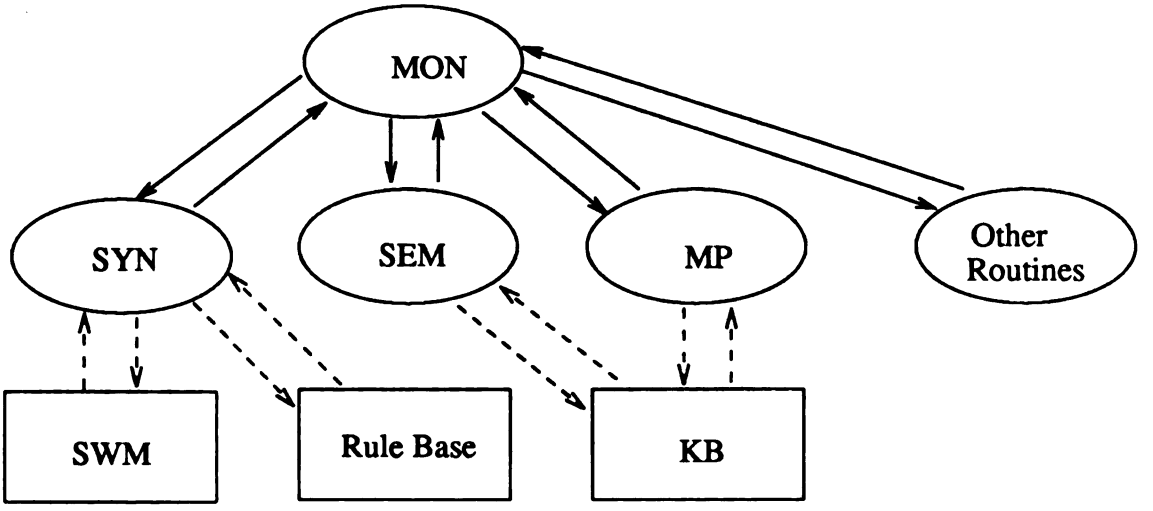


Figure 3.5 Configuration of SYNSEM

processing component(SEM) builds a semantic interpretation for the input sentence based on the analysis of SYN. The most important function of SEM to be discussed in this thesis is to find the paths in the knowledge base to get semantic information and compare these paths to find the best alternative out of multiple analyses proposed by SYN. The marker passing module(MP) is the routine to pass the markers starting from the concept representing input words. It detects the collisions so that SEM can use them.

SYNSEM uses three major data structures: the syntactic working memory(SWM), the rule base(grammar rules), and the knowledge base(KB) which is a semantic network. The SWM consists of two lists called CLIST and CASH and some global variables. CLIST contains the syntactic trees that are being built during parsing. CASH is the auxiliary list that stores syntactic trees temporarily. The rule base contains all grammar rules which represent the grammatical knowledge of the parser. The grammer rules used in SYNSEM are similar to those of Parsifal (Marcus, 1980).

The rules are written in a domain independent way so that the syntactic portion of the system can be portable between different domains. The knowledge base(KB) is a semantic network written in the KODIAK knowledge representation language (Norvig,

1986; Wilensky 1987; Wilensky & etc. 1986). The knowledge base encodes world knowledge that the system knows about the world with which it is dealing. The semantic power of the system depends on knowledge the knowledge base holds. The knowledge base is the part of the system that is dependent on the domain.

Currently SYNSEM can only handle single sentence input. In other words, it can not use the context available in a text consisting of multiple sentences. The output of SYNSEM is a set of knowledge base paths which represents the meaning of the input sentence.

3.3 Flow of Parsing in SYNSEM

In this section, the overall operation of the SYNSEM parser will be described so that we can see how it operates globally. A word of the input string is always read into the system by MON(Monitor). If this word is an *open word*, MON calls the MP with the word being passed. Open words are the words that can represent an entity in the world such as objects, actions, or states. For example, "train", "apple" and "eat" are the open words while the prepositions, articles, etc. are examples of closed words.

After marker passing is done, the word goes through the morphological processing and it is pushed into CLIST. This changes the content of CLIST, which might cause some rules' patterns to match CLIST. Then for each branch in the parser, MON calls SYN to do syntactic analysis. SYN matches the patterns of the rules in the rule base against CLIST. SYN returns the rules found during the pattern matching to MON. Note that more than one rule can be found if there is ambiguity. For each rule returned by SYN, a branch of parsing is created and the actions of the rule are executed. At this point some action might call the semantic processing component to do semantic processing corresponding to the rule.

Note that some rules may not have any action that invokes semantic processing. These rules (thus the branches) have no semantic processing result after the execution of

the actions. For example, the rule Parse-det which processes the word "the" has no semantic processing. It only attaches the word "the" to a node NS which is newly created (see Figure 3.6).

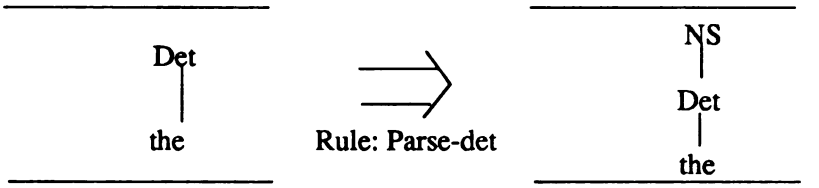


Figure 3.6 "the" Processing

It should be noted that the execution of the actions of a rule does syntactic processing such as updating the content of the SWM as well as calling the semantic component to invoke some related semantic processing. The details of the actions of the rules will be explained in the next chapter. The syntactic processing can be divided broadly into two tasks: finding rules whose patterns match the SWM of the branch, and constructing the syntactic trees in the SWM.

If there are some rules found during the pattern matching, the branches will be created for these rules and each branch executes the actions of the rule and the processing starts from the pattern matching again.

If there is no rule found during the pattern matching of a branch, the parsing has encountered a blocked situation. A blocked situation can happen in two ways. One blocked situation is not an actual blocked situation but it represents the situation where a new input is required to be input into CLIST. Let us call this the read-again situation. This happens when all possible processing has been done on the material in CLIST and thus no rule can be found that matches CLIST. The other blocked situation happens when the analysis corresponding to the branch is not compatible with the input string. Because of the incompatibility of the input string, no grammar rule matches CLIST. This is called the dead-end situation of the branch. MON can determine which blocked situation it is when no rule matches CLIST (refer to Appendix B). If the blocked

situation is not the dead-end, MON puts the element in CASH into CLIST if CASH is not empty. If CASH is empty, it is required to read the next input word. This situation is the point where the branch reaches the semantic comparison stage. The branch is pushed into the waiting pool so that the branches can be compared and filtered when all branches reach the stage.

When all branches get to the semantic comparison stage, the branches are compared and filtered according to the semantic processing results that have been collected since the last semantic comparison stage. At this stage, the branches whose semantic results are worse than other branches are removed (actually stored into the backup stack) and the other branches are kept as active. Note that the branches with no semantic processing result since the last semantic comparison stage are just retained without comparison. Then MON reads a new word from the input string and pushes it into CLIST of each active branch after morphological processing. From this point, each branch flows as explained so far.

Let's consider Figure 3.7 to see the creation and removal of the branches.

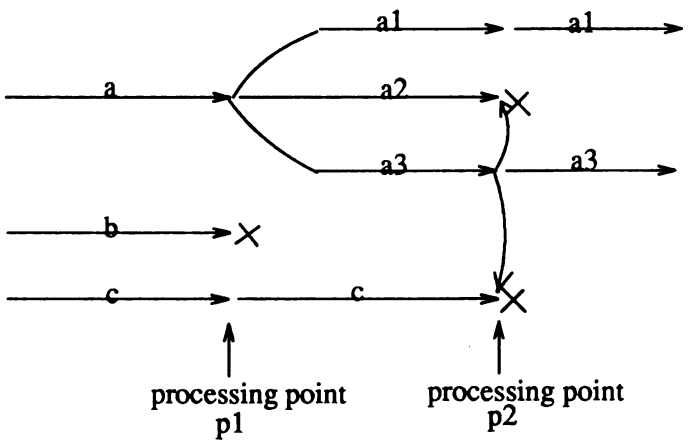


Figure 3.7 Lives of Branches

The branch b in Figure 3.7 is removed because it encounters the dead-end situation at p1. This is one way in which syntactic information is used for disambiguation. Branch a forks into three new branches (a1, a2, and a3) because three rules are found during the

pattern matching of the rules while branch c gets only one rule and thus one branch is forked. The rule matching and execution process is repeated until all branches request a new input word. At processing point p2, let's assume that all the branches reached the semantic comparison stage. At this stage, branch a2 and branch c are removed because their semantic processing results are worse than that of branch a3. But branch a1 is kept without comparison because it has either the same semantic processing result as branch a3 or no semantic processing result.

It is reiterated that a branch is completely thrown away if it proves to be a wrong syntactic analysis (this appears as the situation in which no rules are found during the pattern matching and it is determined to be the dead-end situation). This inactivation is because of the syntactic reason. If the branch is not one of the branches with the best semantic processing result during the semantic comparison stage, the branch is inactivated but stored in the backup stack (if the semantic processing result is too bad, the branch is immediately thrown away without being stacked). This inactivation is because of the semantic reason. Thus, the disambiguation is done for both syntactic and semantic reasons.

Usually all the branches except one will be removed during the semantic comparison stage, which means that the ambiguity is resolved completely. The period during which there is more than one branch (i.e. there is ambiguity) is usually short. Thus, the whole parsing consists of some periods during which there is only one active branch and other periods which have multiple active branches. This phenomenon is called *partial parallelism* and is illustrated in Figure 3.8. The branches that drop off during the semantic comparison stage are stored in the backup stack so that they can be used when it is necessary to do backtracking. A push down stack is used as the storage so that the most recently stored branch can be used first in the case of backtracking.

When all branches in the parser get to the dead-end, the parsing can not go any further. This situation is the dead-end of total parsing (note that this dead end is more

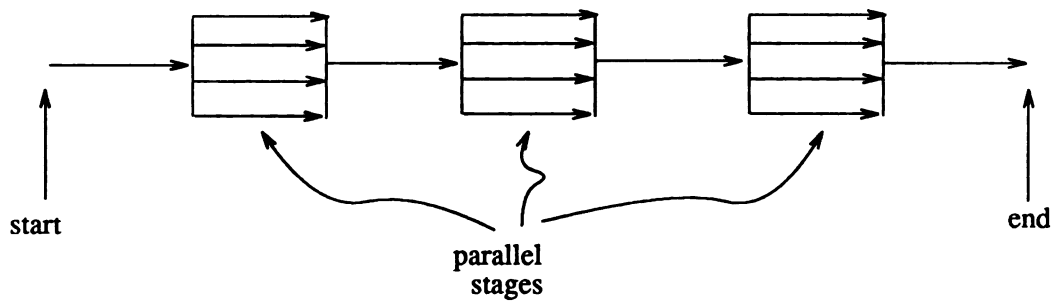


Figure 3.8 Partial Parallelism

serious than that of a branch). In this situation, the parser should back up to some previous point. This is done by popping one image of a branch that was stored in the stack and using it as the active branch. Figure 3.9 shows the flow of the parser in a precise form.

3.4 Conclusion

In this chapter, the general framework of the new parser with improved interleaved semantic processing has been developed. The basic approach is to make the parser consider all possible alternatives of syntactic processing at each step of parsing. This resulted in a system that allows multiple branches of parsing to exist at the same time. The disambiguation is done by removing the branches for one of two reasons. First, any branch proven to have a semantic processing result inferior to any other branch is suppressed during the synchronized semantic comparison stage. Second, any branch that is not compatible with the following input material is removed. This is realized in the way that the branch finds no rules during its pattern matching step.

The final version of the parser studied in this chapter corresponds to Kurtzman's parsing model which adopts the IPA with strong parallelism with the conceptual selection hypothesis. The configuration of the parser has been introduced and its detailed operation has been explained.

```
step 1: read a word and push into SWM of each active branch;
        (if all words in sentence has been processed, then stop.)
step 2: make a queue Q consisting of all active branches;
step 3: if Q is empty, goto step 4;
        pop a branch from front of Q;
        find rules that match SWM of the branch;
        if no rules found,
         then if read-again situation
                then if CASH is nonempty
                        then read from CASH
                        else push this branch to All_done_queue
                else throw away this branch
           else
             begin
              for each rule found
               do begin
                   execute actions of the rule;
                   fork a new branch for this rule;
                   insert this new branch into rear of Q;
                  end
             end;
        goto step 3;
step 4: (Semantic Comparison Stage)
        compare the branches in All_done_queue and
        choose the branches with best semantic results;
        (All branches chosen become active ones and
         others are thrown away.)
        goto step 1;
```

Figure 3.9 Flow of the Control

CHAPTER 4

SYNTACTIC ANALYSIS

The design of the syntactic analysis component has been influenced by the three considerations in SYNSEM. The first consideration is related to parallelism. The system should synchronize multiple branches that run in parallel and try to remove some branches based on both syntactic and semantic information. Thus SYN should be designed in such a way that management of parallel branches is relatively easy to implement. The second consideration is the early attachment strategy. A partial NP is created until the head noun is found. Then this partial noun phrase is treated as if it is the completed noun phrase. Thus the partial noun phrase can be attached to other structures as soon as its head noun is known. The NP's post modifiers after the head noun can then be attached to the NP incrementally. The third consideration is to delay some syntactic decision until the context is collected enough to make the correct decision. Based on these considerations, the rule-based syntactic analysis component has been developed.

4.1 Syntactic Working Memory

The syntactic working memory(SWM) is the data structure which contains the syntactic representations that are being built during the parsing of the input sentence. Two major components of the SWM are the two lists called CLIST and CASH. CLIST and CASH contain the syntactic trees. CASH is storage which contains syntactic trees that have been pushed out of CLIST temporarily as illustrated in Figure 4.1. When a word is input to the system, it is pushed onto the right side of CLIST after going through morphological processing (let us just distinguish the two ends of a list as the left and right ends). The actual entity that is pushed is a node representing the word. A node in the

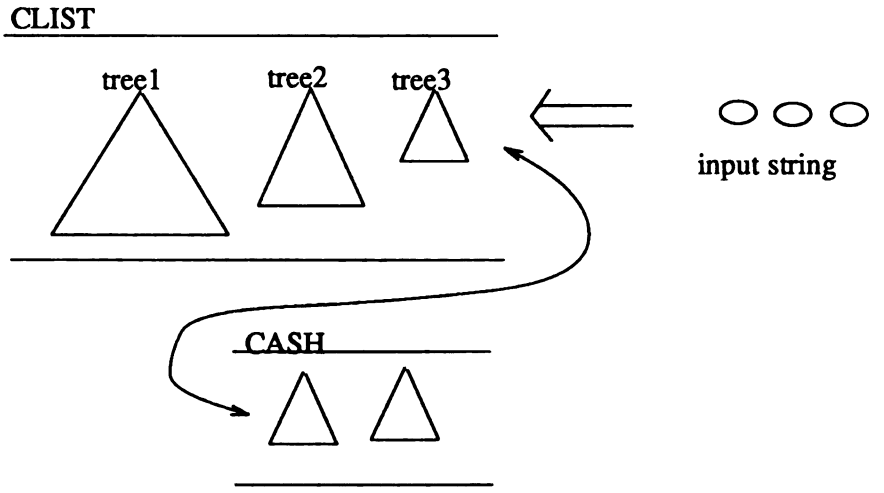


Figure 4.1 Syntactic Working Memory

trees is called an *Snode* which stands for a "syntactic node". The content of an Snode is shown in Figure 4.2. It shows a node for the word "eat" and "rocks".

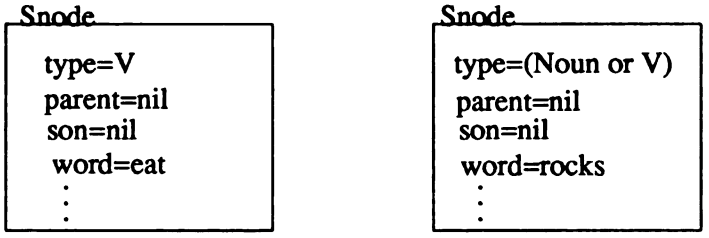


Figure 4.2 A Syntactic Node

Note that an Snode is itself a tree with one node. Syntactic trees are made with the Snodes. The structure of a syntactic tree is illustrated in Figure 4.3.

As explained above, each word that is input by MON is pushed into the right side of CLIST after it is converted into an Snode by the morphological routine. The rightmost tree of CLIST can be popped out and pushed into CASH by the request of an action of a rule. This request is made if the rule used the rightmost tree just to determine what analysis should be done for the second rightmost tree in CLIST (i.e. the rightmost tree is used only for providing context for making decisions related to the second rightmost tree). After the execution of this rule, that rightmost tree need not be in CLIST

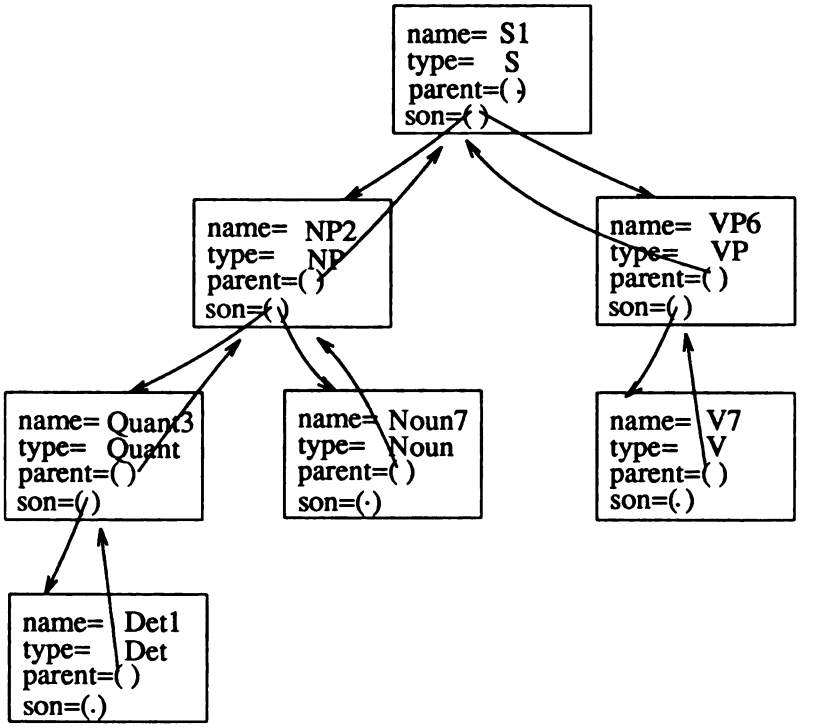


Figure 4.3 Tree in SWM

because the purpose of its being in CLIST has been fulfilled and the second rightmost node needs to become the rightmost tree of CLIST for further processing related to it. This fact will become clear when the format of rules is explained in the next section. So one of the actions of a rule can be the request for the transfer of a tree from CLIST to CASH. When the system needs to read a new input into CLIST, then the tree in CASH is pushed into the right side of CLIST if CASH is not empty. If CASH is empty, the input should come from the input string.

A new tree can be created and put into CLIST (usually a tree of one node is created). For example, the nodes of type "S", "NP" and "VP" are the nonterminal nodes which are created by the system. One important operation related to trees is attaching a tree in CLIST (usually the rightmost tree) to another tree in CLIST so that a bigger tree can be constructed.

The name CLIST reminds us of the data structure called "clist" used in the conceptual analyzers. But there is only a weak relation. "Clist" in conceptual analyzers is a list of semantic objects that are being built. CLIST of SYNSEM is a list of purely syntactic trees that correspond to the phrase structure trees of the sentence. CLIST is similar to the combination of the C-stack and the lookahead buffer in Parsifal (Marcus, 1980). The stack and lookahead buffer is merged into one list in SYNSEM. The right side of CLIST corresponds to the buffer cells and the left side corresponds to C-stack in Parsifal. However there is no preset boundary in CLIST that divides it into two parts. The use of the SWM will become clear from the examples that will follow later.

4.2 Structure of Rules

As mentioned before, SYNSEM is a rule-based analyzer that is similar to Parsifal. In this section, the rules used in SYNSEM will be explained in detail. Figure 4.4 shows the format of a rule in SYNSEM.

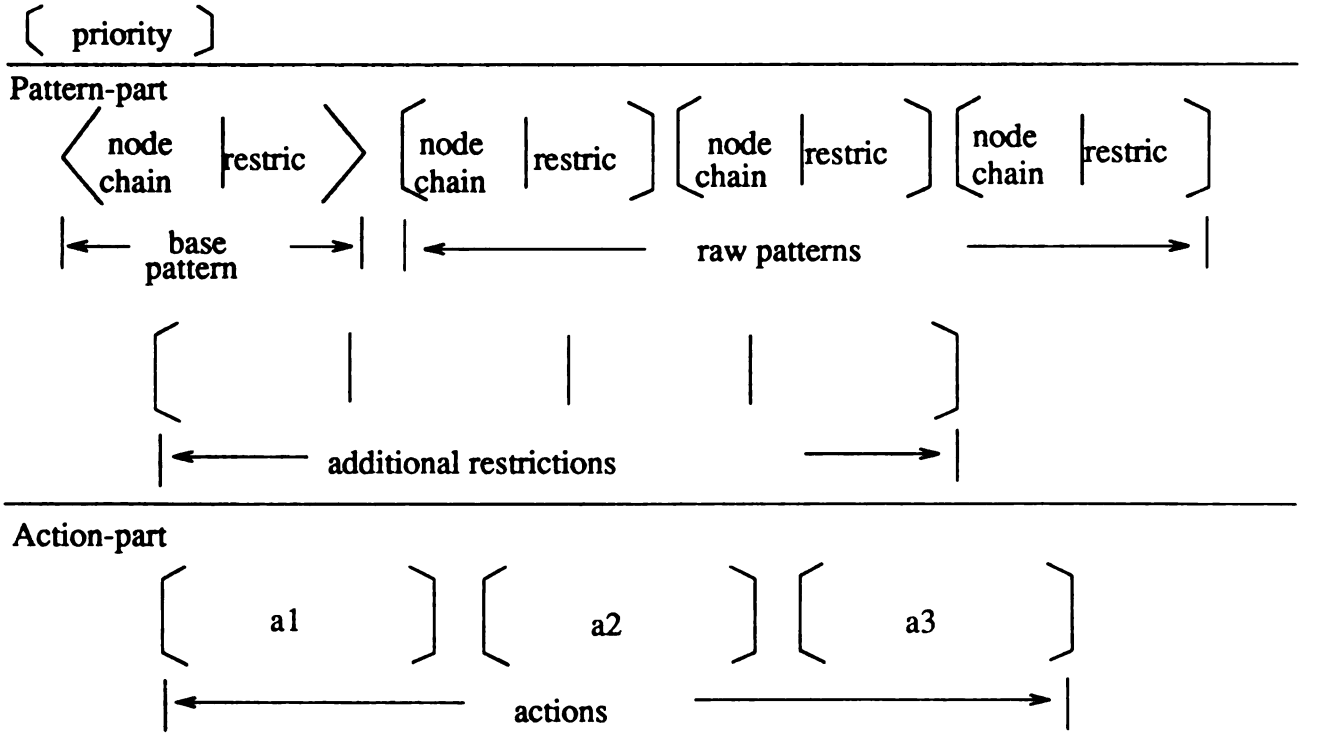


Figure 4.4 Format of Rules

A rule is composed of the priority specification, the pattern part and the action part. The priority specification part specifies the priority of the rule relative to other rules. The priority is specified by enumerating the rules which has higher priority than this rule and the rules which has lower priority than this rule. Note that numerical number is not used for the priority specification in SYNSEM. It is either an empty list or a list of the form, ((l rule1 rule2 ...) (h rule-a rule-b ...)). The first case is is used when there is no rule which has relative priority relationship with this rule (most of the rules are in this case). In the second case, the sublist starting with "l" contains the rules which should have lower priority than this rule and the sublist starting with "h" contains the rules which should have higher priority than this rule. The priority is used when more than one rule is found during the pattern matching. From the set of the rules from the pattern matcher, any rule with lower priority than a rule in the set need not be considered and removed.

The pattern part of a rule specifies the condition that should be satisfied by CLIST in order for the rule to be selected during the pattern matching. We also say that *the rule matches CLIST* if the pattern part is satisfied by CLIST. The action part of a rule consists of actions that need to be executed when the rule matches CLIST. The pattern part consists of two parts: the patterns and the additional restrictions. The patterns are matched against the trees in CLIST. There are two kinds of patterns. One is the base pattern and the other is the raw pattern. One pattern is matched against one tree. A raw pattern is a pattern which needs to be specified by only stating a property of the root node of the corresponding tree. This specification can only be done by either naming the type of the node or specifying a feature of the node. Note that the specification is only about the root node of the tree.

There can be zero or one base pattern in a rule. In the base pattern, other nodes in addition to the root node of the corresponding tree can be involved in the specification. This means that the inside of the tree is examined using the base pattern. The base pattern is usually used for checking *the state of parsing*. The state of parsing is implicitly

represented by the trees in the SWM. By examining the syntactic trees, the state of parsing can be gleaned. In the ATN parser, the state of parsing is represented by the state node of the ATN to which the parsing has reached. Parsifal used the active packets to represent the state of parsing. In SYNSEM, the rule base is not partitioned into packets as it is in Parsifal; the reason will be explained later. The root node of the tree that is matched against the base pattern is usually of type S(clause node). The state information can be checked by specifying the right side of the S tree as shown in Figure 4.5.

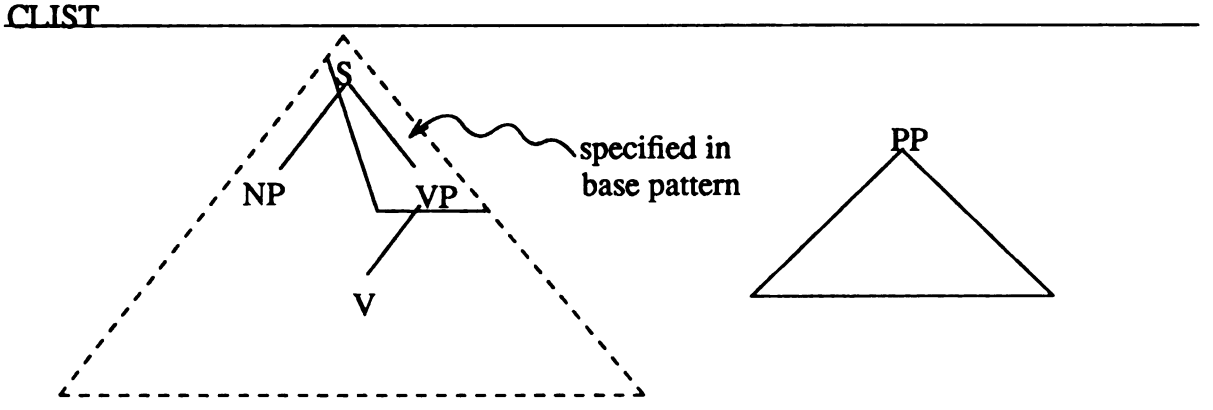


Figure 4.5 Specification of the Base Pattern

The base pattern can specify only a part of the tree that corresponds to this pattern. Note that it is not necessary to specify the shape of the full tree. This leads to the fact that matching the base pattern to a tree won't be computationally too costly. Let's take an example using Figure 4.5. For the rule that attaches a PP to a VP, the only portion of the S node that needs to be specified in the base pattern is "VP-S" as shown in the figure. It is not necessary to specify the NP or V node in the tree. The PP can be attached to a VP without worrying about other parts of the S tree because the existence of the VP guarantees that there is an NP under S (in the subject position) and a V under the VP. The base pattern should always be the leftmost pattern if it exists in a rule. The number of raw patterns can be from one to three and they are put consecutively on the right of the base pattern.

A pattern consists of two parts: the node chain and the node restriction as shown in Figure 4.4. The node chain specifies the skeleton of the portion of the tree (actually the right portion of the tree) that corresponds to the pattern. The node restriction specifies the restrictions that the nodes on the tree should satisfy, such as some specific features. The node chain for the raw pattern should consist of only the node type of the root node of the tree or the basic property of the root node. The node chain of the base pattern consists of a set of chains of nodes such as "(VP-1 S-1)"(the number suffix will be explained shortly).

The additional restrictions of the pattern part of a rule consists of a set of restrictions that the corresponding trees should satisfy. Each additional restriction specifies the constraint that is related to two nodes that are in the different patterns of the rule. As an example the main-verb2 rule is shown in Figure 4.6.

```
(main-verb2  ()
  [ ( < (np-1 s-1) 1 ((link subj)) > ; base pattern(indicated by 1)
      [ (v-1) 0 (main (not aux)) ] ) ; raw pattern(indicated by 0)
       ; additional restrictions
    ( [equal (feature-of num np-1) (feature-of num v-1)]
    ) ]
  [ (create vp-1)          ; action 1
    (attach v-1 to  vp-1) ; action 2
    (transfer-features v-1 to vp-1)
    (attach vp-1 to s-1)
    (sem np-1 v-1 subj)
    (if (have-feature intransitive in v-1)
        (add-feature struc-ok to s-1))
    (connect_main_sec2 v-1) ]
)
```

Figure 4.6 Main-verb2 Rule

This rule matches CLIST when the subject NP of a clause is followed by a verb which is in the form of a main verb. This NP should already be attached to the S node and this is checked in the base pattern of the rule (see the node chain "(np-1 s-1)"). The first raw pattern checks to see if the node of the rightmost tree is a verb which is not an auxiliary verb. The additional restriction checks the agreement of the number feature of the NP

and the verb. This rule actually interprets the NP as the subject of the sentence whose main verb is the verb corresponding to v-1. In the action part of the rule, a node of type VP is created on CLIST and the v-1 node is attached to the VP node.

The action part of a rule consists of the list of actions. An action is a command to the rule interpreter about manipulating the content of the SWM, setting or resetting the global variables, sending a request to the semantic processing component, etc. Some actions are shown in Table 4.1.

Table 4.1. Some Actions

```
(create vp-1)   ; create and put a new node in Clist.
(attach v-1 to  vp-1)   ; attach a tree to a node.
(transfer-features v-1 to vp-1)   ; copy the features.
(if (have-feature passive in aux-1) ...)   ;test a feature in a node.
(add-feature dummy to np-2)   ; add a feature to a node.
(sem connect np-1 v-1 (dative-prep v-1) 2)   ; semantic suggestion to SEM.
(push-to-cash-lastnode)   ; move the rightmost node in Clist to Cash.
(change-link np-1 to iobj)   ; change a name of a link.
(remove-feature modifiable from np-1)   ; delete a feature from a node.
(set-whcomp np-2)   ; set a wh-phrase pointer to an NP.
(make-pointer np-3 to np-1)   ; make a node point to another node.
```

Actions use the node names such as NP1 or VP1 identified during the pattern matching process of the pattern part of the rule. Because more than one node of the same category can appear in the pattern part, a number suffix is used for identification (example, NP-1, NP-2, etc.). For example, the pointer corresponding to NP-1 found during the pattern matching is used during the execution of actions when NP-1 is specified in an action of the same rule. Some complexity arises because more than one rule can match during the pattern matching and the actions of those rules can change the SWM in a different way. Therefore the data structures of the branch are copied into each new branch created for each rule found during the pattern matching. Then the actions of a rule are applied on the SWM of the rule's branch.

Now, it is necessary to explain how each rule's pattern part is matched against the trees in CLIST. Note that one pattern is matched against one tree. The mapping of the patterns to the trees is illustrated in Figure 4.7.

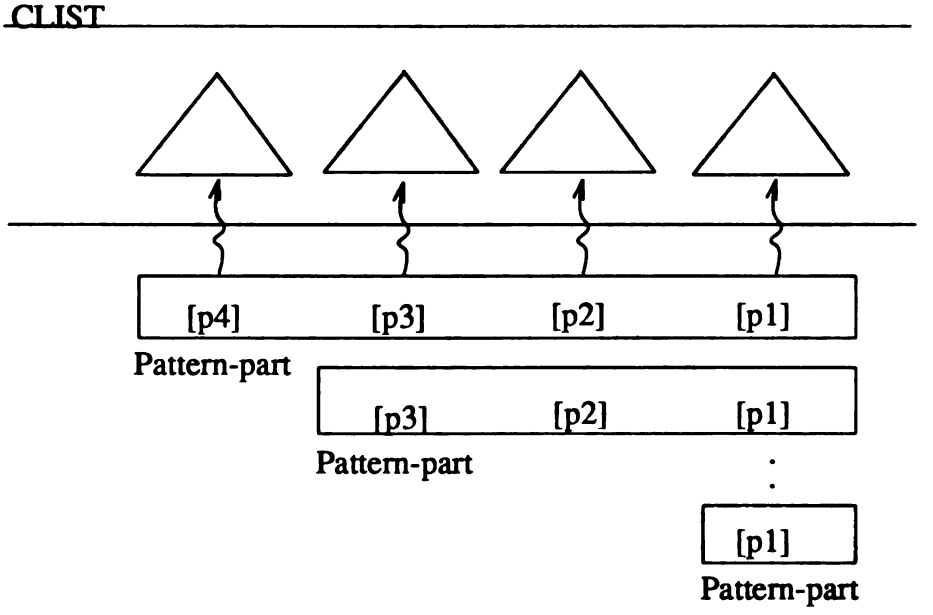


Figure 4.7 Mapping Patterns to Trees

The rightmost pattern should be compared with the rightmost tree and the next rightmost pattern with the next rightmost tree, and so forth. This mapping implies that a rule always specifies the righthand part of CLIST. The rightmost tree is the starting point of the matching process. This pattern matching strategy is different from that of Parsifal. In Parsifal, the first buffer cell is the starting point as illustrated in Figure 4.8.

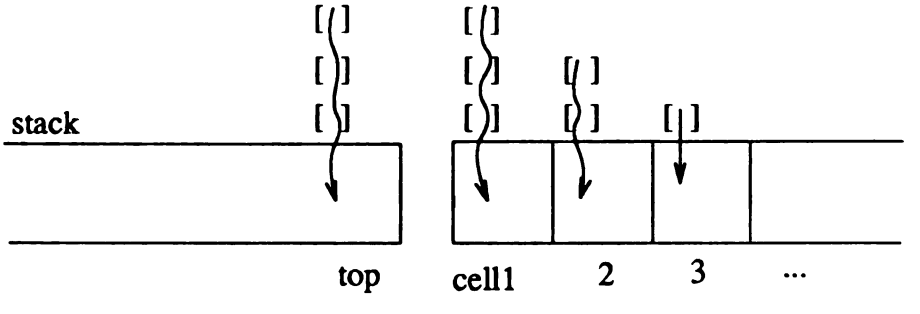
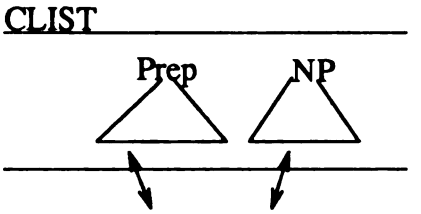


Figure 4.8 Parsifal's Pattern Matching

4.3 Delay Used for Increasing Syntactic Context

Because of the pattern matching scheme explained in the above section, it is possible for SYNSEM to delay the construction of structures. The lookahead capability in Parsifal can be achieved in SYNSEM by delaying the syntactic decision and receiving further material into CLIST until a sufficient amount of context is collected in CLIST. Figure 4.9 is for illustrating how delaying can be used for the construction of a PP. Consider sentence fragment (4-1) along with the figure.



Rule Build-PP: [Prep][NP] => ([create new PP]
[attach Prep to PP]
[attach NP to PP])

Rule PP-create in Packet Cpool
[Prep][NP]=>([....]
[.....]
[.....])

Figure 4.9 Delay in the build-PP Rule

(4-1) .... 1 in 2 the 3 basket 4 ....

In the case of Parsifal, the PP for "in the basket" is constructed while the processing point remains at position 1 by looking ahead to the preposition and the NP "the basket". To prepare the NP "the basket" which the normal processing point has not yet reached, the attention shifting mechanism is used. While the normal processing point remains at position 1, Parsifal moves its attention to position 2 and it processes the input starting from this position. After completing the construction of the NP, the parser returns to position 1. At this time, the input looks like "in [NP]". Now the PP-creation rule matches this input. The working of Parsifal can be understood in a way that the parsing process forks a child process and the child process analyzes and prepares an NP that lies ahead

(this is the attention shifting mechanism).

But SYNSEM does not use the complex processing mechanism such as the attention shifting mechanism. When the parser's processing point is at 2 in (4-1), the parser gets to the blocked situation because no rule matches CLIST in which the preposition "in" is the rightmost tree and no part of the NP has yet appeared. This is not the dead end situation but the read-again situation. This situation is shown in Figure 4.10(a).

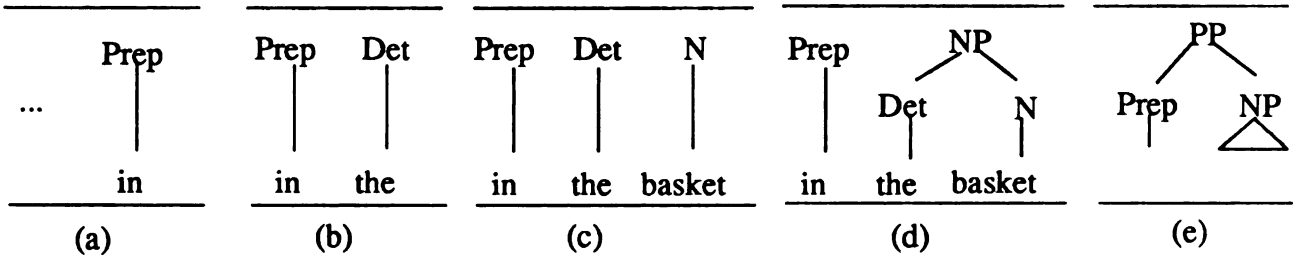


Figure 4.10 Steps of Building a PP

Thus SYNSEM reads the next word "the" to provide more context, and the result is shown in (b) in the figure. But CLIST in (b) still does not have enough context to find a rule that matches it. Thus, the next word, "basket", is input and CLIST becomes (c). Note that the processing point is at position 4 instead of staying at position 1. Against CLIST in (c), the NP construction rule matches and it builds the NP as shown in Figure 4.10(d). (This account is a simplied version. In the actual grammar of SYNSEM, it is more complicated than this. Actually several rules are used to build the NP.) Still the processing point (i.e. the rightmost node) is at position 4. Now the rule build-pp is found during the pattern matching and creates the PP as shown in (e). Note that the PP has been built using only the uniform processing mechanism. SYNSEM did not use some special mechanism such as the attention shifting mechanism. Instead of looking ahead and shifting the attention, SYNSEM delayed the decision of creating the PP (i.e. applying the build-pp rule) until CLIST gets the sufficient amount of context. "Delaying the decision" is implicitly implemented in "reading the next word if no rule matches CLIST".

A complex NP such as that shown in (4-2) gives more of a problem to Parsifal.

(4-2) ... 1 in 2 the 3 basket 4 which 5 the man 6 brought 7 from 8 the 9 store....

In Parsifal, while the main processing point remains at position 1, the sub-processing begins at 2 to prepare the NP headed by "the basket". But the relative clause is a part of this NP and it contains several NP's such as "the man" and "the store". Thus, inside of the sub-processing (i.e. the attention shifting), another sub-processing should be embedded. This results in the recursive embedding of the processing levels as shown in Figure 4.11 because another attention shifting should be done inside an attention shifting.

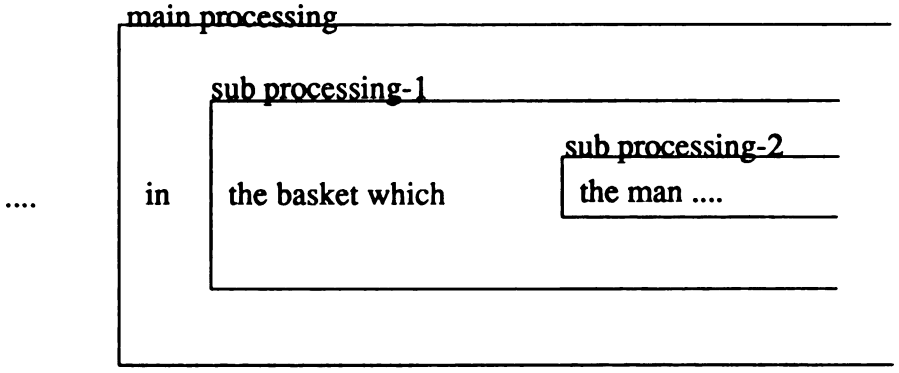


Figure 4.11 Recursive Embedding of Attention Shifting

This results in undesirable complexity in the processing. It seems intuitive that this is not the processing strategy that is used in the human language processing system. To avoid this problem, the "node reactivation" mechanism is used in the grammar of Parsifal. As soon as "the basket" is analyzed, the parser returns to the main processing point. Thus when the relative clause is being analyzed in (4-2), there is no embedded attention shifting. As soon as "which" is seen, Parsifal uses the node reactivation rule(similar to the attention shifting rule) to resume the processing of the NP "the basket". Note that the NP may have already been attached to a previous tree at this point. This technique solves the problem of deep recursion of the attention shifting, but it is directly against the "limited looking ahead" theory in Parsifal. The "lookahead" theory of parsing can achieve its full power only when the whole NP can be prepared by one attention shifting. (4-3) can be

used to illustrate this point:

(4-3) It is natural for the man who stole the apple to be punished.

After "the man" is analyzed as an NP, the buffer would be like:

[Prep;for] [NP; the man] [who]

The "PP-create" rule in Parsifal will match against the buffer cells shown above. But the "For-np-to" rule in Parsifal is the rule that actually needs to match instead of the PP-create rule.

SYNSEM parses sentence (4-2) using the uniform processing strategy. As soon as "the basket" is analyzed as an NP, the rule build-pp matches CLIST. As soon as the head noun of the NP is found, the partial NP can be regarded as a complete NP. In other words, the partial NP corresponding to "the basket" is treated as a legitimate NP and can be used by other rules. This is an early attachment strategy. This idea is similar to the node reactivation rule in Parsifal but the account in SYNSEM is more easily conceived and implemented. When SYNSEM reads "which", the rule which initiates a relative clause is found and it attaches the secondary clause node "S" to the NP, "the basket" (here the NP has already been attached to the PP node). Let's call this rule wh-relative. This rule is shown in Figure 4.12.

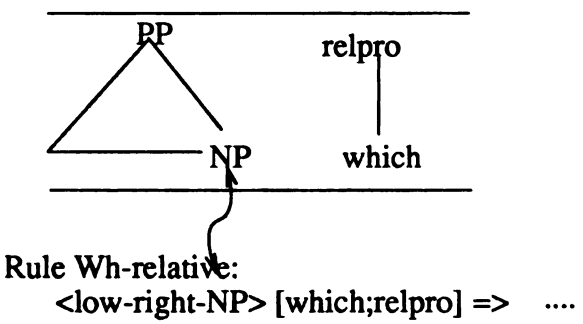


Figure 4.12 Matching Lowest and Rightmost NP

In the rule wh-relative, the base pattern is <low-right-np> (low-right-np is a special case of node-chain), which indicates that the lowest and rightmost node of the corresponding tree (i.e. the second rightmost tree in CLIST) should be an NP. After the execution of

this rule, the snapshot of CLIST will be Figure 4.13.

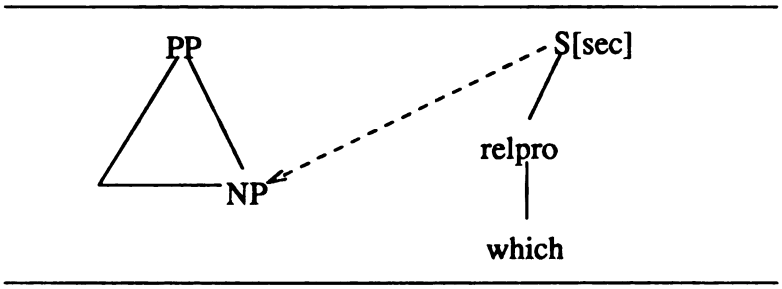


Figure 4.13 Implicit Attachment of Secondary Clause

The secondary clause node, "S", is implicitly attached to the NP which is attached to the PP node. When the NP for "the man" is built, it will be attached to the S node as a subject. As the following input is received, the secondary clause node, S, is built incrementally as if it is the main clause. The secondary S node is implicitly attached to be seen as the root of a tree (indicated by the dotted line) so that the rules for the main clause processing can be used for the processing of the secondary clause. When the secondary clause S is complete, this tree will just be popped out (but it will still be linked to the NP node). As will be shown in the examples later, the PP can be attached to the previous tree before the processing of its relative clause begins.

But SYNSEM's processing strategy is not optimal. For the sentence (4-3), SYNSEM can not use a rule whose pattern part is of the form "[Prep;for] [NP] [Prep;to]" (let us call this the For-np-to rule) because of the following reason: as soon as the NP for "the man" is constructed the build-pp rule will match the SWM and create a PP. One solution is to add some restriction to the build-pp rule so that it can not match if the preposition is "for". Another solution is just to add another rule whose pattern part is "[PP;prep=for][Prep;to]" that corresponds to the for-NP-to rule. Neither solution is the best.

4.4 Rules and State of Parsing

In Parsifal, the grammar rules are partitioned into packets. Each rule is a member of only one packet. SYNSEM has no packets. It has only one pool of all rules. This design strategy requires some explanation. The reason that Parsifal use packets is two-fold. The first is efficiency and the second is conciseness of the grammar. Each node in the stack of Parsifal is associated with a list of packets. The node at the top of the stack (the most recently pushed node) is called the current active node. The packets associated with the current active node are the active packets. Adding(deleting) a packet to(from) the set of active packets is requested in the action part of a rule.

When a node is pushed into the top of the stack, the packets related to the previous active node are not active any more. If a node is popped from the top of the stack, the second node from the top of the stack becomes the current active node and the packets that are associated with this node become active. The whole idea of having the active packets for the active node is that only a small subset of the rule base is related to the parsing at each point (represented by the current active node). The other rules are irrelevant to the parsing and need not be accessed. For example, a small number of rules are related to the parsing of the subject of a clause. These rules are contained in the packet called PARSE-SUBJ. The efficiency of parsing is obtained by looking at only the rules in the packet PARSE-SUBJ during the time when the subject is analyzed. Now it is likely that the set of active packets represents the state of parsing (see (Marcus, 1980) for more detailed account). The rules in a packet will be formulated under the assumption that they will be applied only during the state of parsing represented by the packet. Thus the rules need not test the state of parsing in their pattern part, which makes the grammar rules concise.

In SYNSEM, the advantage of using packets can not be used because of its parsing strategy that "any rules that match CLIST should be found." Consider the next sentence to see the reason:

(4-4) The granite rocks near the shore.

Note that "rocks" can be both a noun and a verb. After "rocks" is input, two rules should be found during the pattern matching: a rule that analyze "rocks" as the head noun of the NP and another rule that analyze "rocks" as the main verb (actually there is one more rule). But these two rules can not be put in the same packet because they are related to different parsing stages, which makes SYNSEM unable to use packets. One of the disadvantages of using packets is that there should be some rules(or actions) that add or delete packets to and from the set of active packets. This causes some difficulty to the grammar writer. The management of the active packets gives difficulty to both the designer and the reader of the grammar. Charniak(1983b) attempted to remove this difficulty without much success. This difficulty can be totally eliminated by removing the packets. Our experience is that it is hard to keep track of the active packets during the reading of Parsifal's grammar rules.

Instead of using the active packets to get the state information of parsing, SYNSEM uses rules with the base pattern to check the state of parsing. It has been found that the base pattern can be specified in a simple way. There are many rules that do not even need the base pattern. It is argued that rules are still concise even with the base pattern. We need to consider the possible inefficiency caused by not adopting the packets. During the pattern matching, all rules in the rule base should be checked to find the rules that match CLIST. But this computational cost can be minimized by using a special rule indexing technique. It has been verified in the experiment that the time taken for the pattern matching is a small fraction of the total parsing time. The bottleneck of the speed of SYNSEM is in the semantic processing instead of syntactic processing.

The raw patterns in the rules of SYNSEM are analogous to the patterns for checking the buffer cells in Parsifal. Marcus argued that the lookahead of three buffer cells is enough for deterministic parsing. He used this lookahead of only up to three buffer cells to explain why the garden-path effect occurs in sentence (4-5). But this argument has

been challenged in several ways. Refer to (Kurtzman, 1985).

(4-5)The horse raced past the barn fell.

Following Parsifal, SYNSEM uses up to three raw patterns to make a rule. It has been found that three raw patterns are enough to write the grammar.

Marcus explained that some linguistic generalizations such as Ross's complex NP constraint can be obtained automatically by using Parsifal's grammatical constraint that a rule can not access the inside of the nodes in the active node stack other than the current active node and the first S node. In SYNSEM, such grammatical constraint does not exist. The base pattern should be able to access the inside of the corresponding tree. Hirst(1984:211) provides the following counter example to Marcus' claim. In the prepositional attachment problem, the verb of the VP should be examined to determine the attachment point (either the VP or the NP appearing after the main verb). The NP is the current active node (i.e. the node at the bottom of the stack) and thus it can be examined during the processing. But the VP is the node which is the node just above the NP in the stack and thus it can not be looked at according to Parsifal's constraint.

Another counter example can be found in the parsing of conjunctions. Consider the following sentences with conjunctions in them:

(4-6)The principal ordered the pupil to be punished and scolded.

(4-7)The teacher ordered the student to be punished and taught the other students not to commit a similar misdemeanor.

To determine the counterpart of the conjunction of "scolded" and "taught" in the above sentences, the verb "ordered" should be considered. But according to the constraint of Parsifal, the verb "ordered" should not be available for consideration because it is neither the current active node nor the first S node above the current active node.

4.5 Rules and Ambiguity

In SYNSEM, the existence of a local syntactic ambiguity at a certain point is exhibited by the fact that multiple rules are found during the pattern matching. One of the important guidelines to writing the grammar rules in SYNSEM is to prepare a rule for each possible analysis of the ambiguity. Multiple rules found by the pattern matcher invoke the same number of parsing branches. As the parsing goes on, some branches will die out because of either syntactic or semantic reasons. Later there will remain only one branch, which means that the ambiguity has been resolved. In sentence (4-8),

(4-8) The boy ate a cake with a fork.

there is an ambiguity about the attachment of the PP "with a fork" as shown in Figure 4.14.

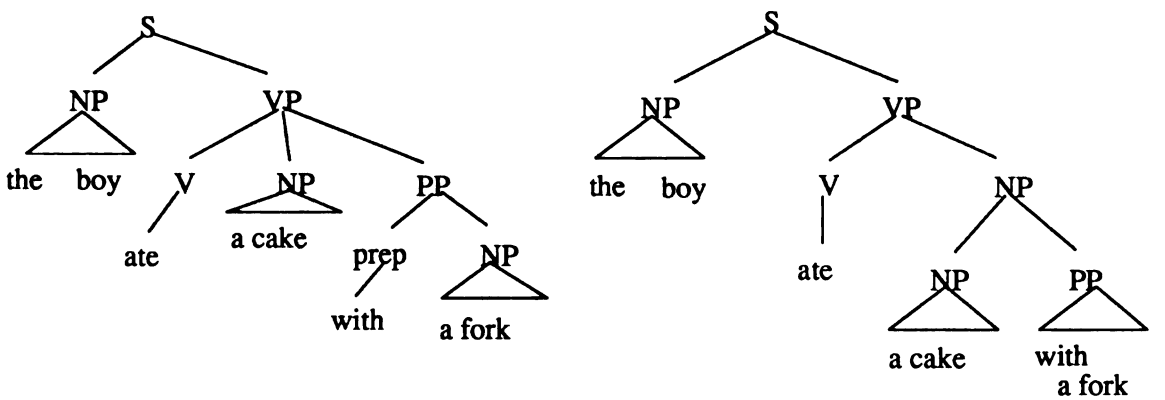


Figure 4.14 PP Attachment Ambiguity

The snapshot of CLIST just after the analysis of the PP "with a fork" is shown in Figure 4.15. There are two rules(np-pp and vp-pp) that match this CLIST (these two rules are also shown in Figure 4.15):

Note that the NP "a cake" (possibly a partial NP) has been already attached to the VP because the head noun has been already analyzed and "a cake" is considered to be a legitimate NP. As explained before, the base pattern <low-right-np> of the rule np-pp matches the NP "a cake" because this NP is the lowest and rightmost node of the second

```
(np-pp  ((l )(h passive-by))
  [ ( [ (np-1 low-right) 1 (modifiable) ]; 1 indicates base pattern.
      [ (pp-1) 0 () ] )                  ; 0 indicates raw pattern.
    () ]  ; no additional restrictions.
  [ (attach pp-1 to np-1)
    (if (have-feature trace in pp-1)
        (sem connect (np pp-1) np-1 (prep pp-1) 2)
        (sem connect np-1 (np pp-1) (prep pp-1)) )
    (remove-feature modifiable from np-1) ]
)

(vp-pp  ((l )(h wh-dative-np-pp))
  [ ( [ (vp-1 s-1) 1 () ] ; base pattern.
      [ (pp-1) 0 () ] )   ; raw pattern.
    () ]
  [ (attach pp-1 to vp-1)
    (if (have-feature trace in pp-1)
        (sem connect (np pp-1) (v vp-1) (prep pp-1) 2)
        (sem connect (v vp-1) (np pp-1) (prep pp-1))  )
  ]
)
```

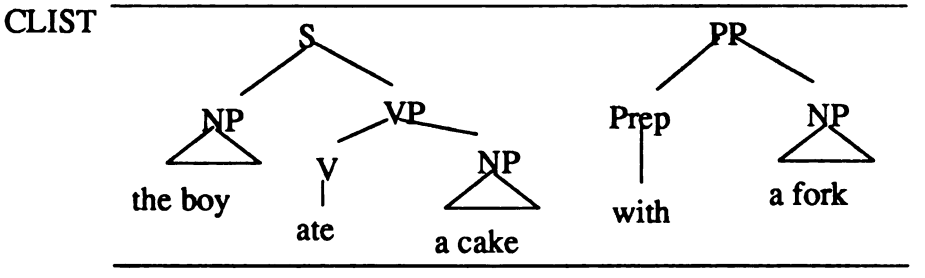


Figure 4.15 PP Attachment and the Corresponding Rules

rightmost tree in CLIST. The base pattern of the rule vp-pp checks if the second rightmost tree of CLIST has the root node of type S and its rightmost child node is of type VP (i.e. the verb of the clause has been analyzed). This is to check the state of parsing if there is a VP which can be modified by the PP that is the rightmost tree in CLIST. For each of np-pp and vp-pp, a branch is created. The actions in the rules are executed and some of them may invoke semantic processing. The result of semantic processing of the rules is used for comparing the branches and the better one is chosen (which will be vp-pp in this case).

The sentence (4-9) shows another kind of ambiguity. After the word "raced" has been input, the SWM becomes that shown in Figure 4.16. Two rules should be found during the pattern matching.

(4-9)The horse raced past the barn fell.

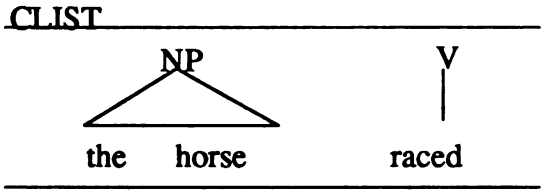


Figure 4.16 Verb Form Ambiguity

The rule main-verb2 analyzes "raced" as the past main verb of the sentence and the rule np-vpp considers "raced" as the past participle leading a reduced relative clause.

Ambiguity also occurs when a word is categorically ambiguous as shown in (4-10).

(4-10)The granite rocks near the shore. (= (4-4))

The word, "rocks", can be both a noun and a verb. The snapshot of CLIST and the rules that match this CLIST are shown in Figure 4.17:

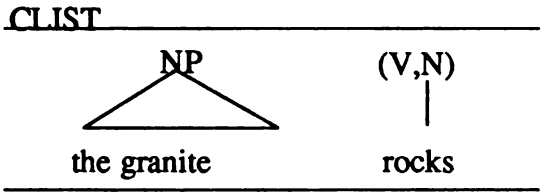


Figure 4.17 CLIST and Categorical Ambiguity

Three rules are found in this example: the noun-parse3 rule which interprets "rocks" as a main verb, the noun-parse4 rule which considers "rocks" as the head noun of the NP and the noun-parse5 rule which closes the NP "granite" and considers "rocks" as a noun that begins a new NP. The disambiguation will be done later by either semantic information or syntactic reasons. This example will be explained in more detail in Chapter 6. These examples show that SYNSEM prepares one rule for each alternative of an ambiguity.

4.5.1 Delay and Parallelism

The fact that a rule can have up to three raw patterns indicates that the parser can delay the decision up to three constituents (i.e. at least three words) to collect the sufficient context. But it seems that the human sentence processing system does not always utilize the delay as fully as possible. Let's consider rules for parsing (4-11) and (4-12).

(4-11)I know the man.

(4-12)I know the man is a doctor.

After "the man" is analyzed as the NP, the rule **object** in (4-13) is used to analyze this NP as the object of the verb "know". This rule is shown below:

```
(4-13) Rule Object:
          <S-VP-V> [NP] => [attach NP to VP] ...

(4-14) Rule Reduced-that-comp
          <S-VP-V; that-comp> [NP] [V] =>...
```

Note that V is specified in the node chain of the base pattern to make sure that there is nothing attached to the VP except the verb. The problem of the **object** rule is that it will analyze "the man" as the object of "know" in (4-12), too. This is the incorrect analysis. In (4-12), the **reduced-that-comp** rule in (4-14) should be used to analyze the complement clause that is led by the NP, "the man". However this rule can match CLIST after a verb(or auxiliary verb) is input to CLIST because of the pattern `[v]` in the rule. Before this rule matches, the **object** rule will match CLIST. Then the parser will reach the dead-end and the parsing will fail.

The solution for the above problem is to change the **object** rule so that it can not be applied for the verbs that can have a "that" complement clause and to add another object rule (called **special-object** shown below) that will wait and see the word after the NP. The updated rules for this purpose are shown in (4-15) and (4-16):

```
(4-15) Rule Object:
          <S-VP-V; not(that-comp)> [NP]
                              => [attach NP to VP] ...

(4-16) Rule Special-object:
          <S-VP-V; that-comp> [NP] [≠V]
                              => [attach NP to VP] ...
```

The **object** rule in (4-15) can not be used for (4-11) or (4-12) because the V, "know", has the feature "that-comp" indicating that it can have a complement clause with complementizer "that" (note that the "that" complementizer in English can be omitted). For (4-11), the **special-object** rule in (4-16) will be used after "." is input, because "." will match the pattern [≠V]. Thus both (4-11) and (4-12) can be successfully analyzed using these two rules.

But these rules are not perfect. The sentence (4-17) causes a problem.

(4-17)I know the man who eats an apple is a doctor.

Just after "who" is input into CLIST, the **special-object** rule matches the CLIST and "the man" is analyzed as the object of the verb "know". But this is a wrong analysis because "the man" should be analyzed as the subject of the complement clause. This decision should be delayed until the correct evidence occurs. To fix this problem, the **special-object** rule can be modified as follows:

```
Rule special-object:
     <S-VP-V; that-comp> [NP] [.] => ....
```

This solution works for the sentence (4-17), but there is a counter example such as (4-18):

(4-18)I have known the man since 1930.

The **special-object** rule can not be used to analyze this sentence because "since 1930" exists between "the man" and ".".

From the discussion done so far, we can notice that even delaying the decision for three words is not enough. It is argued in Frazier and Raynor(1982) that "the man" in (4-12) is first analyzed as the direct object of the verb and then this analysis is thrown away if it proves to be wrong later and the sentence is reanalyzed. They presented data from eye movement experiments as evidence that (4-12) shows some garden path effect. The position taken in SYNSEM is to use the following two rules:

```
(4-19) Rule Object:
          <S-VP-V> [NP] => [attach NP to VP] ...

(4-20) Rule reduced-that-comp:
          <S-VP-V; that-comp> [NP] => ..open sec clause..
```

The above two rules match CLIST (just after "the man" is analyzed as the NP) and the corresponding two branches run in parallel. For the sentence (4-11), the branch created by the rule reduced-that-comp will be removed after the input of "." because the expected complement clause does not follow. The sentence is successfully analyzed by another branch created by the rule object. In the case of (4-12), the input "is" will make the parser remove the branch corresponding to the object rule. The branch issued by the rule reduced-that-comp will analyze this sentence successfully. For the sentence (4-17), the branch issued by the reduced-that-comp will carry out the parsing but the branch for object will die in the middle of parsing. This parsing strategy indicates that the *partial parallelism* concept introduced in Chapter 3 overcomes this problem. But note that this is just an alternative to the reanalysis strategy.

4.6 Syntactic Processing in Operation

To understand the operation of the syntactic analysis clearly, a detailed trace for the analysis of (4-21) will be given in this section (See Appendix D for the system output and see Appendix E for the rules mentioned in the thesis).

(4-21)The teachers taught by the pool passed the test.

The parsing of a sentence starts with the empty CLIST and CASH. Just before reading the first word, a dummy node of type "SS" is inserted into the empty CLIST (the snapshot of the SWM is shown in Figure 4.18). "SS" node indicates

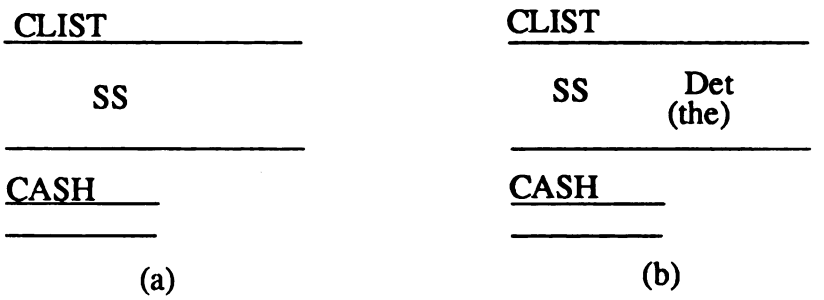


Figure 4.18 Snapshots of SWM

that it is the starting point of a sentence. MON reads the word "the" and calls the morphological routine. This routine looks in the dictionary and finds the part of speech of the word. If the word is an open class word, an instance concept for it is created in the knowledge base and the MP passes a marker starting from this instance concept. Marker passing will be explained in the next chapter. This word is inserted into CLIST of the branches that are active (only one branch here). The SWM becomes as in Figure 4.18(b). For this CLIST, the pattern matcher finds only one rule parse-det (Refer to Appendix A for the NP structure used in SYNSEM.) After executing this rule, The SWM becomes (c). No rules are found for this SWM, which means that the blocked situation has occurred. However it is determined that this is not the dead end (Refer to Appendix B.) Thus MON reads the next word "teachers" and inserts it into CLIST as shown in (d). The pattern matcher finds only one rule, noun-parse1, for this SWM and its execution results in (e). For (e), the pattern matcher finds the rule noun-parse33 and the SWM becomes (f). This rule analyzes the noun "teachers" as the head noun of the NP that is being built, because the noun is in plural form. Refer to Milne(1980) for the use of singular/plural form of a noun to decide the boundary of the NP.

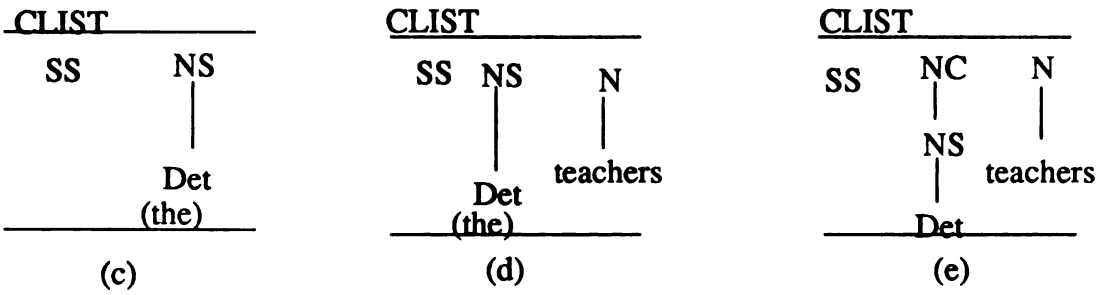


Figure 4.18 Snapshots of SWM(Continued)

Only one rule, subject, matches (f) which changes CLIST to (g).

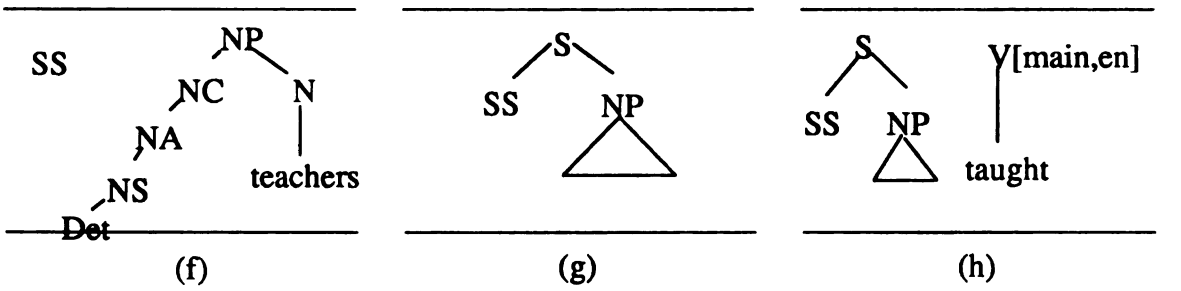


Figure 4.18 Snapshots of SWM(continued)

The NP has been analyzed as the subject of the sentence. No rule matches (g) and the next word "taught" is read in by MON. Because this is an open word, the MP is called to pass markers from the instance concept for this word. The Snode for this word has features "main" and "en" because "taught" can be a main verb and a past participle. For CLIST in (h), the pattern matcher finds three rules: main-verb2, np-vpp and np-vpp1. The main-verb2 rule analyzes "taught" as the main verb of the sentence. The other two rules are for the reduced relative clause analysis of "taught". Note that this verb is a dative verb as shown in "She taught the boy mathematics." Thus "the teachers" in (4-21) can be either a direct or indirect object of "taught". For example, "the theory" in (4-22) is the direct object but "the person" in (4-23) is the indirect object of the verb "taught".

(4-22)The theory taught by the professor is really hard to understand.

(4-23)The person taught by the professor is a farmer.

The np-vpp rule analyzes the NP, "the teachers", as the direct object and the np-vpp1 rule analyzes the NP as the indirect object of the verb "taught". The pattern parts of the two rules are almost the same. The difference comes from the action part. The action part of np-vpp has an action which suggests that the NP and the V should be related via the pseudo-preposition "obj". This indicates that the NP is the direct object of the V. But an action in the action part of the rule np-vpp1 suggests that the NP and the V should be related via the preposition "to" (which is the dative preposition of "teach"). This means that the NP is the indirect object of the V.

Each of the above three rules create a branch as shown in Figure 4.19.

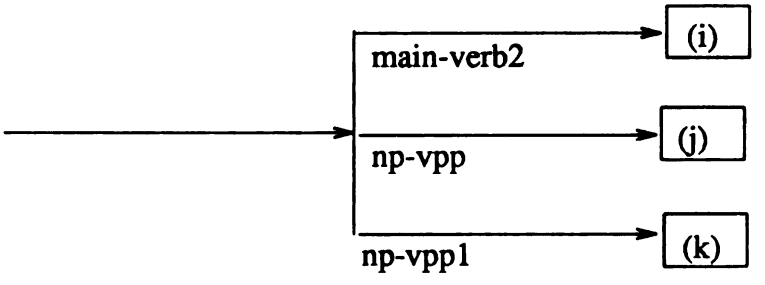


Figure 4.19 Creation of Multiple Branches

The SWM's in (i), (j) and (k) shown in Figure 4.18 are the results of the execution of the rules of the three branches.

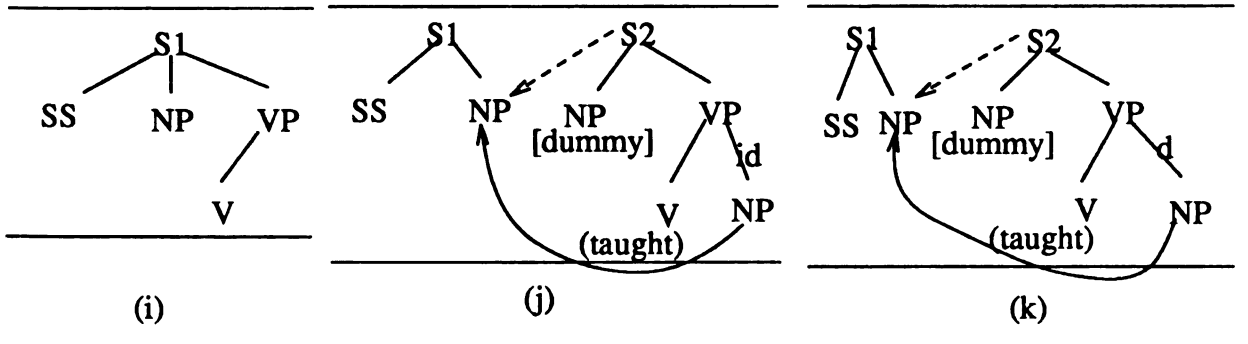


Figure 4.18 Snapshots of SWM(Continued)

In (j) and (k), the secondary clause S2 is attached to the NP. But S2 is a root node of the rightmost tree because this is an implicit attachment as indicated with the dotted line. The reason that S2 should be a root node is that this secondary clause is treated as if it is

the main clause so that the rules for building the main clause can be used to build the secondary clause. When S2 is complete, it will just be popped out from CLIST (but it will still be attached to the NP). An attachment like this is called an implicit attachment because it is done in the background. S2 actually hides S1 until its construction is finished and it is popped out.

For each branch, no rule is found during the pattern matching and a new word is requested to be input. As explained in Chapter 3, the semantic comparison stage comes at this point. The branch for the rule main-verb2 is selected after the comparison of the semantic processing results (The reason for this selection will be explained in Chapter 6, but it has to do with the knowledge that teachers do teach). The other two branches are pushed into the backup stack. Out of these two branches, the one with the better semantic result is pushed later so that it can be used first when backtracking is needed later. The output of this semantic comparison stage is one branch. Now MON inputs the new word, "by", and the SWM for the selected branch becomes (l).

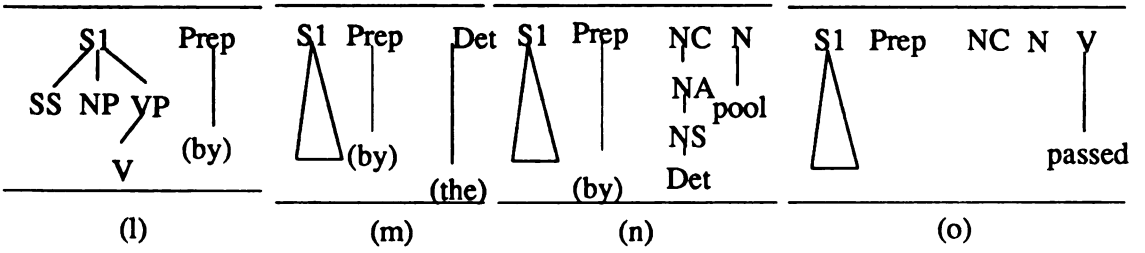


Figure 4.18 Snapshots of SWM(Continued)

No rule matches (l) and MON reads the next word "the" as shown in (m). After several NP processing rules are used and the word "pool" is input, the SWM becomes (n) (this change is same as the change from (b) to (e) before). At this point, it can not be determined that "pool" is the head noun of the NP without looking at the next words. For example, "pool" is the head noun of the NP in "the pool of the house", but it is not the head noun in the nominal compound, "the pool repair". No rule matches the SWM in (n). Thus MON reads the next word "passed" and the SWM becomes (o). The rule

noun-parse3 matches (o). Seeing the V following the noun "pool", this rule decides that "pool" is the head noun of the NP being built. Note the last action of this rule is "(push-to-cash-last-node)" which pops the last node, V(passed), and puts it into CASH. The reason for performing this action is that the V has been used by the rule only for getting enough context to make the decision if "pool" is the head noun or not. After completing this purpose, the V should be retracted from CLIST so that the NP just built can be the rightmost node of CLIST and all further processing can be done for this situation.

After the execution of noun-parse3, the SWM becomes (p). For this new SWM, the pattern matcher finds the rule build-pp which builds a PP as shown in (q).

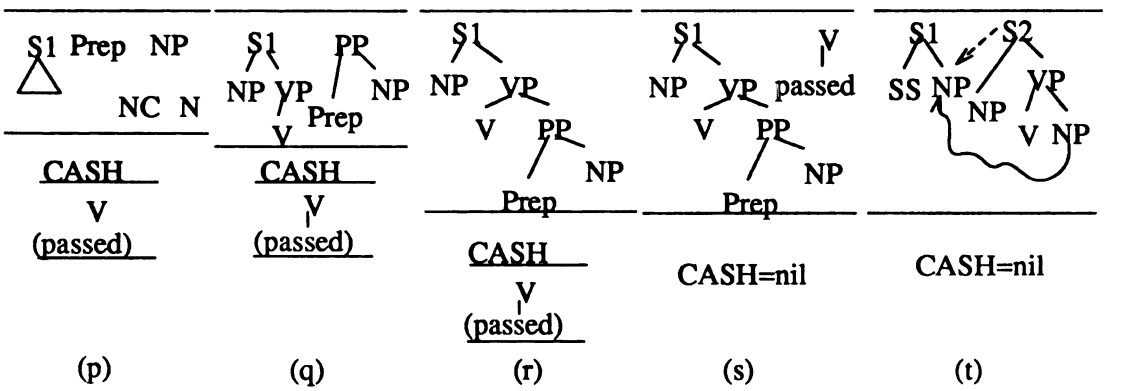


Figure 4.18 Snapshots of SWM(Continued)

The vp-pp rule matches the SWM in (q) and this rule attaches the PP to the VP as shown in (r). (The semantic component returns the signal of accepting this attachment because a "teaching" action can happen at the location close to the pool.) No rule is found for (r) and a new input is required for CLIST. Because CASH is not empty, MON pushes the node, V(passed), in CASH into CLIST instead of reading a new word from the input string. The new SWM is shown in (s). The V of "passed" can be a past main verb or a past participle. The main verb analysis rule does not match the SWM because the main verb is already in the tree. The only rule that matches is np-vpp which analyzes the V(passed) as the past participle that begins the reduced relative clause.

Let's assume that "pass" has two meanings: (1) to hand something to a nearby person, (2) to have enough qualification in a test. The execution of the np-vpp rule gets the response from the semantic component that the result is semantically bad. The reason for this response is that a pool can not be passed because it is not a movable object and a pool is not a kind of test. The branch whose execution of the corresponding rule results in a bad semantic effect is just removed. This is another case of disambiguation due to the semantic reason.

Note that there has been only one rule that is active and the execution of this rule has been rejected by the semantic component. Thus, the parsing is in a dead end because there is no active branch left. Therefore it is necessary to back up. The parser pops out an image of a branch that was stored in the backup stack and makes a new branch which is shown in (t). Note that the SWM of this branch is actually the SWM in Figure 4.18(k) that was pushed into the backup stack. This branch is the branch that was created corresponding to the rule np-vpp1. The next word to be read for the branch corresponding to the SWM in (k) is "by".

No rule matches (t) and MON reads the next word "by". The new SWM is shown in (u). The course of parsing from (u) to (v) is not explained because it is related to the PP analysis which was explained from (l) to (q). Against the SWM in (v),

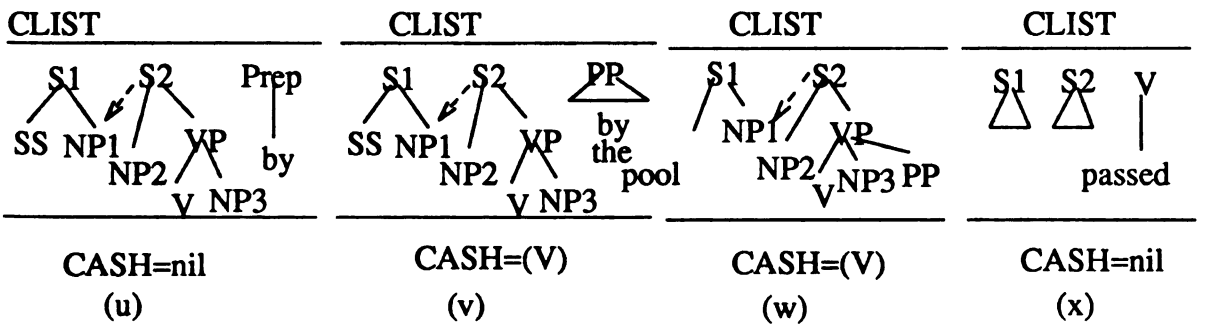


Figure 4.18 Snapshots of SWM(Continued)

the pattern matcher finds two rules: the vp-pp rule and the passive-by rule. passive-by analyzes the NP, "the pool", of the PP as the actual subject of the verb "taught" (of the passive clause) and the vp-pp rule tries to attach the PP, "by the pool" to "taught". Thus,

two branches are created. The execution of the rule passive-by gets the failure signal from the semantic component because a pool can not be the agent of the action "teach". Thus the branch for this rule is removed. The execution of the rule vp-pp is accepted because a satisfactory signal is returned from the semantic component for the execution of the rule. Thus only one branch remains as active. The new SWM for this branch is shown in (w). No rule matches (w) and the node V in CASH is pushed into CLIST as shown in (x).

Only one rule(np-pp) matches (x) which attempts to analyze the V(passed) as the past participle that begins the reduced relative clause that is attached to the NP, "the pool". But this analysis is rejected by the semantic component as explained before. Therefore the parsing is blocked and needs to back up. But before doing back up, the parser always checks CLIST to see if there is an S of the secondary clause and if it is in a complete form. If so, this S is popped out (this is the strategy of closing the secondary clause in SYNSEM). Even if S2 is popped out, it is still attached to NP1. The SWM for this branch becomes as shown in (y). Against (y), the main-verb2 rule matches. This rule

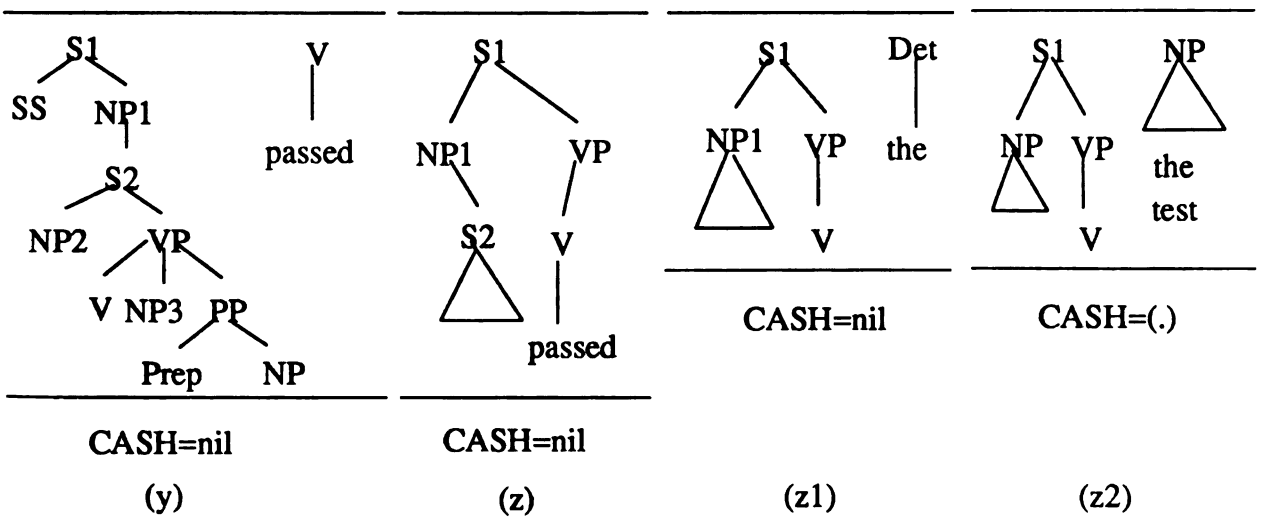


Figure 4.18 Snapshots of SWM(Continued)

analyzes the NP for "the teachers taught by the pool" as the subject of the verb "passed". The result of the execution of this rule is shown in (z). No rule matches the SWM in (z)

and MON inputs the next word "the" as shown in (z1). After the NP for "the test" has been constructed, the SWM becomes (z2). Note that CASH contains "." which was used to provide the context for analyzing the NP.

The object rule matches (z2) and its execution results in the SWM in (z3). No rule matches (z3) and "." in CASH is pushed into CLIST as shown in (z4). The s-close-aff rule matches and this rule

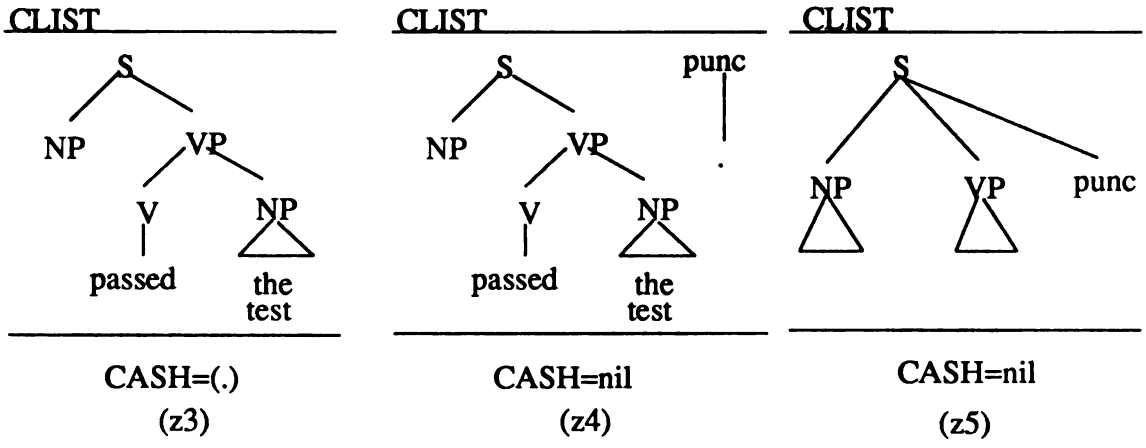


Figure 4.18 Snapshots of SWM(Continued)

attaches the punctuation mark to the tree as shown in (z5). No input is left and the final punctuation mark is attached to the tree. Therefore the parse of the sentence (4-14) is finished.

4.7 Conclusion

The syntactic analysis component of the SYNSEM parser has been explained in this chapter. The data structure employed in the parser is a list containing trees. The grammar rules have the form of fairly straightforward production rules. The main design objective of this component is to allow the parser to be able to delay the syntactic decision until a sufficient amount of context becomes available. This "delaying" is realized by reading the next word when no rule is found during the pattern matching and the situation is not the dead-end. By reading the next input word, the SWM will contain more context and rules that can match this added context are sought again.

The method of delaying the decision achieves the same power as the lookahead used in Parsifal. The use of delaying enables the SYNSEM parser not to use the attention shifting mechanism which psychologically seems to require processing which is too complex. SYNSEM can use a simple and uniform processing technique even for the structures that require attention shifting in Parsifal.

Another feature of SYN is the base pattern that is used to find the state of parsing in each rule if necessary. Instead of employing states in ATN or packets in Parsifal, each rule can test the state of the parsing by specifying the base pattern. This might make the rules look a little complex but actually it makes the rules more readable and makes it easier to follow the grammar.

The rules that have corresponding semantic processing contain actions which issue a suggestion to the semantic processing component to find a path in the knowledge base. The path found for the suggestions are used as the measure of the goodness of the rule(branch).

One important guideline to prepare rules in SYNSEM is to prepare one rule for each alternative analysis of a structural ambiguity. If there are n possible ways of analysis at some processing point, then n rules should be proposed by the pattern matcher. Each alternative will become a branch, which means that multiple branches of parsing can go on in parallel as explained in the previous chapter. Some branches will drop out as the parsing is going on because of either a syntactic reason or a semantic reason.

CHAPTER 5

KNOWLEDGE BASE AND MARKER PASSING

The major motivation of the SYNSEM parser's control strategy is to use the semantic information as much as possible as a resource for disambiguation. We use a knowledge base to provide the parser with this semantic information. Because we are building a general natural language understanding system, the knowledge representation language should be one that can represent general world knowledge instead of one that is suited for a specific domain. The enumeration of the possible candidates is first order logic, KL-One, KRL, Frail, Conceptual Dependency, KODIAK, and so forth. It seems that these languages are the same as far as representational power is concerned, even though each has its own representational characteristics. It is known that the first order logic is equivalent to the frame-based representation languages in the representational power. First order logic has a straightforward inference engine which is the so-called "resolution-based" theorem proving. The problem is that the amount of computation can grow intractably as the size of the knowledge base grows.

The frame-based representation languages add some features to have more control on intractability but end up with reduced capability for making inferences. In SYNSEM, KODIAK is used as the knowledge representation language to build the knowledge base. KODIAK was developed by Wilensky(1987). It has been used in some A.I. systems such as FAUSTUS (the text inference system) (Norvig, 1986) and UC (natural language processing system) (Wilensky & etc, 1986).

One of the motivations behind choosing KODIAK in SYNSEM is that it has a type of concept called *relation* that relates the concepts of objects and actions. Another motivation is that it is a semantic network in which it is appropriate to use the marker

passing method to facilitate the semantic processing. Charniak(1983a) first proposed the use of marker passing in natural language processing systems. He pointed out that marker passing-based systems can provide advantages for problems such as word sense disambiguation, prepositional phrase attachment, and so forth. Norvig used the marker passing mechanism to get plausible inferences in the text understanding system.

In this chapter, KODIAK and marker passing used in SYNSEM will be introduced, which provides the reader the necessary background for the discussions in the following chapters.

5.1 KODIAK

KODIAK will be introduced briefly in this section. (The notations for KODIAK elements used in this dissertation mostly follows the description of KODIAK in (Norvig,1986).) KODIAK has three types of nodes: *absolutes, relations,* and *aspectuals*. The nodes are connected via links. There are eight different types of links. Absolutes represent the objects or entities in the world. Entities here include some concepts that have a meaning by themselves. For example, objects in the world such as "book", "train", "person", "dream", "plan", etc. are represented by absolutes in KODIAK. Non-object entities such as "action", "eat", "run", etc. are also represented by absolute nodes. (The main intention of KODIAK is to derive the meaning of absolutes from that of relations and aspectuals.) A relation in KODIAK connects two absolutes. For each relation, there are two corresponding aspectuals. The aspectuals for a relation can be considered to be the formal parameters of the relation. The absolute which specifies the class whose elements can fill the parameter is connected to the aspectual by a link. Let's consider Figure 5.1 as an example. The double circles are for relations, the single circles for aspectuals, and rectangles for absolutes. The relation `eater-eat` relates the absolute `animal` and the absolute `eat`. The absolute `animal` is connected to `eater-eat` via the aspectual `eater.eater-eat`. (The names of the concepts can become long

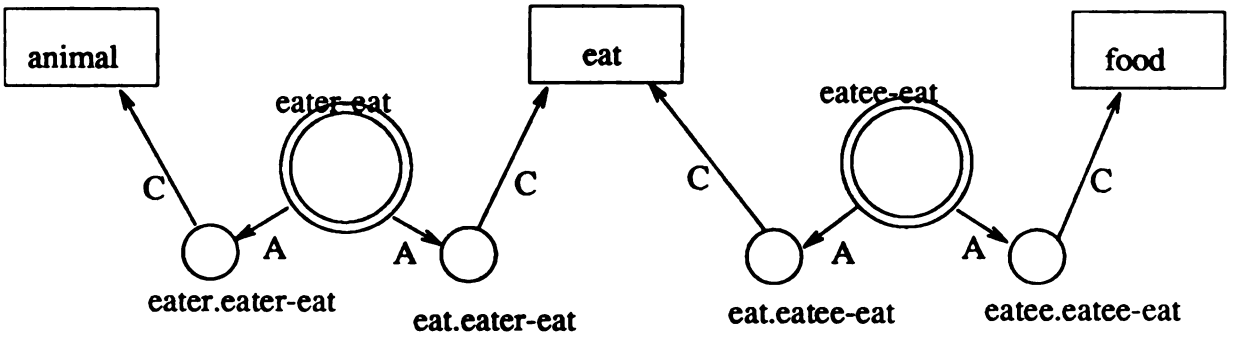


Figure 5.1 Elements of KODIAK

because each concept has a unique name so that a concept can be accessed by its name. We tried to use self-explanatory names but the name itself does not have any meaning for the operation of the system.) We can consider the aspectuals, `eater.eater-eat` and `eat.eater-eat`, as the parameters of the relation `eater-eat`. The absolute `animal` constrains the argument which fills the parameter `eater.eater-eat` and the argument of the parameter `eat.eater-eat` is constrained by the absolute `eat`. As shown in Figure 5.1, the absolute `animal` is pointed by a C(constraint) link from the aspectual `eater.eater-eat`. The "A" link between an aspectual and a relation can be considered to imply *argument*. Anything that is a kind of animal has a relation named `eater-eat` with any concept which is a kind of `eat` and can be connected to (or involved in the relation with) the relation `eater-eat` which is again connected to the concept `eat`. The fragment of knowledge represented in Figure 5.1 can be represented in a frame-based language as shown in Figure 5.2.

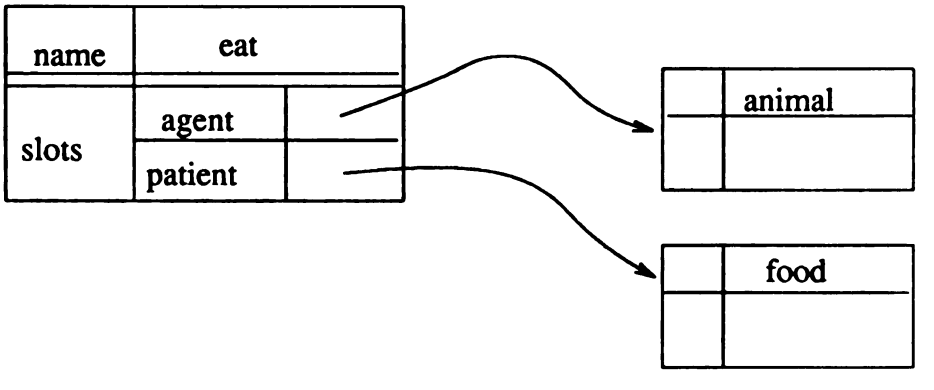


Figure 5.2 Frame Representation

Comparing Figure 5.1 with Figure 5.2, it can be noticed that slots in a frame-based language correspond to the relation nodes in KODIAK. The agent slot of the frame `eat` corresponds to the relation `eater-eat` in KODIAK. Relations are considered to be the most important elements in KODIAK. Thus KODIAK is considered to be a relation-based knowledge representation language. The name "aspectual" is derived from the fact that the meaning of two aspectuals are derived from the relation that holds between them.

KODIAK has eight types of links which are listed in Table 5.1 adapted from Norvig(1986). The Prototype link is the newly added link which points to the concept which can be the prototype filler of an aspectual.

Table 5.1 Links in KODIAK

Dominate(D) - a concept is a subclass of another class

Instance(I) - a concept is an instance of some class

View(V) - a concept can be seen as another class

Constrain(C) - fillers of an aspectual must be of some class

Prototype(P) - prototype filler of an aspectual

Argument(A) - associates aspectuals with a relation

Fill(F) - an aspectual refers to some absolute

Equate(E) - two concepts are co-referential

Differ(Df) - two concepts are not co-referential

The D(dominate) link indicates the subclass relationship between two concepts. If a node, say node1, is a subclass of another node, say node2, then there is a D link which starts from node1 and terminates at node2. D links are also used to specify the subclass relationship between two relations. D links in KODIAK correspond to the IS-A or AKO links in other representation languages. One important implication of D links is that the

properties of concepts are inherited by other concepts via D links. If node1 dominates node2 (i.e. a D link points from node2 to node1), all the relations and aspectuals that node1 is involved with are inherited by node2.

The I(instance) link is used to represent the relationship between a member and the class to which this member belongs. For example, to represent that a specific person called Bill is a person, the I link is used between the absolute `bill01` and `person` as shown Figure 5.3. Note that "01" in `bill01` is used to indicate the specific person, *our* Bill and not, say, Bill Cosby:

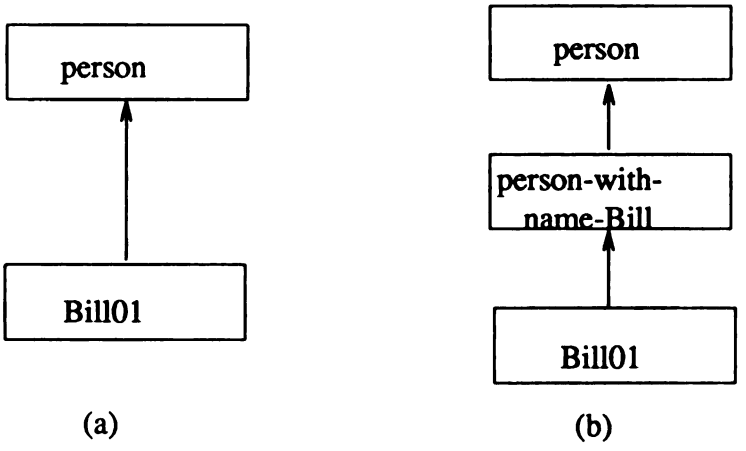


Figure 5.3 Instance Links

Because there can be many persons whose name is Bill, we can introduce a new concept called `persons-with-name-Bill` as shown in (b) of the figure. The A and C links have already been explained. For each relation, there are two A links that associate two aspectuals to the relation. These two aspectuals correspond to the two arguments of the relation. The C link is used to specify what kind of absolute can be connected to an aspectual. This absolute can be considered to be the constraint of a concept which can fill the argument that the aspectual specifies. An E(equate) link is used to connect two absolutes to specify that the absolutes are co-referential. If an E link connects two aspectuals, this means that the aspectuals are filled with the same absolute. Df(different) links are opposites of E links. They are used to assert that two absolutes are explicitly not co-

referential, or that two aspectuals can not be filled with the same absolute.

The V(view) link is used to regard an absolute as another absolute. For example, "person" can be viewed as a "physical-object". In this case, "person" is used actually to refer to the person's body. The representation involving the V link is complex and will not be explained in detail here.

5.1.1 Modeling Knowledge

To have a better feeling of KODIAK, it is useful to look at a piece of a knowledge base written in KODIAK. Figure 5.4 models the knowledge involved with "giving" and "having". Two absolutes named `giver-has-given` and `givee-has-given` are two concepts which are kinds of the concept `have`. The absolute `giver-has-given` represents the idea that one should have something to give to somebody. Note that these ideas are represented by full-fledged concept nodes in KODIAK. This shows that KODIAK encourages the proliferation of concepts. `giver-has-given` is a precondition of `giving` while `givee-has-given` is the result of `giving`, as shown in the figure. Note that several E links are used to express the important relationship among `give`, `giver-has-given`, and `givee-has-given`. The person who gives something to somebody is the same person who has it. The thing that is given by somebody is the same thing that the person has and the same thing that the recipient will have. These are represented by the E links (indicated by "=") in the figure.

The D link between `actor-act` and `giver-give` indicates that the `giver-give` relation is a kind of the `actor-act` relation. The concept `have` is considered to be a kind of the concept `experience` which is again a kind of the concept `stative`. The relation `haver-have` is subsumed by the relation `experiencer-experience`. If `giver-give` had not been specified in the figure, `actor-act` relation would be inherited by the absolute `give` because of the D link between `act` and `give`. But `giver-given` was specified in the figure because

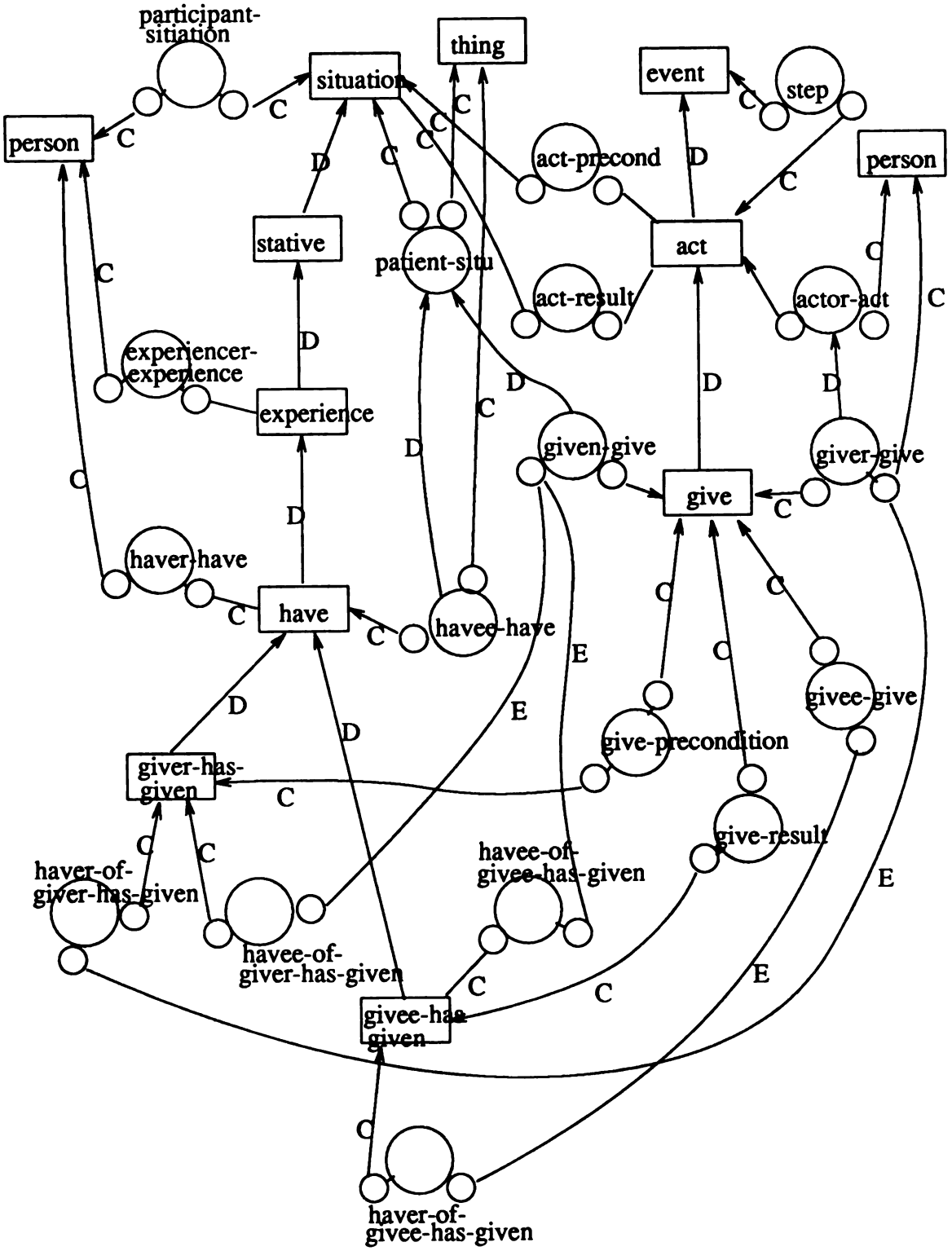


Figure 5.4 Example of a KB

`giver` has more specific meaning than `actor` such as the fact that "the giver of giving" is the same person who originally had the object (this is represented by the E link).

To be a more complete piece of knowledge, some nodes and links should be added to the figure, which specifies the time relation between the concepts. For example, the period of time that `giver-has-giving` occurs is followed (disjointly) by the period of the state `givee-has-given`.

5.2 Marker Passing in SYNSEM

Marker passing is one of the computational methods of using the knowledge base encoded in a network-based model. A marker starts from an origin and spreads through links and nodes. Some information is obtained by examining the collisions of markers from different origins. The idea of marker passing is rooted in the research related to spreading activation models in psychology.

The spreading activation model was first proposed by Quillian(1968, 1969). After that, much research has been done on this model such as Collins and Loftus(1975), Anderson(1983), and Lorch(1982). The main idea in Quillian's language understanding system, TLC, was that there should be a connection in the memory between two concepts if they are related semantically. The method to find the connection(or path) is to activate the two concepts and let the activation spread through the memory and then find the collisions of the activations from two origins. He called this memory "a semantic network" which represents the factual assertions about the world. One type of node in his semantic network is called a "unit" which represents the concept of some object, event, idea, assertion, etc. Another type of node in his network is called "property" which encodes any sort of predication (that can be stated by a verb phrase, adjectival, etc.). A unit or property may contain pointers to other units or properties. In Figure 5.5, a piece of knowledge about "client" is shown. These pointers are links of the memory model. Based on this organization of memory, the program can trace to all the units that

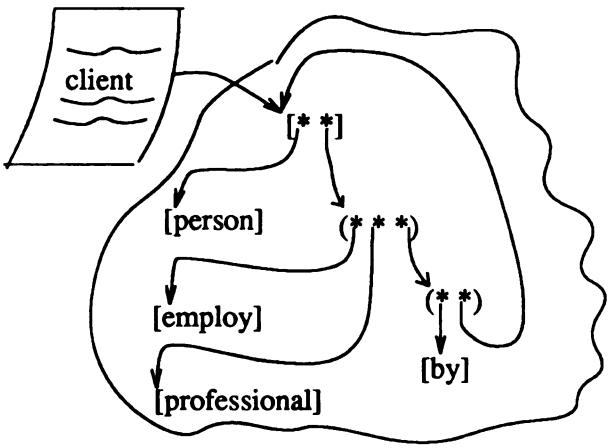


Figure 5.5 Memory of TLC

can be reached from the starting unit by following the pointers, units, and properties. The breadth-first tracing method was used to propagate the tags(i.e. markers). The intersections of tags from different origins were used to interpret the input string.

Following Quillian's idea, Collins and Loftus extended the spreading activation model to provide psychological validity. One of the major extensions is that the activations at the nodes have strengths that decay over time and distance. The intersection can occur only when the total activation at the node should be above some threshold.

The spreading activation model was first introduced into A.I. by Fahlman(1979). Each node of the network memory in his NETL system was implemented using a hardware device called a *node unit.* The nodes are connected by links which are implemented by a hardware unit called the *link unit.* He advocated that the massive parallelism can be achieved with the independent propagations of markers by the node units via the link units. In addition to these hardware units, NETL has a controller which is called the network controller. The network controller can query all nodes to find how they have been marked. He tried to deal with some problems of the knowledge base such as type hierarchies and property inheritance. Posed with certain kinds of questions such as "what color is Clyde?", markers are passed starting from the nodes, "Clyde" and "color". The collisions of the markers from different origins are used to answer the question.

Some special features such as cancellation links are used in NETL to make nodes use more specific information (if it exists) rather than the inherited information.

Charniak(1983a) advocated the use of marker passing for language comprehension. He suggested that the use of context for language understanding can be easily done by using the marker passing paradigm. For example, the effort to solve the word sense disambiguation problem benefits much from the marker passing paradigm. The idea is that the senses that do not form a reasonable path(or collision) are deleted from the list of candidate senses of the ambiguous word. This strategy can even be applied in the case of more than one sentence. Charniak(1986) provided the theoretical basis for the marker massing paradigm. He showed that the operation of finding the paths using marker passing is equivalent to theorem proving in logic. He suggested that anaphora resolution, word sense disambiguation and the prepositional phrase attachment problems can be attacked using the marker passing paradigm.

Granger and his colleagues used marker passing for their text understanding system called ATLAST (Eiselt, 1985; Granger & etc. 1986). ATLAST consists of the Capsulizer, the Proposer, and the Filter. The Capsulizer does the intra-phrase syntactic analysis whose result is passed to the Filter in the form of capsules. While the input word is input into Capsulizer, spreading activation starts from the senses of the word. The activation spreads through the memory which consists of MOPs(memory organization packets) (Schank, 1982a, 1892b). When the intersection of two activations from different origins occurs, the Proposer has found some plausible relationships between word senses. The Proposer passes these possible inference paths to the Filter for evaluation. The Filter performs inter-phrasal syntactic analysis to determine the actor, action, and the object slots. Another major function of the Filter is to evaluate the inference paths and select one path among the competing ones.

Hendler(1986) used marker passing for problem solving. His main idea is to reduce the back tracking in planning by having the marker passing mechanism make

appropriate suggestions as to what to try first.

Most recently, Norvig(1986) used marker passing for getting the plausible inferences in a text understanding system. The knowledge base is written in KODIAK. From the collisions of markers whose origins are input words, six types of plausible inferences are produced. He argued that these inferences are the inferences that should be made to understand the text properly. His major claim is that his approach of using the declarative knowledge base and producing the inferences based on this knowledge base is a generalization unifying the SCRIPT-based and Plan/Goal-based story understanding approaches.

Another recent approach of using spreading activation can be found in the so-called connectionist network model (Feldman and Ballard, 1982). In this memory model, all the work related to syntax and semantics is presumed to be done by spreading the activations through the network. One of the features they use is inhibition as well as activation. (Waltz and Pollack, 1984) and (Small, Cottrell and Shastri, 1982) are examples of attempts to build natural language parsers using the connectionist model.

5.2.1 Links and Flow of Markers

As each word is input into the system, MON creates a new instance node for the word if it is an open word. An instance node is a kind of absolute node. Then MON calls the MP to pass markers starting from the instance node. (Marker passing done by the MP is called *primary marker passing* to distinguish it from another marker passing done by the semantic analysis component.) Figure 5.6 shows that an instance node Z is used as the origin of a session of marker passing. Z is connected to three absolutes(W, X, and Y) because the corresponding word has three senses.

In SYNSEM, the links and nodes through which markers can flow are restricted so that the markers can not propagate too widely. In each row of Figure 5.7, the node in the left column can send a marker via the link in the middle column to the node on the

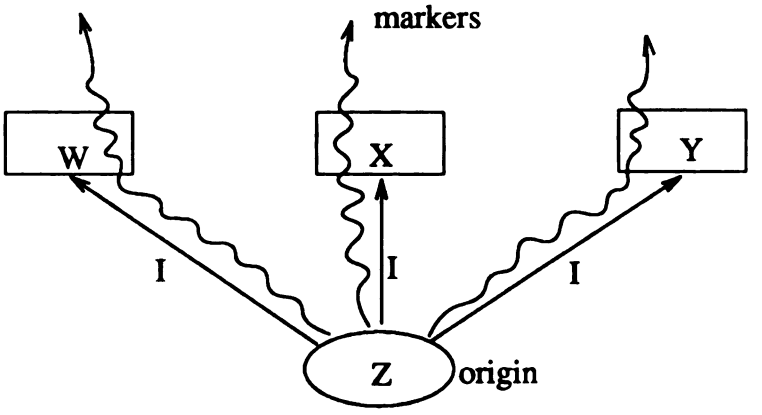


Figure 5.6 Origin of Marker Passing

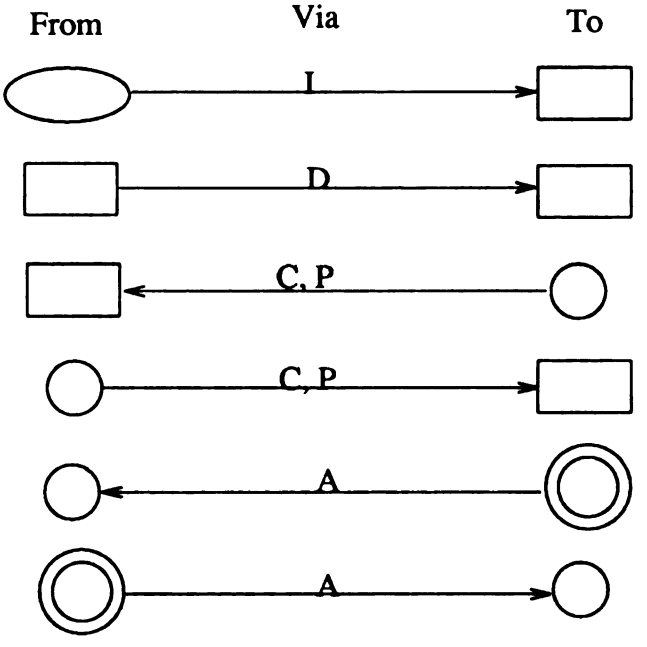


Figure 5.7 Links and Nodes Passing Markers

(ellipse: instance; box: absolute; circle: aspectual; double circle: relation)

right column.

Each marker carries a strength which diminishes as it travels. As a marker passes through each link (except D and I links), its strength is reduced by one. When a marker's strength becomes zero, the marker can not flow any more. Thus, marker passing activates only a subgraph of limited diameter in the overall network. The marker strength at the origin is set to be 6. This value was decided after considering several

factors. If it is too small, the set of collisions produced during marker passing may not include some important paths. But, if it is too big, the initial marker strength may result in too many spurious collisions, which can slow down the system. D and I links do not reduce the marker strength. These links specify the ancestor relationship and it seems that all the ancestors of an activated node should be activated too.

Note that the markers can not flow in the reverse direction of D and I links. The reason is to prevent the markers from propagating too widely in the network. For example, if a high node such as `thing` is marked, then all the nodes under this node (i.e. all objects in the network) will be marked without this restriction. This restriction may result in the reduction of some inferencing power of the system. It has not been found that any important inference is lost because of this restriction.

So the computational cost is the major factor for deciding the initial marker strength and the restriction of links. It is necessary to keep in mind that a marker can not flow back along any link through which it flowed. Without this restriction, markers would get into infinite loops.

5.2.2 Basic Considerations of Marker Passing

The marker passing algorithm used by the MP is strictly a breadth-first algorithm. The whole marker passing operation related to a word is called a session of marker passing. A marker passing session starts from an instance node for an open word from the input string. This instance node is the origin of markers in this session. During the marker passing session, the MP knows the origin and thus the origin need not be included in the information that a marker should carry. A marker carries the following information: (1) the next nodes to visit, (2) the previous node it has just come from, (3) the link through which it has just come, and (4) the strength of the marker.

When a marker visits a node, its trace is put on the node. The trace is a kind of stamp that is used later to find the collisions on the node. A marker trace has the

following information: (1) the origin of the marker, (2) the strength of the marker, (3) the immediate previous node, and (4) the link through which the marker has come to this node from the immediate previous node. A node may be marked by markers from different origins during the parsing of a sentence. Thus a node may have more than one trace. During a session of marker passing, a marker may collide with other markers from other origins. This happens when a marker reaches a node which has a trace of a marker from any previous marker passing session. The collisions which occur during the marker passing session are collected and reported to the caller of the MP after the session.

If a marker reaches a node which does not have the marker trace of the same session, this means that this node is marked for the first time in the session. There is no problem in this case. A marker trace for this marker is put on the node and the marker is passed to the neighboring nodes. This case is illustrated in Figure 5.8.

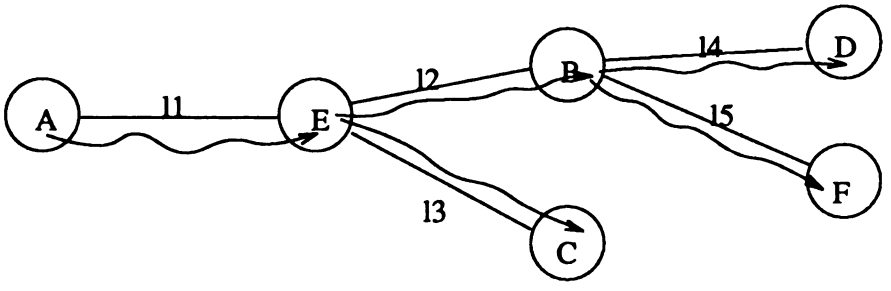


Figure 5.8 Split of Markers

Consider a case that a marker has arrived at node E from node A, and E has never been visited during the session. A marker trace is put on E which is a list: (origin strength A l1). If the strength is not 0, the marker should be propagated to the neighboring nodes A, B, and C assuming that l1, l2, and l3 are legal for the flow of markers. Because this marker has just come from A via l1, it can not flow through l1 again. But l2 and l3 can be used for passing the marker, and thus B and C will get marked. It can be said that the marker at E is split into two markers. One of them will be a marker that flows through l2 and mark B. The other one will flow through l3 and mark C. The mark at B will mark D

and F later. If a marker at a node, say node-x, flows to the next node, say node-y, via a link, then the marker at node-y is called a descendant of the marker at node-x.

Complication arises if a marker reaches a node which already has a marker trace of the same marker passing session (i.e. this node was already visited during this marker passing session). Two possibilities can be considered in this case: (1) the marker is the descendant of the marker that already marked the node, (2) the marker is not the descendant. Case (1) happens when a loop has been formed as shown in Figure 5.9.

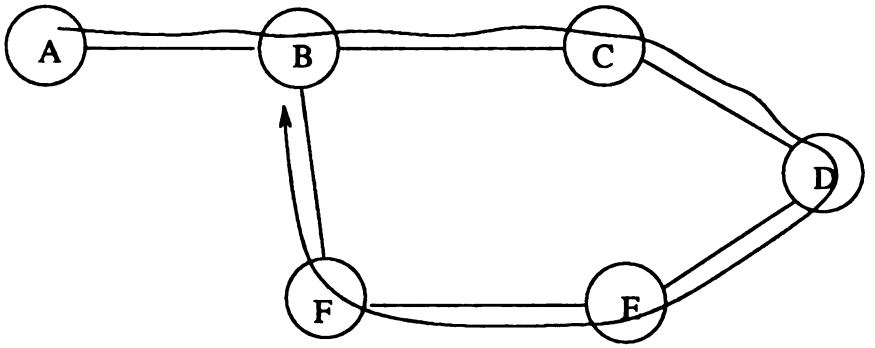


Figure 5.9 Loop in Marker Passing

The marker that marked the node B propagates to B again (in the guise of a descendant) via the path that is a loop in this figure. In this case, B is not marked again. This marker which returned via a loop is just thrown away; Otherwise, marker passing will go on infinitely along the loop. Case (2) is illustrated in Figure 5.10. Node B was marked by a

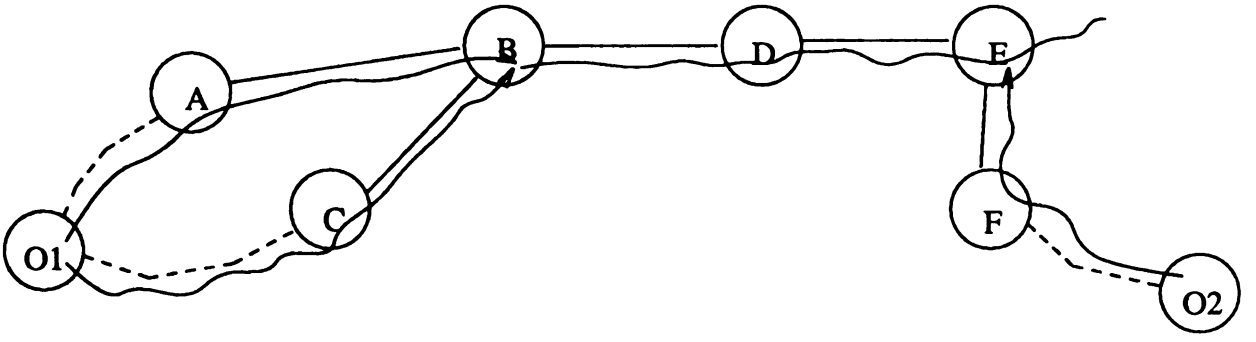


Figure 5.10 Merge of Markers

marker that reached B via A from the origin. Later, a marker which followed a different path reaches B via C. This marker is not the descendant of the marker which put a

marker trace on B earlier. In this case, the marker trace at B is modified in such a way that the previous node field of the marker trace contains both A and C. We call this operation "merging of two markers". But additional processing is required for this operation. Note the following fact. When a collision is reported, the paths to the origins are computed and stored as part of the data about the collision. Because of this, the path information of the collisions whose path goes through B should be updated so that the new added branch can be a part of the paths in the path information of the collisions. The easiest way to explain this processing is to use an example. In Figure 5.10, let's assume that a marker propagated from origin O1 to E via A, B, and D. At E, a collision occurred with a marker from origin O2 (of a previous marker passing session). According to this collision, the path (O1 ... A B D E F ... O2) is computed and stored. Later, "merging of two markers" occurs at node B. Then the path information of the above collision should be updated so that another path corresponding to the new branch can be added. Thus the following path should be added to the collision: (O1 .. C B D E F .. O2).

5.2.3 Follow-on Collisions

As explained before, a collision is reported if a marker reaches a node which was marked from another origin in a previous marker passing session. As soon as a collision is detected, the MP finds all the paths corresponding to the collision by tracing the markers to the two origins. There can be more than one path found because of the merge of two markers as explained in the previous section. For each collision, the following information is gathered and stored: the node at which the collision occurred, paths corresponding to the collision, and the two origins involved. At the node of a collision, the fact that a collision between two origins occurred is logged for later use.

Normally collisions are expected to provide useful information. But, in reality, many useless collisions called *follow-on collisions* (Hendler,1986) are reported. Let's consider Figure 5.11 and assume that the nodes, A, B, and C were marked by markers

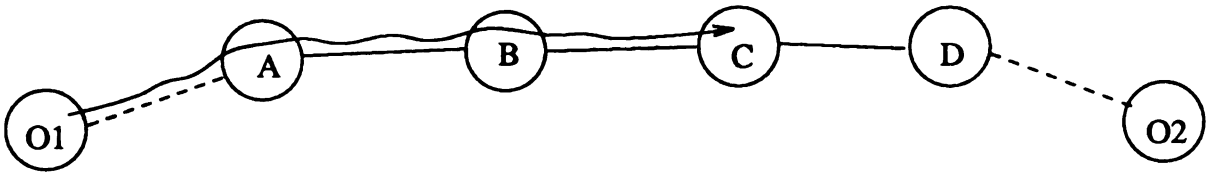


Figure 5.11 Follow-on Collisions

whose origin was O1 during the previous marker passing session. Assume that a marker reaches C during the marker passing session corresponding to the origin O2. This collision is a normal collision and is called a *regular* collision. The marker will reach B from C. A collision is also reported at B because B was marked by a marker from O1. The marker will reach A from B and a collision will also be reported at A. But these collisions at A and B are the dummy collisions corresponding to the regular collision at C, because they provide no additional information. All these collisions correspond to the same path, (O1 .. A B C D .. O2). From this example, we can see that more than one collision can be reported for the same path. Out of these collisions, only one can represent the path and others can be considered to be dummy collisions. Thus we call the first collision *a regular collision* and the others the *follow-on* collisions. But note that there can be more than one regular collision between two origins as shown in Figure 5.12.

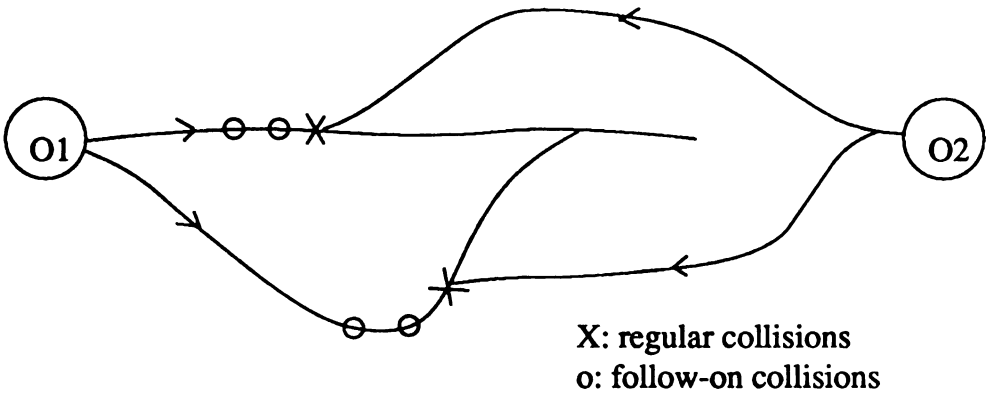


Figure 5.12 Regular and Follow-on Collisions

Detection of the follow-on collision is done as follows:

When a marker (whose origin is O2) reaches a node, say A, which was visited by a marker of the previous session (related to the origin, say O1); this means that a collision occurred at A; Identify the previous node (say B) of this marker and check if either a follow-on or regular collision between markers from O1 and O2 occurred at that previous node (and the marker from O1 at node A was from node B); If it did, the collision at node A is a follow-on collision; Otherwise, the collision is a regular collision.

According to this detection method, it is clear that the collision at node C in Figure 5.12 is the regular collision and the collisions at node B and A are follow-on collisions. The follow-on collisions are also logged on the node at which they occur so that the information can be used later.

5.2.4 Overlapping Collisions

It is necessary to consider another kind of useless collision in addition to the follow-on collisions. Let's consider Figure 5.13 and assume that the marker

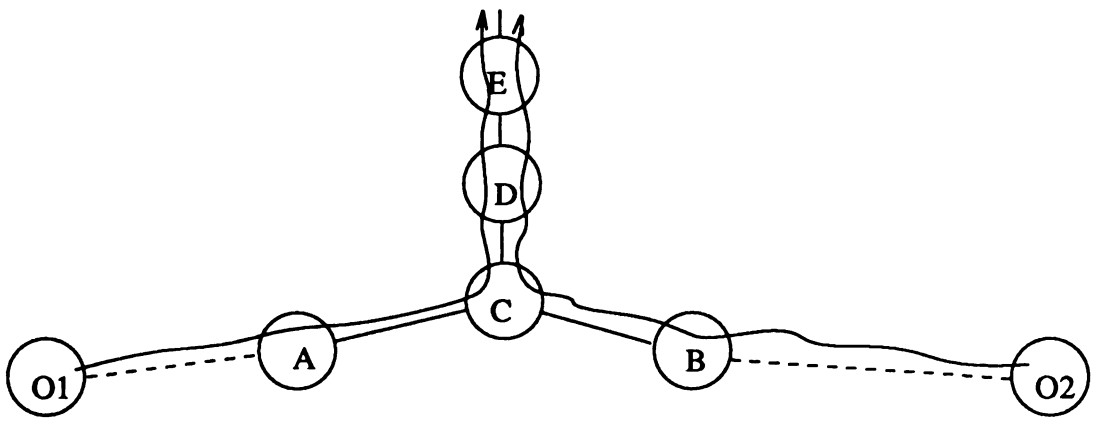


Figure 5.13 Overlapping Collisions

whose origin is O1 flows through the nodes in the following order: A, C, D, E, etc. Assume also that the marker from origin O2 flows through the nodes: B, C, D, E, etc. The collisions between the two markers occur at C, D, and E. The collision at C is the regular collision. The fact that the collisions at D and E are false collisions is clear when

the paths for the collisions are compared as shown below:

Collision C1 at C: (O1 .... A C B ..... O2)

Collision C2 at D: (O1 .... A C D C B ....O2)

Collision C3 at E: (O1 .... A C D E D C B ....O2)

The paths C2 and C3 have no more information than C1, because the segment "CDC" and "CDEDC" does not add any information. Thus the collisions, C2 and C3, are the dummy collisions of C1. Note that these dummy collisions can not be detected by the method to detect the follow-on collisions. We call the collisions such as those at D and E *the overlapping collisions* because the flows of the two markers overlap.

The scheme to detect an overlapping collision is simple. At a node at which a collision occurs, say D, compute the paths to the two origins. If the two paths share some nodes and links, then the collision at the node D is an overlapping collision. In Figure 5.13, a collision occurred at D. The paths to the two origins are (D C A .... O1) and (D C B ....O2). Because (D C) is shared between the two paths, the collision at D is an overlapping collision. The collision at E is also an overlapping collision because the path from E to O1, i.e. (E D C A ...O1), and the path from E to O2, i.e. (E D C B ...O2), share the portion (E D C). Overlapping collisions are just thrown away.

5.2.5 Marker Passing Algorithm

Now the overall flow of the marker passing algorithm used in SYNSEM can be introduced. It is shown in Figure 5.14.

The initial strength of the marker is set to some fixed number(6 in the current implementation) and the strength is reduced by one as it passes a link(except D or I link). Therefore the number(say W) of nodes being marked during a marker passing session does not increase exponentially according to the size of the KB(say N) (

$$W=(kN)^6$$

because the branch factor can be assumed to be kN where k is a constant and N is the number of nodes in the KB). It follows that the number of collisions and the number of possible paths for collisions do not increase exponentially according to the size of the KB.

```
step 0: insert a marker (origin nil nil 6) into an empty queue, Mqueue.
step 1: if Mqueue is empty, goto stop.
        Marker <- a marker popped out from the front of Mqueue.
        Set currentnode, previousnode, fromlink and strength from Marker.
        if currentnode already has a trace of this session
          then if the situation is infinite loop
                then goto step 1
                else begin
                      update the trace so that previousnode is added;
                      update related collisions to change the path
                        information;
                      goto step 1
                     end
           else begin
                 find and prepare the markers for
                     all neighboring nodes to which the marker can flow;
                     (if strength is 0, A or C links can not be used.)
                 insert the markers into the rear of Mqueue;
                 if there are traces for previous sessions
                  then
                    for each trace  do
                     if overlapping collision
                       then do nothing
                       else if follow-on collision
                              then report and log follow-on collision
                              else report and log regular collision;
                 goto step1;
                end
step 2: stop;
```

Figure 5.14 Marker Passing Algorithm

5.3 Conclusion

A knowledge base written in KODIAK is used as the source of semantic information for the SYNSEM parser. KODIAK has three types of nodes: *absolutes* representing the objects, states, and actions in the world; *relations* representing the possible relationships that can exist between two absolutes; and *aspectuals* which are formal parameters of relations. The important feature of KODIAK is that it is a relation-based knowledge representation language. The concepts hidden behind the slots in the frame-based languages become full-fledged concepts in KODIAK. In KODIAK, the proliferation of concepts is encouraged in contrast to languages such as Conceptual Dependency that provide a set of primitives. The meaning of a concept in KODIAK is generated by its connections to other concepts. KODIAK provides nine primitive links that can connect the concepts.

The marker passing mechanism is used in SYNSEM as a simple way of providing semantics to the parser. The detailed considerations on the design of the marker passer have been explained. To reduce the space of the KB that gets marked and thus minimize the computational cost, some constraints have been put on the marker passing: restricting the possible link and node combinations that markers can flow through and restricting the distance that a marker can travel. The critical design problems such as follow-on collisions, merging of markers, detecting the loop, and overlapping collisions have been explained. Use of marker passing for providing semantics to parsing and for resolving word sense ambiguity is discussed in the next two chapters.

CHAPTER 6

PROVIDING SEMANTICS TO PARSING

It was explained that the parsing is split into several branches when more than one rule is found by the syntactic pattern matcher. Each branch is a legitimate way of parsing at that point. The parser removes the branches whose semantic processing result is worse than others by pushing these into the backup stack (i.e. the parser pursues only the top-rated parses). The number of branches will be reduced (usually down to one) as soon as syntactic or semantic information allows the parser to remove some branches. Therefore the important problem in this parsing framework is how the semantic processing result is computed and how the semantic processing results of the multiple branches are compared.

It was explained in the previous chapter that the source of semantic information is the KODIAK knowledge base and the major technique to utilize it is the marker passing mechanism. A decision situation that requires semantic information which can not be represented in KODIAK can not be handled in SYNSEM. Thus, we can only claim that SYNSEM can handle the ambiguities which can be resolved by the kind of knowledge that can be encoded using KODIAK.

This chapter will start with the explanation of the interface between the syntactic analysis and semantic processing component. The most important concept is the *suggestion* which the syntactic analysis component sends to the semantic processing component. Any rule that invokes some semantic processing has an action which issues a suggestion. Each suggestion is interpreted by the semantic processing component and the result is stored as a part of the semantic interpretation of the branch. The semantic processing result of each suggestion is realized as a path (or a set of multiple paths if

there is ambiguity) in the KODIAK knowledge base. Marker passing is useful for finding a path for the suggestion. One problem with marker passing is that its computational cost is big. One reason for this big computational cost is that there are too many spurious collisions(or paths) reported by the marker passing routine. In this chapter, an efficient way to remove false collisions is introduced. The problem of comparing the branches according to the semantic processing result becomes a matter of comparing the paths. This problem is also investigated in this chapter.

6.1 Interfacing Syntax and Semantics

The main method of interfacing the syntactic processing component(SYN) and the semantic processing component(SEM) is to use the suggestions that SYN sends to SEM. A rule whose execution should invoke some semantic processing contains actions that issue a suggestion. For example, the rule main-verb2 has an action:

(sug NP1 V1 subj)

This suggestion indicates that the syntactic constituent NP1 and V1 are related by the syntactic relationship "subj". It says that NP1 is the syntactic subject of the clause whose main verb is V1. The semantic component should consider this relationship when it is computing the semantic relationship between them.

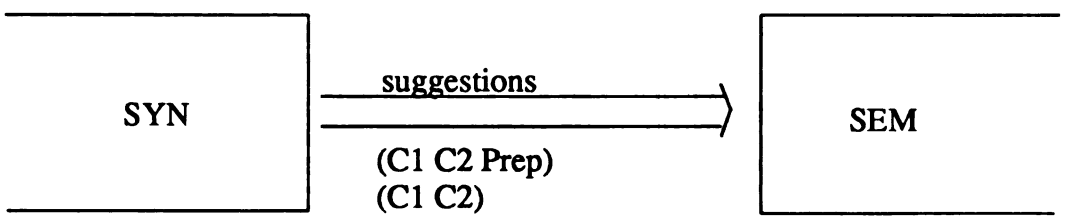


Figure 6.1 Flow of Information from Syntax to Semantics

What the rule main-verb2 tells SEM is that SEM should find the appropriate semantic connection between the semantic object corresponding to the syntactic constituent NP1 and the semantic object corresponding to V1. But SYN includes a hint "subj" which should be utilized by SEM. When SYN has analyzed a syntactic object of a clause, it

will send a suggestion to SEM saying that the verb of the clause and this object should be connected via the semantic relation that can be implied by the syntactic relationship "object". Thus the rule **object** (which does this analysis) has an action of the form:

(sug V1 NP1 obj)

The suggestion sent by SYN according to the action has the form of "(C1 C2 Prep)". C1 and C2 are syntactic constituents and Prep is a preposition (including pseudo-prepositions) indicating the syntactic relationship between C1 and C2. Prep is used as a hint to SEM for connecting the semantic objects corresponding to C1 and C2. Pseudo-prepositions are the dummy prepositions that actually do not exist but can be assumed to exist. "Subj" and "obj" are examples of pseudo-prepositions. "Indirect-object" can also be a pseudo-preposition. But an indirect object can eventually be replaced by the dative preposition of the dative verb. For example, "to" is the dative preposition of the dative verb "give", and "for" is the dative preposition of the dative verb "buy". Thus, suggestion (V1 NP1 indirect-object) can be replaced with (V1 NP1 to) if V1 is the verb "give".

Let's consider (6-1) and (6-2) to provide examples for suggestions.

(6-1) The man ate a cake with a fork.

(6-2) The man ate a cake with frosting.

In (6-1), "eat" and "a fork" are related via the PP which is headed by the preposition "with" (refer to Figure 6.2(a)). The preposition "with" provides some information about how "eat" and "a fork" are related. Using the fact that the V, "eat", and the PP, "with a fork", compose the VP syntactically, SYN knows that the V and the NP of the PP should be connected in some way and this relationship needs to be implied by the meaning of the preposition "with". The preposition "with" might have many meanings. It is up to the semantic component to use the correct meaning of "with" to connect "eat" and "a fork". So, for (6-1), the syntactic rule which recognizes this syntactic relationship (i.e. the rule **vp-pp**) has an action which issues the suggestion (V1=eat NP1=a fork prep=with) and sends it to SEM. For this suggestion, SEM finds the KB path which connects "eat" and

"a fork". The preposition "with" is used to find the correct path (i.e. the path that conveys one of the meanings of "with").

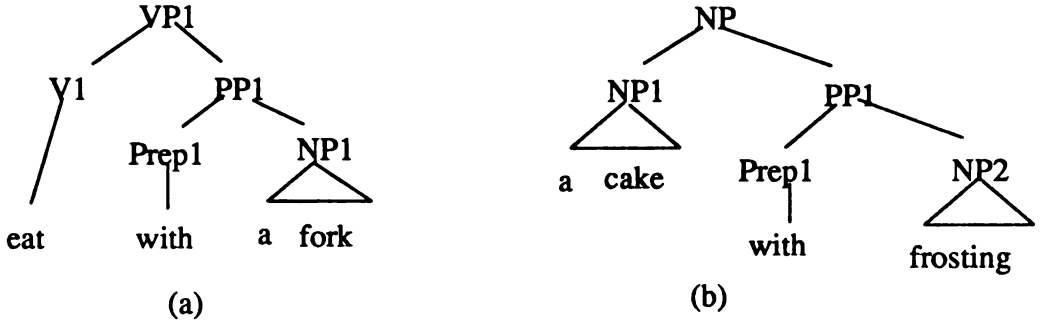


Figure 6.2 PP Attachment Depending on Semantics

In the case of (6-2), the NP "a cake" is modified by the PP "with frosting" (consider Figure 6.2(b)). This analysis is done by the rule np-pp. This rule knows that NP1 and NP2 would be related in some way semantically and this relation is implied by one of the meanings of "with". Thus, this rule has an action which issues the suggestion (NP1=a cake NP2=frosting prep=with).

According to the framework described so far, it is clear that a syntactic rule to which some semantic processing is related has some actions which issue suggestions to SEM. The decision about which rules should have what kind of suggestions should be determined by the grammar writer. Note that a suggestion action is of the form (C1 C2 Prep) where C1 and C2 are syntactic constituents. Thus the specification of a suggestion action is in purely syntactic form which is domain independent.

So far, we explained how the syntactic analysis component provides SEM with a message which is used for semantic processing. This is the way syntax requests semantics to do some semantic processing. The next question that arises is how semantic processing influences the processing of syntax. This is achieved by the removal of some branches based on the semantic processing result. Consider Figure 6.3. Each branch is created for each syntactic rule whose patterns match CLIST. For each branch, there are suggestions which are sent to SEM. The KB path for each suggestion is found by SEM

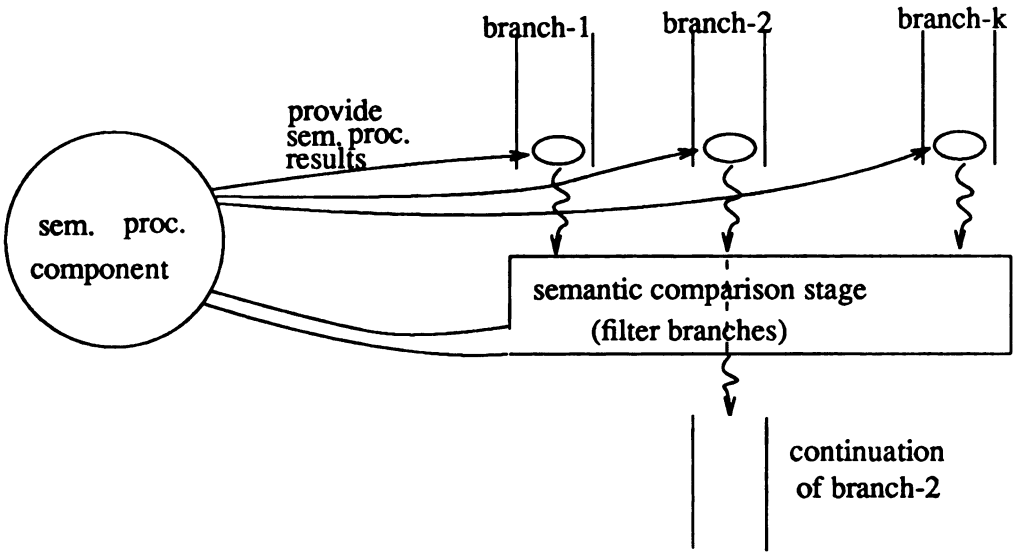


Figure 6.3 Influence of Semantics in Parsing

and it is returned to the branch to be stored. At the semantic comparison stage, the branches are filtered according to the semantic processing result (actually the goodness of the KB path). Thus the branches having inferior semantic effect are suppressed by pushing them into the backup stack and the branches (i.e. syntactic analysis alternatives) with best semantic effect survive. This is one way that semantics can influence the processing of syntax.

Another way that semantics can influence syntactic processing is to notify SYN that the suggestion is so bad that a reasonable path can not be found. In this case, the branch that issued the suggestion is just thrown away (without being backed up in the stack).

Interfacing syntax and semantics explained in this section are the major ways of how syntax and semantics communicate in SYNSEM. But it should be noted that it might be necessary to have other ways of making syntax and semantics exchange information. Natural language is too complex to classify the communication between the syntactic processing level and the semantic processing level in only a couple of ways. But we claim that the methods of interfacing SYN and SEM explained in this section are

major ways of communication.

6.2 Finding a Path for a Suggestion

As explained in the previous section, SEM should find a path in the KB corresponding to the suggestion. The path should be the best path for the suggestion. Normally only one path is the best one, but there can be more than one best path if there is word sense ambiguity. The suggestion that is issued by the suggestion action of a syntactic rule is specified using the syntactic constituents. Then this suggestion is converted into the suggestion specified using the nodes in the knowledge base. Let's consider sentence (6-1) as an example. The result of syntactic analysis after "ate" has been read is shown in Figure 6.4(a). This analysis is done by the rule main-verb2 which contains a suggestion

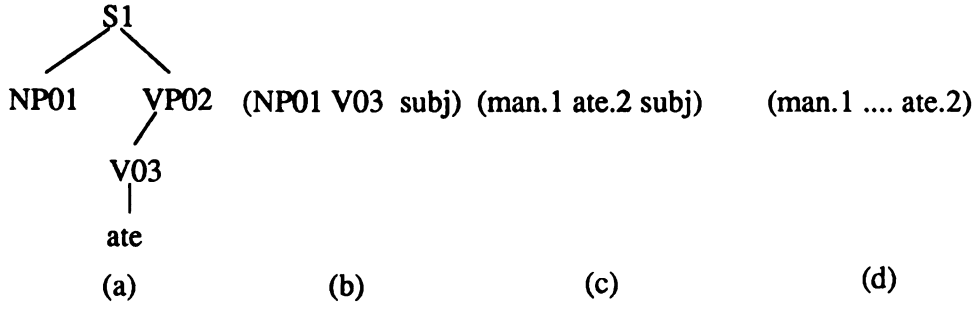


Figure 6.4 Conversion of Suggestions

action (sug NP1 V1 subj) as explained before. During the pattern matching of the rule, NP-1 is mapped to NP01 and V-1 is mapped to V03. According to this action, the suggestion in (b) is produced by SYN. Note that NP01 and V03 are actually pointers to the syntactic constituents in the tree in CLIST (shown in (a)). The suggestion in (b) is converted into the suggestion in (c) by replacing the pointers with the corresponding concept nodes in the KB. Refer to the KB shown in Figure 6.5 to follow the example in this section. The suggestion in (c) is of the form `(I1 I2 Preposition)` where `I1` and `I2` are the KB nodes and `Preposition` is the preposition given by the rule ("subj" in this example). NP01 in (b) is replaced with `man.1` which is the instance

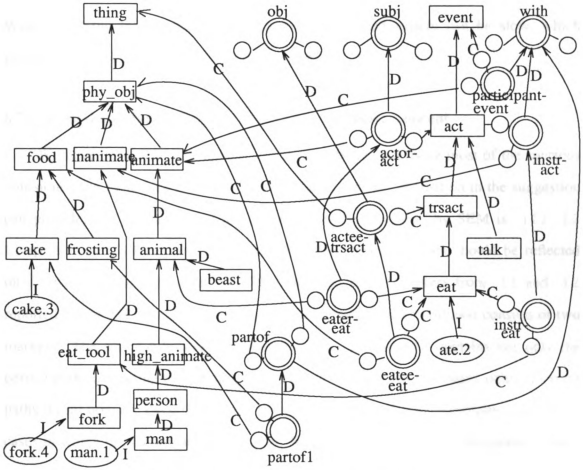


Figure 6.5 Another Example of the KB

absolute created by MON. MON created it when it input the word "man". In a similar way, V03 in (b) is replaced with the instance node `ate.2` in the KB. Thus the suggestion in (c) is obtained. The conversion of the form is done by the interface routine.

After getting the suggestion in (c), all SEM should do is to find the best path between the node `man.1` and `ate.2` via some relation node that has one of the meanings that "subj" can indicate. Note that the MP passed markers from the instance nodes when the nodes were created. The collisions of markers whose origins are `man.1` and `ate.2` are available. A collision corresponds to one or more paths between the origins of the markers that collide. One of the biggest problem that marker passing-based systems suffer from is that there are too many spurious(false) collisions (Charniak, 1983a,

1986). Charniak said that the ratio of good collisions to the spurious collisions is 1 to 10. Without solving this problem, the marker passing-based system can be slow, which prevents it from being used as a practical system.

6.2.1 Secondary Marker Passing: Motivation and Basic Concept

We have developed a technique which can efficiently remove most of the spurious collisions. The basic idea is to use the information that the preposition in the suggestion can provide. Let's assume that the suggestion to be processed by SEM is `(I1 I2 Preposition)`. `Preposition` provides some meaning which should be reflected on the path between `I1` and `I2`. The collisions of the markers from `I1` and `I2` reported by the MP are called two-way collisions because each collision consists of two markers. The set of two-way collisions between `I1` and `I2` contains not only the correct paths (reflecting the meaning of `Preposition`) but also many other incorrect paths. From now on, the collision and the path will be used interchangeably, except for cases in which some confusion may occur. Note that a collision can correspond to more than one path because of the "merging" of markers explained in Chapter 5. Without the information `Preposition` provides, it is difficult to remove the spurious collisions. (Even if it can be done, it takes too much time.) It is necessary to develop a technique by which the spurious collisions are removed efficiently.

The technique developed for this purpose is called *secondary marker passing*. Most of the spurious collisions can be efficiently removed using this technique. This technique removes all the collisions whose path does not reflect one of the meanings of `Preposition`. Let's consider sentence (6-1). The fact that "the man" has the syntactic relation of "subject" with "eat" indicates that they are related by a semantic relation that can be implied by the pseudo-preposition "subj". The input string "ate" and "a fork" are syntactically related via the preposition "with". This indicates that the concept for "ate" and the concept for "a fork" should be connected in the KB via a semantic relation that can

be implied by one of the meanings of the preposition "with". From this consideration, we can say that the best path between `I1` and `I2` should pass through a concept node whose meaning can be implied by `Preposition` in the suggestion `(I1 I2 Preposition)`.

The concept nodes which can represent the semantic relations that are implied by the meanings of the prepositions are the nodes of the type "relation" in KODIAK. The relation nodes such as `actor-action`, `eater-eat`, `recipient-receive`, etc. are some of the nodes that can be implied by the pseudo-preposition "subj". The relation nodes such as `instrument-action`, `instrument-eat`, `has-part`, etc., are implied by the preposition "with". We can think of the hierarchy of the relation nodes for each preposition in the KB. The root of the hierarchy can be assumed to be the preposition itself. We can recognize these hierarchies in the KODIAK knowledge base. The hierarchy in Figure 6.6(a) is for the pseudo-preposition "subj", and

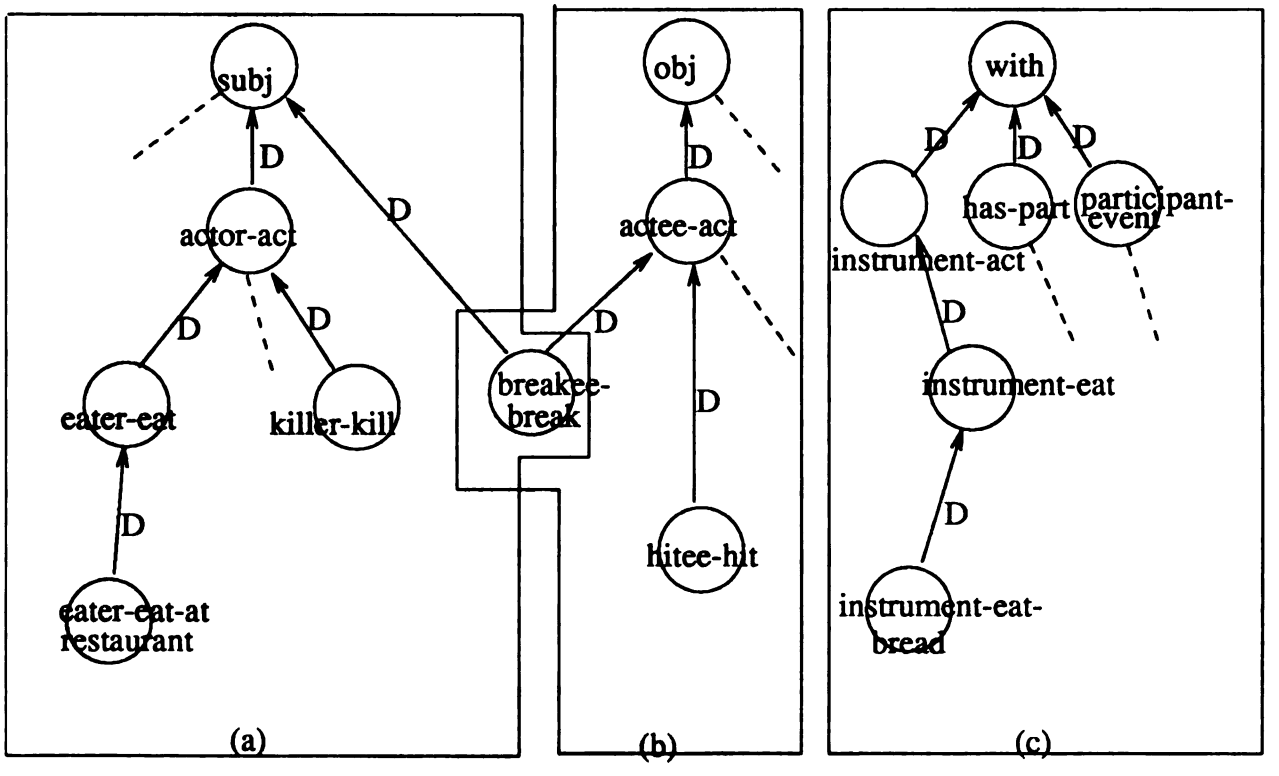


Figure 6.6 Hierarchies of Relations

that in (b) is for the pseudo-preposition "obj", and that in (c) is for the preposition

"with". Note in the figure that `breakee-break` belongs to the two hierarchies. For example, "the rock" in (6-3) is the syntactic object of "broke" and the relation `breakee-break` is implied by "obj" but the rock in (6-4) is a syntactic subject of "broke".

(6-3) The man broke the rock with a hammer.

(6-4) The rock broke.

Thus the relation node `breakee-break` is implied by "subj". The root node of each hierarchy in Figure 6.6 is a dummy node representing the preposition. The link between the relation nodes in the hierarchy is implemented using the D link in KODIAK. If the relation node A is a super-class of another relation node B, then there is a D link coming from B to A.

For the suggestion `(Instance1 Instance2 Preposition)`, the best path between `instance1` and `instance2` should pass through a relation node that is implied by `Preposition`. This means that the path should pass through a relation node that is in the hierarchy whose root is `Preposition`. The paths found by the MP (i.e. two-way collisions) that do not pass through a relation node in the corresponding hierarchy need not be considered.

This idea can be implemented using another marker passing. A marker different from the markers used by the MP are passed from the root node of the hierarchy for the preposition in the suggestion. The markers used in this process are called *the secondary markers* (note that the markers used by the MP are called the primary markers). All the relation nodes in the hierarchy can be marked by the secondary markers if the markers propagate via D links between relation nodes in the hierarchy. Any path which does not contain a relation node that is marked by a secondary marker can be removed as a false path because it does not pass through any relation node that has a meaning of the preposition. At this point, it is important to note that the relation node of a good path which is marked by a secondary marker should also have markers from `instance1` and

instance2 as shown in Figure 6.7.

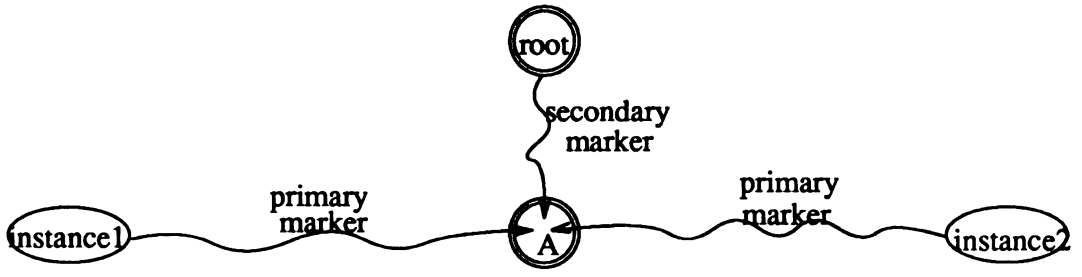


Figure 6.7 Three-way Collision

The reason is that this relation node, node A in the figure, represents the semantic relation between instance1 and instance2 and thus the node is related to both instance1 and instance2. Thus, the markers from the two origins should be able to reach the node A. This means that three markers should collide at the relation node A as shown in Figure 6.7. This collision is called *a three-way collision* because the markers from three different origins form the collision. From this consideration, the task of removing the paths which do not contain a relation node marked by a secondary marker can be achieved by collecting only the three-way collisions. Then the routine for finding the best path needs to consider the three-way collisions instead of the two-way collisions. This will save a lot of computation because the number of two-way collisions(or the paths) should be much bigger than the number of three-way collisions (the data of the experiment will be given later).

It has been explained that all the relation nodes in the hierarchy for Preposition are marked during the secondary marker passing session. But this is not true. The amount of computation can be reduced using an updated scheme. For a large knowledge base, the number of relation nodes in a hierarchy for a preposition can still be big. Consider, for example, the pseudo-preposition "subj". The concept nodes for almost all verbs in the domain are linked to relation nodes that are implied by "subj", which means that the number of relation nodes in the "subj" hierarchy is almost the same as the number of verbs in the domain. To reduce the number of relation nodes that are marked during the

secondary marker passing session, the following scheme is used. When a relation node gets marked by a secondary marker, SEM can pass the secondary markers from this node only if a three-way collision has occurred at this node. Thus further propagation of the secondary markers stops at a relation node at which no three-way collision occurs. According to this scheme, the system need not mark all the relation nodes in the hierarchy. Only part of the hierarchy will get marked. Consider the suggestion `(eat.1 spoon.3 with)` for an example. Only the portion of the "with" hierarchy corresponding to the node `instrument-action` will be marked as shown in Figure 6.8.

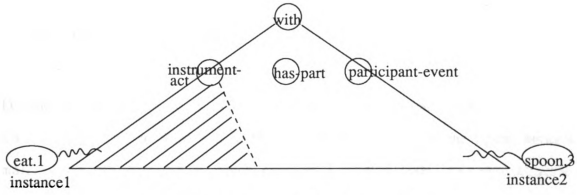


Figure 6.8 Cut-down of Marker Passing Space

The sub-hierarchies corresponding to `has-part` and `participant-event` will not get marked during secondary marker passing for the suggestion because there is no three-way collision at `has-part` or `participant-event`.

Now, the secondary marker passing session can be stated more precisely as follows:

1. A secondary marker passing(SMP) session occurs for each suggestion from SYN. But the suggestion should contain a preposition. Secondary marker passing is done by SEM before it attempts to find the best path for the suggestion.
2. The output of an SMP session is a set of three-way collisions. This output is used by the routines for finding the best path.

3. The origin of secondary marker passing is the dummy root node of the hierarchy corresponding to the preposition.
4. A marker is passed only through the D links (in the reverse direction) between the relation nodes.
5. A marker which reaches a relation node can not flow any more if a three-way collision does not occur at this relation node.
6. There is no limit to distance that a secondary marker can travel.
7. All secondary markers are removed after each SMP session.

6.2.2 Consideration on the Polarity of Paths

Related to secondary marker passing, is one problem that should be considered during the SMP session. This problem can explained using (6-5) and (6-6).

(6-5)The man with moustache ....

(6-6)The moustache with the man ....

During the analysis of (6-5), SYN would send a suggestion `(man.1 moustache.2 with)` to SEM and SEM will find the path, (1) in the figure, between `man.1` and `moustache.2` which contains a relation node `has-moustache`, because a three-way collision occurs at `has-moustache` (illustrated in Figure 6.9).

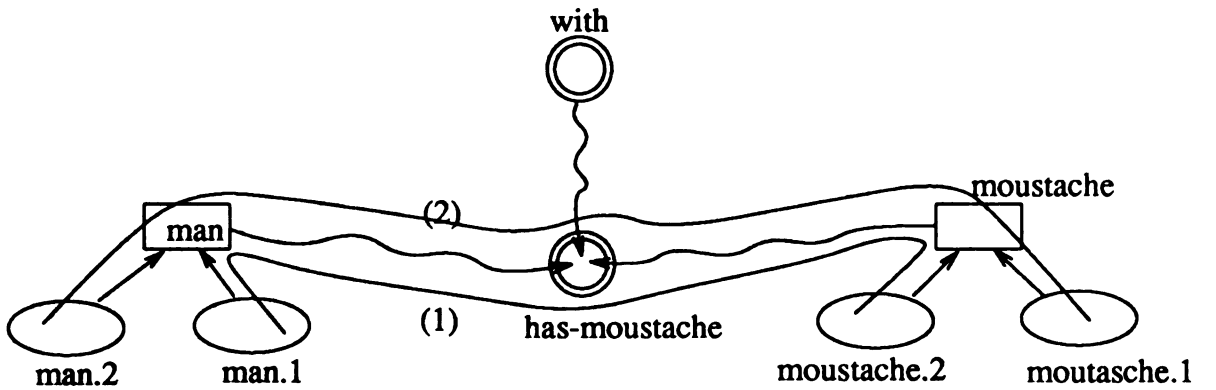


Figure 6.9 Two Collisions with One to be Removed

In the case of (6-6), the suggestion `(moustache.1 man.2 with)` is sent to SEM. Note that the same path, (2) in the figure, as the path for (6-5) is found in this case according to the three-way collision during the secondary marker passing operation. But

(6-6) is semantically unacceptable even if a good path is found. Thus either this path should not be found during secondary marker passing, or the path is found during secondary marker passing but is rejected later by some filtering routine. From this observation, it is realized that the order of two concepts in the suggestion is important. (man.1 moustache.2 with) is semantically acceptable and a good path is found, but (moustache.1 man.2 with) is semantically bad and thus no path should be found by SEM.

This kind of false path can be removed during secondary marker passing. The idea is to use the order of aspectuals for a relation. We put an order for two aspectuals of each relation in such a way that the aspectual whose corresponding word appears before the word corresponding to the other aspectual is given the first place and the other aspectual is given the second place. For example, the relation subj (the root node for the hierarchy of pseudo-preposition "subj") has two aspectuals, pre.subj and pos.subj. Pre.subj is the aspectual that corresponds to the input string of the syntactic subject. Pos.subj is the aspectual that corresponds to the verb of the clause. Pre.subj is called the first aspectual and pos.subj is called the second aspectual of subj because the subject occurs before the main verb in a clause. Note that aspectuals of two relations that are connected by a D link are paired by the role/play relationship as illustrated in Figure 6.10. Actor-act dominates eater-eat (i.e. there is a D link from the latter to the former node). The aspectual actor.actor-act is in the role/play relationship with eater.eater-eat. Similarly act.actor-action is in the role/play relationship with eat.eater-eat. The first aspectual of subj, pre.subj, is linked to actor.actor-act. Actor.actor-act is linked to eater.eater-eat by the role/play relationship. The second aspectual of subj, pos.subj, is linked to act.actor-act which is linked to eat.eater-eat by role/play. The role/play relationship is used to link the aspectuals as shown in the figure. Because the subject appears before the verb the aspectuals that denote the subject are

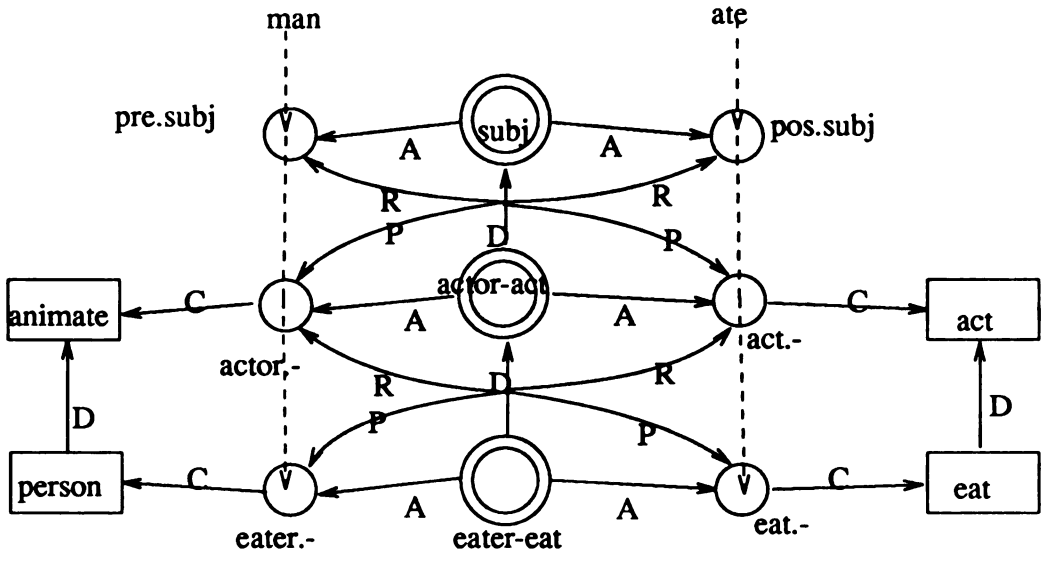


Figure 6.10 Role and Play of Aspectuals of Relations

linked to the first aspectual of `subj`, `pre.subj`.

Figure 6.11 shows the case for the preposition "with". Note that the first aspectual of `with` is linked to the aspectual `whole.has-moustache` of `has-moustache` because the referent of `whole.has-moustache` appears before the preposition "with" followed by the referent of `part.has-moustache`.

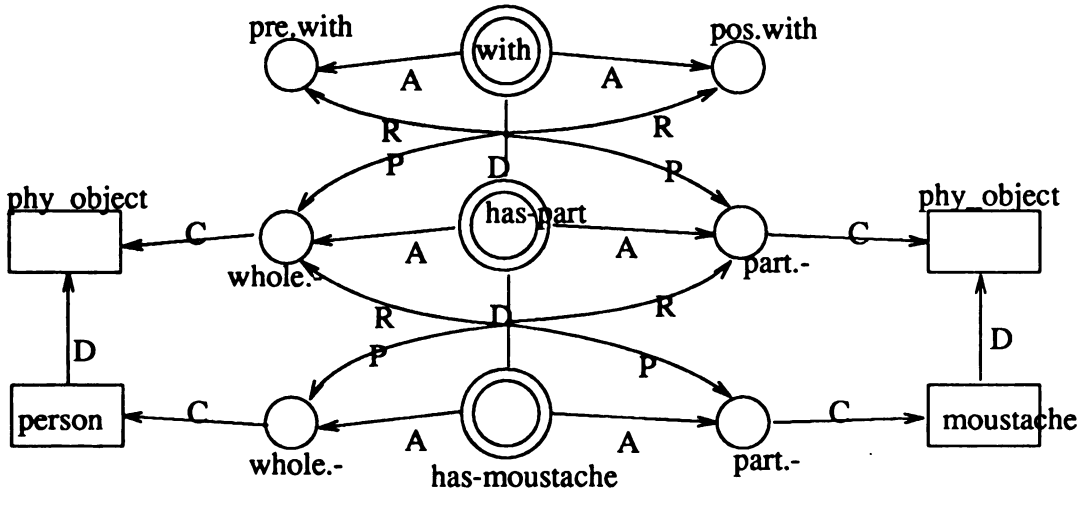


Figure 6.11 Example of Role and Play

Let's explain how the path is rejected for (6-6). SYN always puts the constituent appearing before the preposition (here "with") in the first element of the suggestion and the

constituent after the preposition in the second element of the suggestion. Thus the suggestion for (6-6) is `(moustache.1 man.2 with)`. For each three-way collision that has been found, SEM checks to see if the first instance node in the suggestion (`moustache.1` in this example) and the aspectual (of the relation of the three-way collision) which is in the role/play relationship with the first aspectual of the preposition node, are on the same part when the path is divided into two halves by the three-way collision node. In the example shown in Figure 6.11, the aspectual of `has-moustache` which is linked to the first aspectual of `with` is `whole.has-moustache`. The two half paths are:

```
(man.2 person whole.has-moustache has-moustache)
(has-moustache part.has-moustache moustache mous-
tache.1)
```

We can see that `whole.has-moustache` and `moustache.1` are not on the same half path. Therefore this three-way collision is thrown away.

Related to the problem discussed so far, it is worth considering (6-7) and (6-8).

(6-7) The man ate the apple.

(6-8) The apple was eaten by the man.

During the analysis of (6-7), SYN will send the suggestion `(man.1 ate.2 subj)` after "ate" is analyzed. During secondary marker passing for this suggestion, a three-way collision occurs at the relation node `eater-eat`. This suggestion is issued by the action (sug NP1 V1 subj) of the rule **main-verb2**. NP1 is put before V1 because NP1 appears before V1 in the input string. What should be the suggestion action of the rule **passive-by** which analyzes "the man" as the actual subject of "eaten" in sentence (6-8)?

This rule is shown below:

```
(passive-by ((1 np-pp np-pp1)(h ))
  [ ( [ (np-1 s-1) 1 () ]
      [ (aux-1 s-1) 1 (passive) ]
      [ (vp-1 s-1) 1 () ]
      [ (pp-1)  0 () ] )
    ( [equal-exact (symbol by) (prep-of-pp pp-1)] )
  ]
  [ (sem connect (v vp-1) (np pp-1) subj 2)
    (attach pp-1 to vp-1)
  ]
)
```

In the suggestion of this rule, "(sem connect (v vp-1) (np pp-1) subj 2)", the flag, 2, is used to signal to SEM that the V appeared before the NP in the input string but it should be assumed that it appeared after the NP. Then SEM uses this information to change the order of the V and NP and the analysis is done correctly.

6.2.3 Heuristics for Finding the Best Path

After the secondary marker passing session for a suggestion, a set of three-way collisions is reported. The number of three-way collisions in this set is small compared with the number of two-way collisions, but there is no guarantee that only one three-way collision is returned. Even if there is only one three-way collision reported, the best paths should be found (note that one collision can represent more than one path). Thus it is necessary to develop heuristics that can find the best paths, given a set of three-way collisions. The course of finding the best path is shown in Figure 6.12.

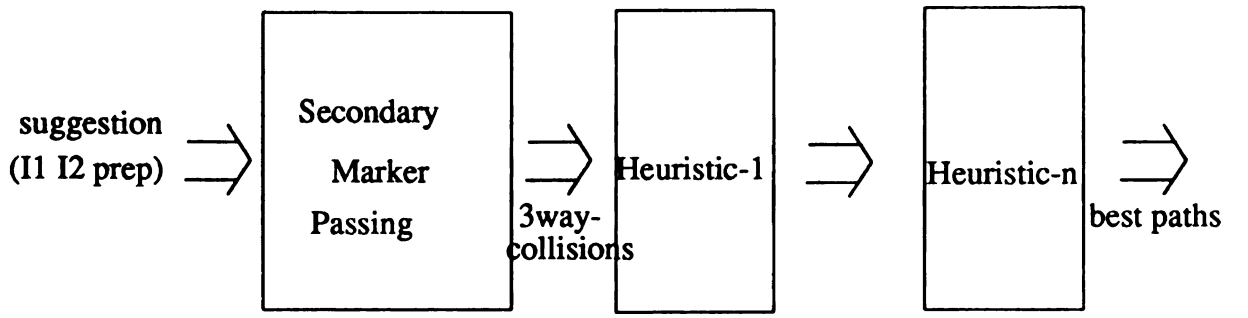


Figure 6.12 Stages of Finding Best Paths

Note that the scheme in this figure applies only to the suggestions containing a preposition. Each heuristic in Figure 6.12 is actually for reducing the number of paths(or collisions) which are the candidates for the best path.

6.2.3.1 Hierarchy: H-heuristic

The first heuristic that is in the order is called the H-heuristic (H stands for *hierarchy*). The input to this heuristic should be the set of three-way collisions which is the output from the secondary marker passing session. The idea behind the H-heuristic can be illustrated using Figure 6.13.

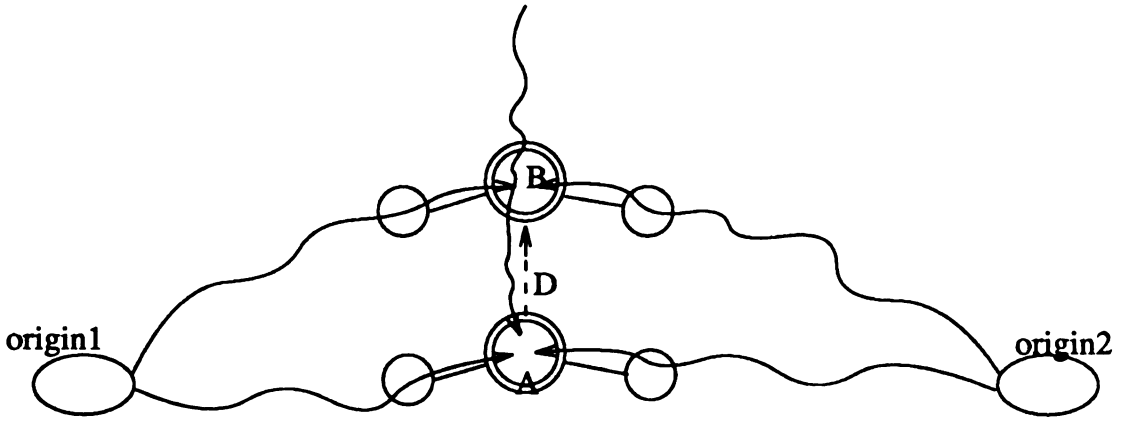


Figure 6.13 Removal of a Collision Using Ancestor Information

Three-way collisions occur at node A and B. B is the ancestor of A. This means that A can be reached from B via D links (in the reverse direction). It is obvious that the path for the three-way collision at A is better than the path for the collision at B because A is a more specific concept than B. The H-heuristic is defined precisely as follows:

> **H-heuristic**: A three-way collision is removed if its collision node is an ancestor of a node at which a three-way collision also occurs.

The result after applying the H-heuristic becomes the input to the next stage of heuristics.

6.2.3.2 Length: L-heuristic

The heuristic called the L-heuristic is applied to the result after applying the H-heuristic. Thus the input to this stage is a set of three-way collisions. The set is converted into a set of the KB paths according to the following scheme: For each three-way collision,

(i) Starting from the collision node, trace the marker back to the origin node. This results in a set of paths.

(ii) Starting from the collision node, trace the marker back to the other origin node and get the set of paths.

(iii) For each path from the set in (i) and for each path from the set in (ii), form a full path by combining the two paths. The result of this step is the set of full paths.

To the output of step (iii) above, the L-heuristic(L stands for length) is applied. This heuristic is defined as follows:

L-heuristic: Select the paths with the smallest number of relation nodes.

The idea behind this heuristic is that the shorter the path is the better the path is. The reason why the number of relation nodes is used instead of the number of all nodes should be noted. The number of absolutes and D links between the absolutes might not be a reliable source of information for filtering paths. Consider an example in Figure 6.14 to illustrate this. Let's assume that node A is an absolute which is as specific as the absolute node B. But the knowledge engineer put more descendants under B than under A because he happened to know the hierarchy under B was better than the hierarchy under A. Therefore the number of nodes between A(or B) and the instance node should not be used in computing the length that is used in the L-heuristic.

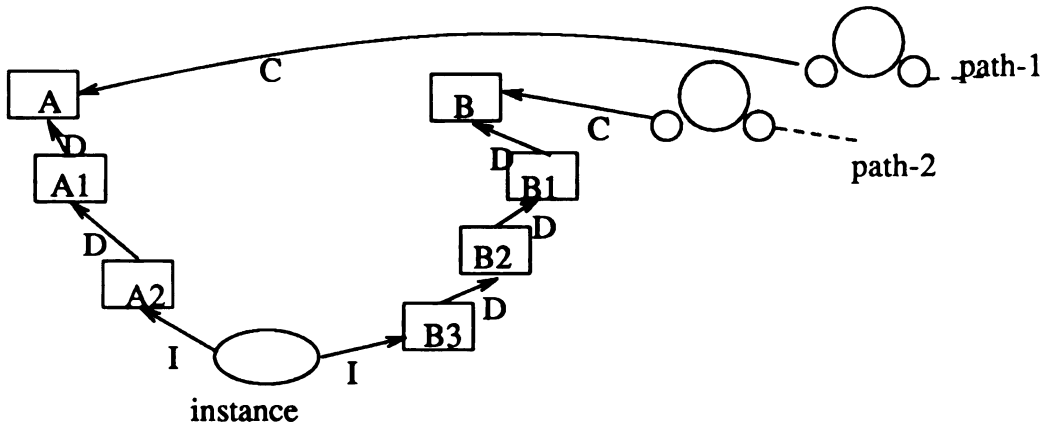


Figure 6.14 Number of Nodes in the D-hierarchy

6.2.3.3 Specificity: S-heuristic

The heuristic called the S-heuristic is applied to the set of paths output from the stage of the L-heuristic. The source of information for the S-heuristic is the specificity of the absolutes of the path (S stands for the specificity). To compare two paths based on the specificity, it is necessary to decide which absolutes(or relations) of the paths should be used for comparison. Note that a path might have many absolutes and up to three relations (because the initial strength of a marker is 6). There are three possible forms of paths as shown in Figure 6.15.

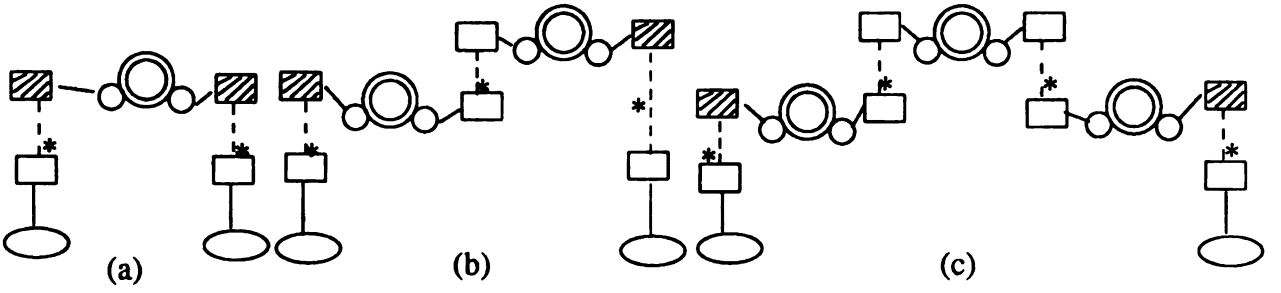


Figure 6.15 Possible Forms of Paths

A path with any of the three forms might be compared with another path with any of three forms. The nodes that are used for the comparison should be chosen in a way that does not lose any information. In the S-heuristic, two absolute nodes are used for this purpose: the absolute node that is pointed to by the first C(constraint) link and the

absolute node that is pointed to by the last C link (the dark nodes in the figure). These two absolute nodes are called the *end absolutes* of a path. The comparison of any two paths (say, path-a and path-b) is depicted in Figure 6.16. Note, in this figure, that `instance1` of path-a is the same concept as `instance1` of path-b. Similarly, `instance2` of path-a is the same as `instance2` of path-b. The reason for this is that path-a and path-b are the paths for the same suggestion, `(instance1 instance2 preposition)`.

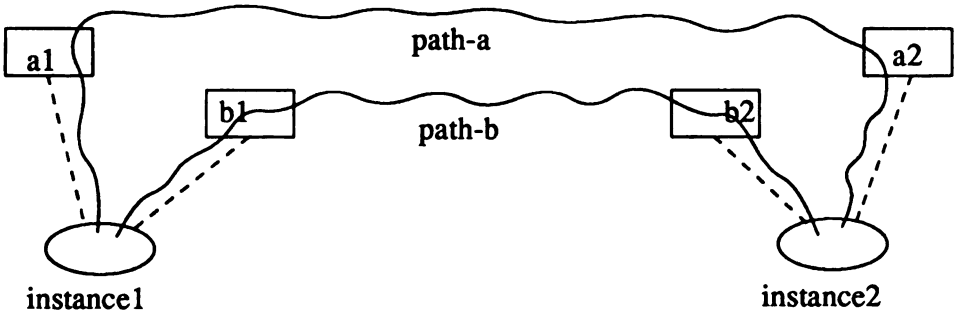


Figure 6.16 End Absolutes of two Paths to be Compared

The first end absolute of path-a, a1, is compared with the first end absolute of path-b, b1, and the second end absolute of path-a, a2, is compared with the second end absolute of path-b, b2. Now, the S-heuristic can be defined as follows (using the above notation) :

> **S-heuristic**: If a1 is an ancestor of b1 and a2 is an ancestor of(or same with) b2, then path-b is better than path-a and thus path-a is removed. Symmetrically, if a2 is an ancestor of b2 and a1 is an ancestor of(or same with) b1, then path-b is better than path-a and thus path-a is removed.

Any path which is inferior to any other path based on this heuristic is removed.

6.2.4 Operating Example of Finding the Best Path

In this subsection, some examples for finding the best path will be given. Let's consider how the path finding task is done for the suggestions sent by SYN during the analysis of sentence (6-1). The syntactic rule main-verb2 sends the suggestion (`man.1`

ate.2 subj) to SEM after "ate" is analyzed. Let's use the small KB shown in Figure 6.5 to make the explanation easy. There are four two-way collisions of primary markers from the origins, man.1 and ate.2. The collisions correspond to the following six paths (nodes at which a two-way collision occurred are underlined. Links and aspectuals are omitted):

(6-9) (man.1 man person high_animate animal animate actor-act act trsact eat ate.2)

(6-10) (man.1 man person high_animate animal animate phy_obj food eatee_eat eat ate.2)

(6-11) (man.1 man person high_animate animal animate phy_obj inanimate eat_tool instr-eat eat ate.2)

(6-12) (man.1 man person high_animate animal animate phy_obj inanimate instr-act act trsact eat ate.2)

(6-13) (man.1 man person high_animate animal animate phy_obj thing actee-trsact trsact eat ate.2)

(6-14) (man.1 man person high_animate animal eater-eat eat ate.2)

SEM passes secondary markers starting from subj. Subj, actor-act, and eater-eat get marked during this secondary marker passing session. Then two three-way collisions at actor-act and eater-eat are reported. The paths corresponding to these three-way collisions are (6-9) and (6-14) and they are shown below again with the three-way collision node underlined:

(6-9) (man.1 man person high_animate animal animate actor-act act trsact eat ate.2)

(6-14) (man.1 man person high_animate animal eater-eat eat ate.2)

Note that the collision node has been changed from `act` to `actor-act` in (6-9). The reason is that a *follow-on* collision collected during primary marker passing can also be used to form a three-way collision. Note that SEM needs to consider only two paths after secondary marker passing. Otherwise the six paths from (6-9) to (6-14) should have been considered. Then the H-heuristic is applied to the paths (6-9) and (6-14). Because `actor-act` (the three-way collision node for (6-9)) is an ancestor of `eater-eat` (the three-way collision node of (6-14)), the path (6-9) is removed. So only one path, (6-14), is left after the stage of the H-heuristic, which means that the path finding operation need not go through the stages of the L-heuristic and the S-heuristic. The best path for the suggestion (`man.1 ate.2 subj`) has been determined to be the path (6-14).

After "the cake" has been analyzed in sentence (6-1), the **object** rule sends the suggestion `(ate.2 cake.3 obj)` to SEM. The two-way collisions (actually the corresponding paths) between `ate.2` and `cake.3` are as follows:

(cake.3 cake food eatee-eat eat ate.2)
(cake.3 cake has-part1 frosting food eatee-eat eat ate.2
(cake.3 cake food psy_object animate animal eater-eat eat ate.2)
(cake.3 cake food psy_object animate actor-act act trsact eat ate.2)
(cake cake food psy_object inanimate eat_tool instr-eat eat ate.2)
(cake.3 cake food psy_object inanimate instr-act act trsact eat ate.2)
(cake.3 cake has-part1 frosting food phy_object animate animal eater-eat eat ate.2)
(cake.3 cake has-part1 frosting food phy_object animate actor-act act trsact eat ate.2)
(cake.3 cake has-part1 frosting food phy_object inanimate eat_tool instr-eat eat ate.2)
(cake.3 cake has-part1 frosting food phy_object inanimate instr-act act trsact eat ate.2)
(cake.3 cake food phy_object thing actee-trsact trsact eat ate.2)
(cake.3 cake has-part1 frosting food phy_object thing actee-trsact trsact eat ate.2)

Instead of considering these 12 two-way collision paths, secondary marker passing is done starting from the node `obj`. `Actee-trsact` and `eatee-eat` get secondary markers and three-way collisions occur at these two relation nodes. The three-way collisions and the corresponding paths are as follows (the three-way collision nodes are underlined):

(6-15) (cake.3 cake food psy_object thing actee-trsact trsact eat ate.2)

(6-16) (cake.3 cake has-part1 frosting food phy_object thing actee-trsact trsact eat ate.2)

(6-17) (cake.3 cake food eatee-eat eat ate.2)

The H-heuristic is applied to the two three-way collisions. Because actee-trsact is ancestor of eatee-eat, the paths for the three-way collision at actee-act, i.e. (6-15) and (6-16), are removed. The output of the stage of the H-heuristic is only one three-way collision and this collision corresponds to only one path, (6-17). Thus the best path is determined to be (6-17). The following stages of the heuristics need not be applied because only one path is left.

Let's consider the other situation during the analysis of the sentence (6-1). After the PP, "with a fork", has been analyzed, two rules(vp-pp and np-pp) are found by the pattern matcher. Each of these two rules creates a branch of parsing. The branch for vp-pp issues the suggestion (ate.2 fork.4 with) and the branch for np-pp issues the suggestion (cake.3 fork.3 with). The best paths for these two suggestions are found in a similar way. For these two collisions, the data will be provided using the bigger knowledge base below.

So far we used the KB shown in Figure 6.5. But this KB is very small. The following data was obtained using a bigger KB which has about 100 concepts. The rules and their suggestions are from the analyses of the same sentence (6-1).

- Suggestion (man.2 ate.2 subj) for the main-verb2 rule :

  Number of paths corresponding to two-way collisions : 295

  Number of three-way collisions after secondary marker passing : 2

  Number of three-way collisions after applying H-heuristic : 1

  Best path found:

  (6-18) (<abso ate.2> I <abso eat> C <asp eat.eater-eat> A <rel eater-eat> A <asp

eater.eater-eat> C <abso animal> D <abso high_animate> D <abso person> D <abso man> I <abso man.1>)

Note in the above data that the number of paths for the two-way collisions has been increased from 6 to 295 according to the change of the KB. But the result of secondary marker passing is still 2. This shows the good performance of the proposed mechanism. This good performance is also observed in the case of the other suggestions shown below. The L-heuristic and the S-heuristic need not even be applied because only one path is found after applying the H-heuristic. (According to the experiments, it has been found that the L-heuristic and the S-heuristic are usually used when there is word sense ambiguity.)

- Suggestion (`ate.2 cake.2 obj`) for the **object** rule :

  Number of paths corresponding to two-way collisions : 220

  Number of three-way collisions after secondary marker passing : 2

  Number of three-way collisions after applying H-heuristic : 1

  Best path found:

  (6-19) (<abso cake.3> I <abso cake> D <abso food> C <asp eatee.eatee-eat> A <rel eatee-eat> A <asp eat.eatee-eat> C <abso eat> I <abso ate.2>)

- Suggestion (`ate.2 fork.4 with`) for the rule **vp-pp** :

  Number of paths corresponding to two-way collisions : 112

  Number of three-way collisions after secondary marker passing : 2

  Number of three-way collisions after applying H-heuristic : 1

  Best path found:

  (6-20) (<abso fork.4> I <abso fork> D <abso eat_tool> C <asp instr.instr-eat> A <rel instr-eat> A <asp eat.instr-eat> C <abso eat> I <abso ate.2>)

- Suggestion (`cake.3 fork.4 with`) for the rule **np-pp** :

  Number of paths corresponding to two-way collisions : 808

  Number of three-way collisions after secondary marker passing : 1

  Best path found:

  (6-21) (<abso fork.4> I <abso fork> D <abso eat_tool> D <abso inanimate> D <abso phy_object> C <asp part.has-part> A <rel has-part> A <asp whole.has-part> C <abso phy_object> D <abso food> D <abso cake> I <abso cake.3>)

In the above data, it is shown how effective the secondary marker passing mechanism is for the path finding problem. This can be contrasted with the case where secondary marker passing can not be used. When "the granite rocks" is analyzed as a nominal compound, a suggestion (`granite.1 rocks.2`) is sent to SEM. Note that this suggestion has no preposition in the suggestion because there is no syntactic cue about the relation between the nouns in the nominal compound. The semantic relation between nouns should be inferred by the semantic processing component (Finin, 1980). Data from the path finding operation for the suggestion (`granite.1 rocks.2`) of the **noun-parse3** rule is:

Number of paths corresponding to two-way collisions = 309

Number of paths after applying S-heuristic : 29

Best path found:

(6-22) (<abso rocks.2> I <abso stone> D <abso granite> I <abso granite.1>)

Secondary marker passing could not be used because there is no preposition available. After the stage of the S-heuristic, the number of paths is 29 which is still a large number of paths. The time taken for the stage of the S-heuristic in this case is about 50 times longer than the time taken for the whole path finding operation for the suggestions with prepositions. This data shows how efficient the secondary marker passing mechanism is for the path finding problem.

6.3 Semantic Comparison of Branches

It has been explained in this chapter that the branches running in parallel are filtered at the semantic comparison stage(SCS). The comparison is made based upon the goodness of semantic processing results. The semantic processing results are represented by the KB paths returned by SEM for the suggestions from SYN. The KB paths are stored in the corresponding branches so that they are used in the semantic comparison stage. Thus the problem of comparing the branches based on the semantic processing results can be reduced to the problem of comparing the paths of the branches.

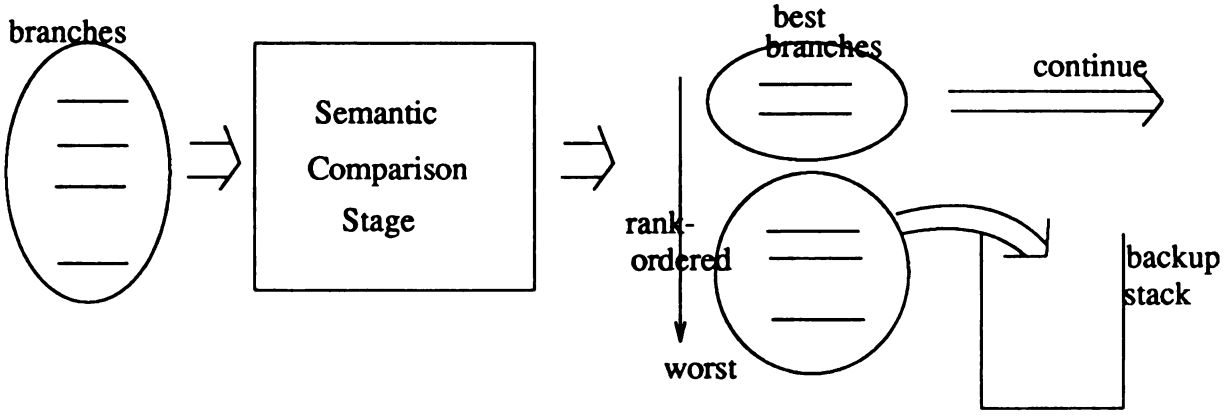


Figure 6.17 Rank-ordering the Branches

For the moment, let's assume that any two paths can be compared and the decision can be made about which path is the better path (or equally good path). Then the branches that are input into the semantic comparison stage can be rank-ordered. The branches with the highest rank are kept as active (see Figure 6.17) and the other branches are pushed into the backup stack, the worse one first so that the branch with better rank can be used first in case of backtracking. Only one branch will normally be given the highest rank. But there can be more than one branch which receives the highest rank because two branches may have the same paths as a result of semantic processing. The branches with no semantic processing result are always inserted in the set of branches having the highest rank for the following reason: these branches can not be compared to the branches with semantic processing results and should be kept as active branches. Having

no semantic processing result means that the corresponding rule has no action issuing a suggestion. Examples of these cases will be shown later.

One simple example can be given using the analysis of sentence (6-1). It was explained that the np-pp rule and the vp-pp rule were found during the same pattern matching operation. Two branches are created for the two rules. For the branch of the np-pp rule, the semantic processing result is the path (6-21) that is found for the suggestion `(cake.3 fork.4 with)`. The path (6-20) found for the suggestion `(ate.2 fork.4 with)` is the semantic processing result for the branch of the vp-pp rule. Figure 6.18 illustrates this case.

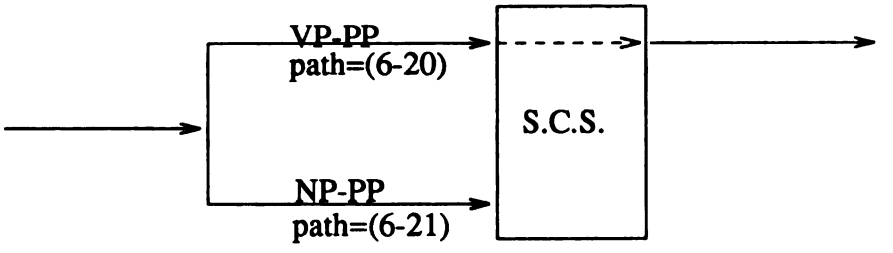


Figure 6.18 Comparison of Two Rules(vp-pp & np-pp)

It has been stated that rank-ordering the branches during the semantic comparison stage can be done if any two paths can be compared. We have seen in section 6.2.3 that the process of finding the best path for a suggestion also requires the capability of comparing two paths and the paths to be compared have the same origins as illustrated in Figure 6.16. But it is necessary to consider one more possibility in the case of the path comparison problem in this section. This is illustrated in Figure 6.19.

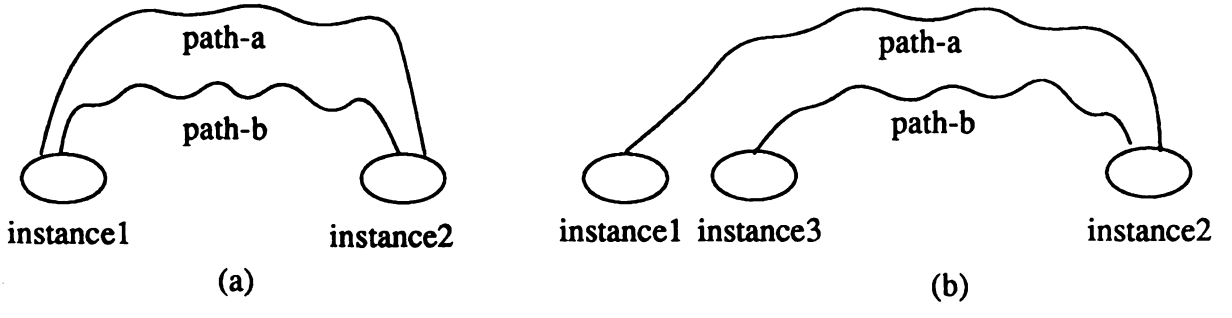


Figure 6.19 Instances and Two Paths to be Compared

In Figure 6.19(b), path-a and path-b share one end (i.e. `instance2`) but not the other end. The reason that two paths should share at least one end(origin) is that the newest constituent built by the parser is always involved in the suggestions issued by the rules that are found during the same pattern matching. This fact makes it easy to decide which end of a path corresponds to which end of the other path.

The comparison of path-a and path-b can be classified into four possible cases (note that a1 and a2 are the end absolutes of path-a, and b1 and b2 are the end absolutes of path-b) as shown in the following heuristic.

SCS Heuristic:

(case-i): The relation of end absolutes conforms to the condition of the S-heuristic explained in section 6.2.3.3. Let's state it here again. If a1 is an ancestor of b1 and a2 is an ancestor of(or same with) b2, then path-b is better than path-a. Symmetrically, if a2 is an ancestor of b2 and a1 is an ancestor of(or same with) b1, then path-b is better than path-a.

(case-ii): If a1 is an ancestor of b1 but a2 and b2 have no ancestor relationship, then consider the following three possibilities, (a), (b) and (c). Here we need to consider what the term *promiscuous* concept is. A concept is said to be promiscuous if it is high in the hierarchy of the KB and has many descendants(Charniak, 1983a). For example, "action", "thing", "physical-object", etc. are promiscuous concepts.

(a) If a2 is promiscuous, then path-b is better than path-a.

(b) If neither a2 nor b2 is promiscuous, then path-b is better than path-a.

(c) If b2 is promiscuous but not a2, then it is undecidable.

(case-iii): If a1 is the same as b1 and both a2 and b2 are promiscuous concepts(or symmetrically, a2 is the same with b2 and both a1 and b1 are

promiscuous), path-a and path-b ties in goodness. In this case, any path with a smaller number of relation nodes is better than the other. If the number of relation nodes can not even break the tie, other information such as the frequency of usage of relation nodes in the path is used to decide which path is better.

(case-iv): If a1 is not b1 and they do not have an ancestor relationship, and a2 and b2 do not have an ancestor relationship, then any path having a smaller number of promiscuous end absolutes is better than the other.

In Figure 6.18, the path (6-20) for the vp-pp rule was chosen over the path (6-21), which is determined using (case-ii) of the SCS heuristic. The end absolutes of (6-20) are `eat` and `eat_tool`, and those of (6-21) are `phy_object` and `phy_object`. Consider Figure 6.20 for this problem. Because `phy_object` is an ancestor of `eat_tool` but `phy_object` and `eat` have no ancestor relationship, (case-ii) can be applied and the decision that (6-20) is better than (6-21) can be made. Thus the branch for the vp-pp rule is chosen in the SCS.

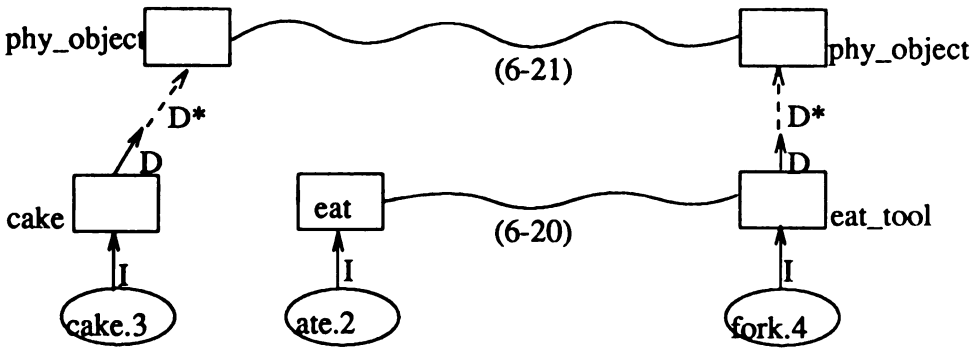


Figure 6.20 Two Paths in "eat" Example

6.4 Semantic Processing Component in Action

In this section, two examples of semantic processing will be given to provide a clearer picture of the actual working of the SYNSEM parser. Especially, in the second

subsection, it will be shown in detail how the parallel running of branches is influenced by semantic processing.

6.4.1 Example 1

Let's first consider the example given in Section 4.6 which used the example sentence (4-21) shown below again:

The teachers taught by the pool passed the test.

It was explained that three rules are selected during the pattern matching operation as shown in Figure 4.19: main-verb2, np-vpp, and np-vpp1. For each of these rules, a branch is created. During the execution of the actions of each branch, a suggestion is issued as shown in Table 6.1. The data related to finding the best path for each suggestion is shown in the table, too.

After executing the rules of the branches, the semantic comparison stage comes because each branch needs a new input word. The semantic processing result of the branch corresponding to the rule main-verb2 is (6-23). Similarly, the path (6-24) is the semantic processing result of np-vpp, and (6-25) is for the branch of np-vpp1. Thus these three paths are compared at the semantic comparison stage. Note that the end absolutes for (6-23) are `teach` and `teach_professional`, those for (6-24) are `trsact` and `thing`, and those for (6-25) are `teach` and `person`. Each pair of paths is compared as follows:

(i) Comparison of (6-23) and (6-24):

The first EA of (6-24) is an ancestor of the first EA of (6-23) and the second EA of (6-24) is also an ancestor of the second EA of (6-23). Therefore, according to (case-i) of the SCS heuristic, (6-23) is better than (6-24).

(ii) Comparison of (6-23) and (6-25):

The first EA of (6-23) is the same as the first EA of (6-25) and the second EA

Table 6.1 Data for Finding Paths

Branch	main-verb2	np-vpp	np-vpp1
Suggestion	(teachers.1 taught.2 subj)	(teachers.1 taught.2 obj)	(teachers.1 taught.2 to)
# of 2-way collisions	398	398	398
# of 3-way collisions	2	1	1
H-heuristic	1	1	1
S-heuristic	1	1	1
L-heuristic	1	1	1
path	(6-23)	(6-24)	(6-25)

(6-23)(<abso taught.2> I <abso teach> C <asp teach.teacher-teach> A <rel teacher-teach> A <asp teacher.teacher-teach> P <abso teach_professional> D <abso teacher> I <abso teachers.1>)

(6-24)(<abso taught.2> I <abso teach> D <abso trsact> C <asp trsact.actee-trsact> A <rel actee-trsact> A <asp actee.actee-trsact> C <abso thing> D <abso phy_object> D <abso animate> D <abso animal> D <abso high_animate> D <abso person> D <abso teach_professional> D <abso teacher> I <abso teachers.1>)

(6-25)(<abso taught.2> I <abso teach> C <asp teach.rec-teach> A <rel rec-teach> A <asp rec.rec-teach> C <abso person> D <abso teach_professional> D <abso teacher> I <abso teachers.1>)

of (6-25) is an ancestor of the second EA of (6-23). Thus, according to (case-i) of the SCS heuristic, (6-23) is better than (6-25).

(iii) Comparison of (6-24) and (6-25):

The first EA of (6-24) is an ancestor of the first EA of (6-25), and the second EA of (6-24) is also an ancestor of the second EA of (6-25). Thus, according to (case-i) of the SCS heuristic, (6-25) is better than (6-24).

From (i), (ii) and (iii), it is concluded that (6-23) is the best, (6-25) is the next best, and (6-24) is the worst. Therefore the branch for main-verb2 is the one that is selected by the semantic comparison stage. The other two branches are pushed into the backup stack. The branch for np-vpp1 is pushed later than that for np-vpp so that the branch with the better semantic result is used first when backtrack is required later. This situation is shown in Figure 6.21.

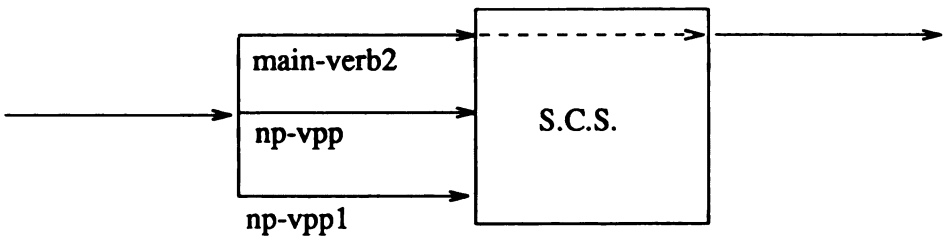


Figure 6.21 Competition of Three Branches

Consider the situation shown in Figure 4.18(w). It was explained that the pattern matcher found the passive-by rule and the vp-pp rule that matched Figure 4.18(w). Thus, for each of the two rules, a branch is created. These branches, the corresponding suggestions, and the paths for the suggestions are shown in Figure 6.22.

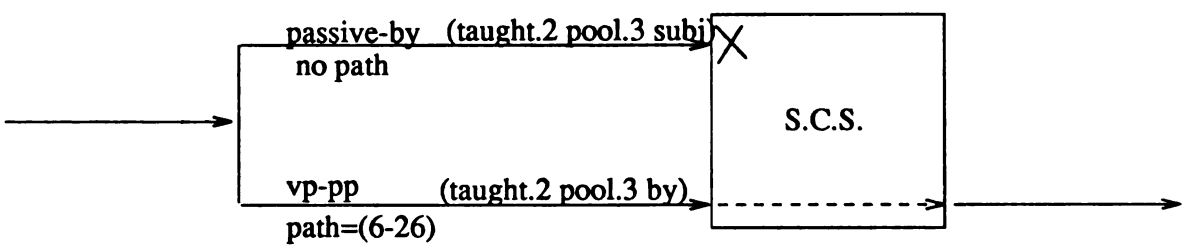


Figure 6.22 Removal of a Branch with Bad Semantic Processing

(6-26)(pool.3 I pool D place C loc.loc-teach A loc-teach A teach.loc-teach C teach I taught.2)

Note that no path can be found for the suggestion for the passive-by rule. Therefore the branch corresponding to this rule is just thrown away because the semantic effect is too bad to find a reasonable path. But for the suggestion for the vp-pp rule, the path (6-26) is found. Thus the branch for the vp-pp rule is the only rule that is left and it is kept active without any comparison as shown in Figure 6.22.

6.4.2 Example 2

Let's consider how sentence (6-27) is processed in SYNSEM.

(6-27)The granite rocks near the shore.

The snapshot of the SWM after reading "rocks" is shown in Figure 6.23. The root of the tree that corresponds

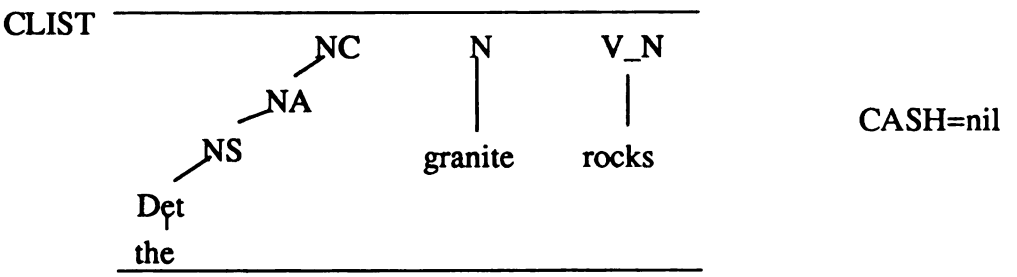


Figure 6.23 Snapshot of CLIST

to the word "rocks" is of two types(V or N), because "rocks" can be either a verb or a noun. Three rules match this SWM. They are noun-parse3, noun-parse4 and noun-parse5. Noun-parse4 analyzes "rocks" as the head noun of the NP that corresponds to the input string "the granite rocks". This rule matches the SWM because "rocks" is a plural noun. If it is not a plural noun, it is necessary to see the next word to make the decision. The rule noun-parse3 regards "rocks" as a verb and analyzes "the granite" as the complete NP. noun-parse5 also analyzes "the granite" as the complete NP but interprets "rocks" as a noun that leads a relative clause (i.e. "rocks" is considered to be the NP which is the subject of the relative clause that will follow).

Each of the three rules creates a branch of parsing. The system takes a note that CASH was empty at this time. From this point to the next semantic comparison stage, each of the branches will run independently. Note that the semantic comparison stage comes when all branches want to read the next word (just after "rocks"). Let's follow each branch until the semantic comparison stage comes. First, the flow of the branch for noun-parse3 is shown in Figure 6.24. After the execution of noun-parse3, the SWM becomes (a) in the figure. But there is no suggestion issued by this rule. The V of

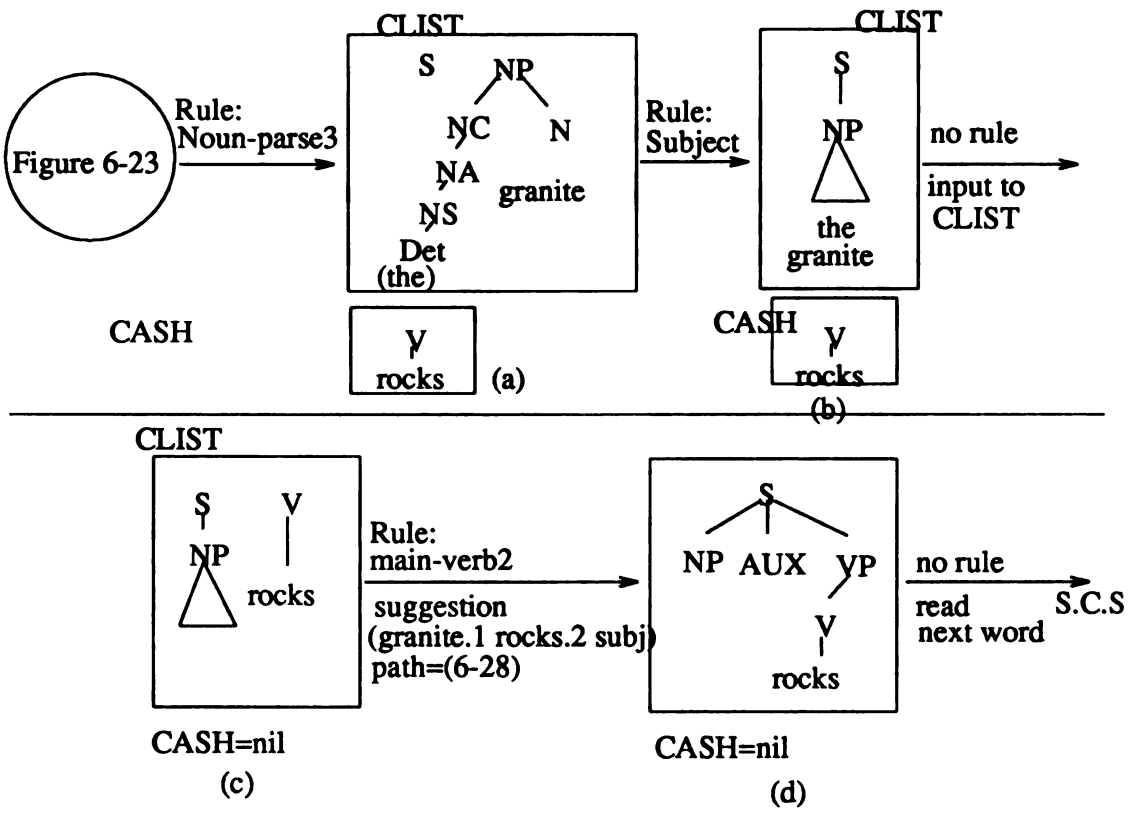


Figure 6.24 Flow of Branch for Noun-parse3

"rocks" is pushed into CASH as shown in (a). The subject rule matches this SWM and its execution results in the SWM in (b). No rule matches (b) and CLIST needs an input. Because CASH is not empty, the V of "rocks" in CASH is inserted into CLIST as shown in (c). The main-verb2 rule matches this SWM, and its execution updates (c) to (d). This main-verb2 rule issues a suggestion `(granite.1 rocks.2 subj)`. The best path for this suggestion is found by SEM and is shown in (6-28).

(6-28)(<abso rocks.2> I <abso shake> C <asp act.stuff-shake> A <rel stuff-shake> A <asp stuff.stuff-shake> C <abso phy_object> D <abso inanimate> D <abso stone> D <abso granite> I <abso granite.1>)

Against the SWM in (d), no rule matches and a new input is required to be inserted into CLIST. Because CASH is empty, a new word needs to be read from the input string which means that this branch has reached the semantic comparison stage.

Second, the flow of the branch for the rule noun-parse4 is shown in Figure 6.25. After the execution of noun-parse4, the SWM of this branch becomes (a) in the figure. The rule noun-parse33 matches (a) whose execution results in the SWM in (b). This rule notices that "rocks" is in plural form of the noun and thus it determines that "rocks" is the head noun of the NP. Because this NP is a nominal compound, a suggestion `(granite.1 rocks.2)` is sent to the SEM. Note that a preposition is missing in this suggestion.

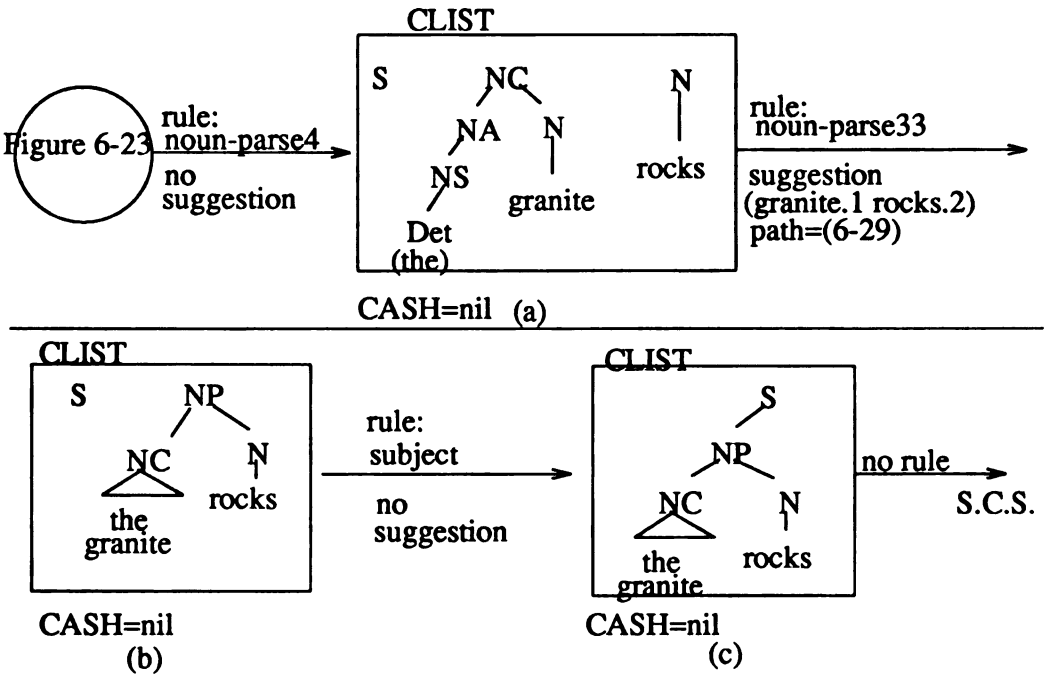


Figure 6.25 Flow of Branch for Noun-parse4

From the nominal compound, no information about the related preposition can be obtained by SYN. SEM finds the path (6-29) for this suggestion.

(6-29)(<abso rocks.2> I <abso stone> D <abso granite> I <abso granite.1>)

The parsing of this branch continues and the subject rule matches the SWM in (b). The execution of this rule results in the SWM shown in (c). No rule is found for (c). Because CASH is empty, a new input word should be read from the input string. Therefore this branch has reached the semantic comparison stage.

Third, let's follow the branch for the noun-parse5 rule. The flow of this branch is shown in Figure 6.26. After the execution of

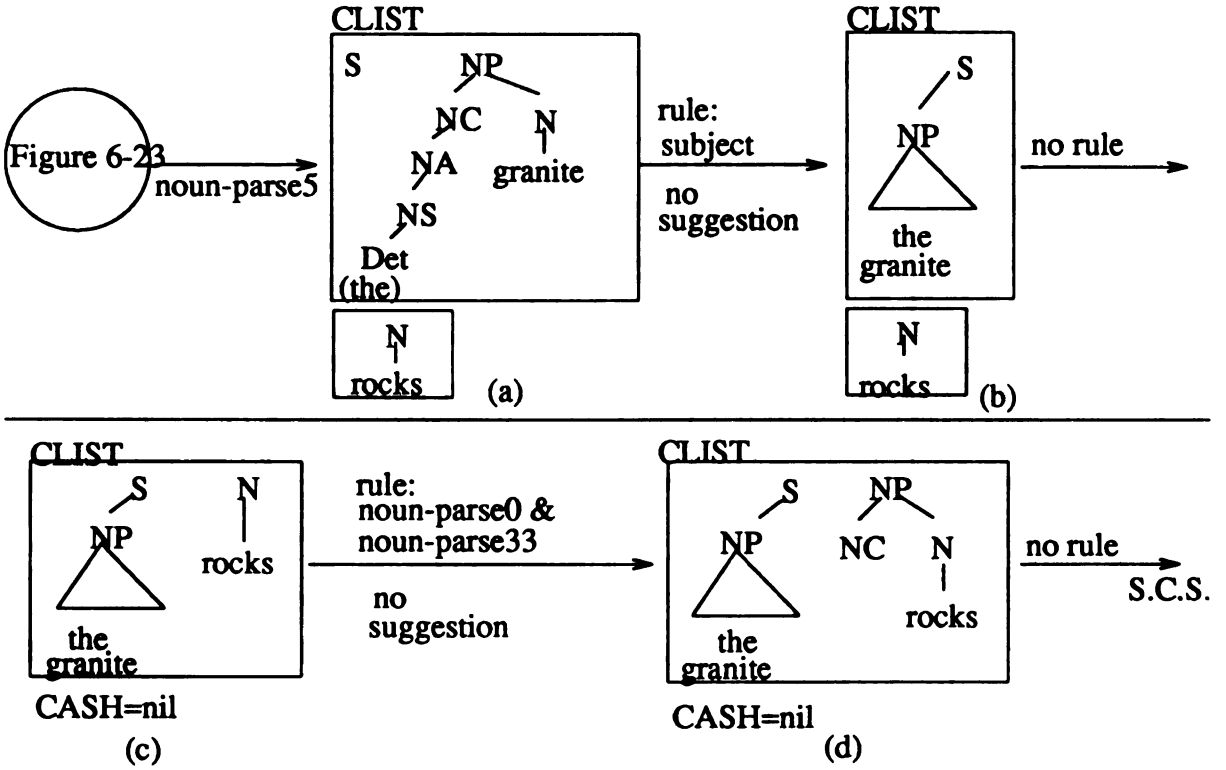


Figure 6.26 Flow of Branch for Noun-parse5

noun-parse5, the SWM of this branch becomes that in (a) of the figure. This rule regards "rocks" as the plural noun which starts a new NP and thus "granite" is analyzed as the head noun of the complete NP as shown in (a). Note that "rocks" is pushed into CASH so that the NP, "the granite", can be used in any further processing while it is the rightmost edge in CLIST. The subject rule matches (a) and its execution results in the SWM in (b). No rule matches (b) and thus a node in CASH is input into CLIST as shown in (c). The SWM in (c) becomes (d) by the noun-parse0 rule and noun-parse33 rule. These two rules determine that "rocks" is the NP by itself. No rule is found that matches (d) and CASH is empty. Thus a new word needs to be input from the input string. This branch has reached the semantic comparison stage. Note that no rule issued a suggestion during the parse of this branch. This is an interesting case since there is no semantic processing result for this branch.

At the semantic comparison stage, these three branches are compared based on the semantic processing results. The semantic processing result of the branch for noun-parse3 is the path (6-28), while the path (6-29) is the semantic processing result of the branch for the rule noun-parse4. But the branch for the rule noun-parse5 has no semantic processing result. This situation is illustrated in Figure 6.27.

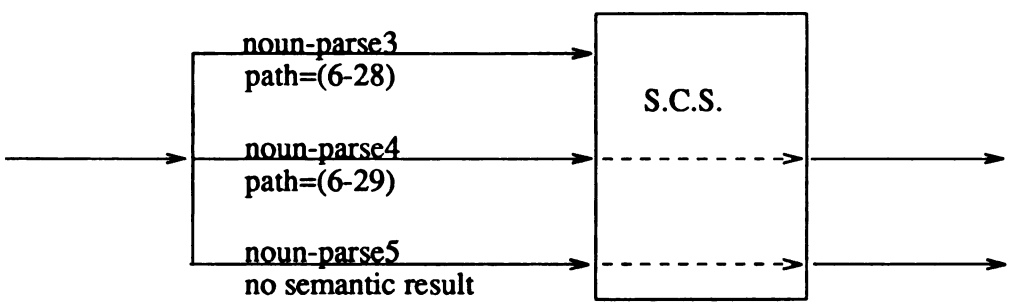


Figure 6.27 Comparison of Three Branches

The path (6-28) will be compared with the path (6-29) to compare the two branches. Note that there is no relation node in the path (6-28). The paths that have no relation generally have the form shown in Figure 6.28.

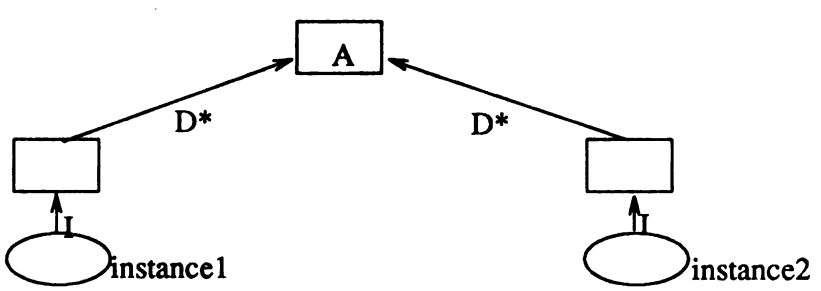


Figure 6.28 Path without a Relation Node

In this case, we set both the first end absolute and the second end absolute to be the plateau (the node A in the figure). Therefore the end absolutes of (6-29) are the concept `stone`. The end absolutes of (6-28) are `shake` and `phy_object`. Because the first end absolutes of the two paths have no ancestor relationship but the second end absolute, `phy_object`, of (6-28) is an ancestor of the second end absolute, `stone`, of (6-29), (6-29) is better than (6-28) according to (case ii-b) of the SCS heuristic. Because the

branch for noun-parse5 has no semantic processing result collected until the semantic comparison stage, it can not be compared with other branches and thus it is kept active by the semantic comparison stage. As shown in Figure 6.27, the branch for noun-parse4 and that for noun-parse5 are selected in the semantic comparison stage. The branch for noun-parse3 is pushed into the backup stack.

After the semantic comparison stage, two branches are active and run in parallel as shown in Figure 6.29 (branch-1 is for noun-parse4 and branch-2 is for noun-parse5). The new input word("near") is input by MON and is inserted into CLIST of every active branch. The SWM at this point is shown in Figure 6.29(a). No rule is found by the pattern matcher in either branch in (a). Thus, a new input word is required to be input in each branch and the semantic comparison stage comes. But there is no semantic processing result collected after the last semantic comparison stage. So both branches are kept without comparison. The next input word "the" is read and pushed into CLIST of both branches as shown in (b). After the node of type "NS" is made, both branches need to read another new input word. The semantic comparison stage comes but both branches are kept because neither has any semantic processing result collected since the last stage. After reading "shore", the SWM's of the branches are shown in (c). After the PP processing is done, branches are shown in (d). Against (d) in the figure, the np-pp rule matches the SWM's of both branches. The semantic suggestion is `(rocks.2 shore.3 near)` and the best path is found to be (6-30):

(6-30)(<abso shore.3> I <abso shore> D <abso place> C <asp pos.loc-place> A <rel loc-place> A <asp pre.loc-place> C <abso phy_object> D <abso inanimate> D <abso stone> I <abso rocks.2>)

After the punctuation mark(".") in CASH is input into CLIST's, the branches become (f). Neither branch finds any rule that matches the SWM and it is found that both branches have reached the dead-end situation. Thus, both branches are thrown away for the reason of *syntactic anomaly*. Backtracking is required because there is no active

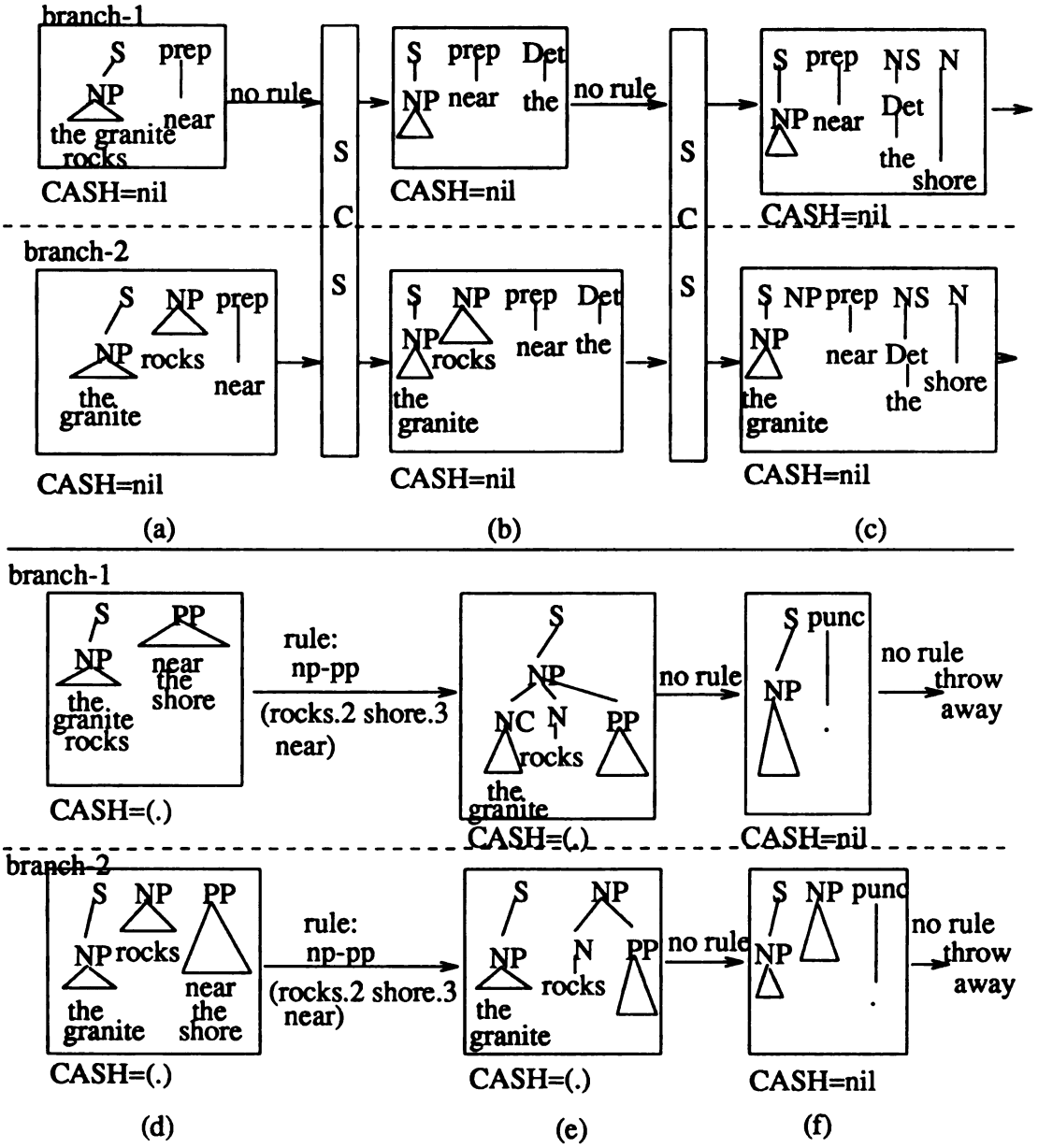


Figure 6.29 Parallel Running of Two Branches

branch. A branch is popped out from the backup stack and it is made to be an active branch. This is the branch that was shown in Figure 6.24(d). From this point, the parsing goes on as shown in Figure 6.30. Against the SWM in (b) of the figure, the vp-pp rule matches. This rule issues a suggestion `(rocks.2 shore.3 near)`. The path found for this suggestion is (6-31). Note that this suggestion is the same as the suggestion issued at Figure 6.29(d)-(e). But the path for the suggestion issued at Figure

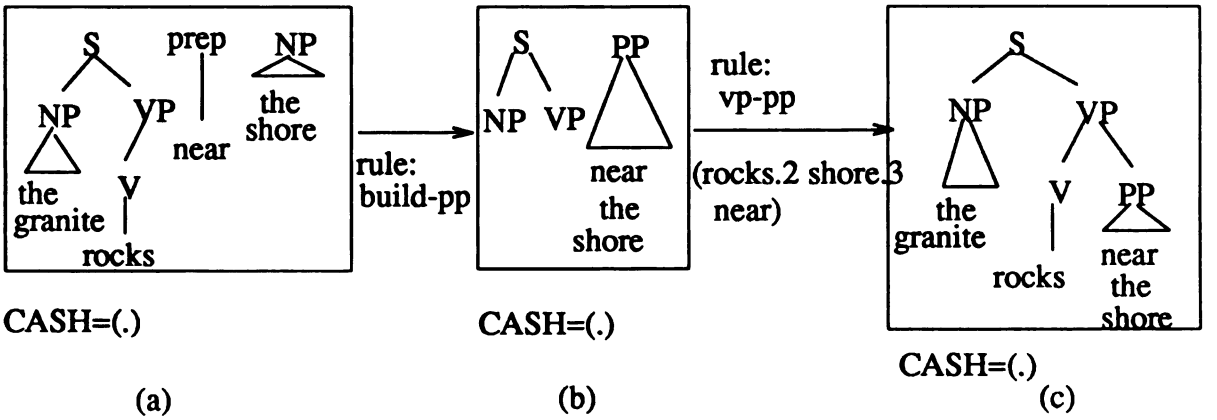


Figure 6.30 Final Branch for the Parse of the Sentence

6.29(d)-(e) is (6-30). The reason that the paths found for the same suggestions are different is that "rocks" is used as a *verb* in Figure 6.30 (thus with sense `shake`) while as a noun in Figure 6.29 (thus with sense `stone`). This shows that SYNSEM uses the correct sense that is appropriate for the part of speech used in the analysis.

(6-31)(<abso shore.3> I <abso shore> D <abso place> C <asp loc.loc-act> A <rel loc-act> A <asp act.loc-act> C <abso act> D <abso shake> I <abso rocks.2>)

6.5 Conclusion

It has been explained in this chapter how the SYNSEM parser gets the semantic information that is necessary to drive the parsing. Each syntactic rule that needs semantic processing contains actions that issue a suggestion to find a best path between two concepts. A preposition or a pseudo-preposition is provided in the suggestion whenever it is available. The semantic component should find the best path for this suggestion. This path is used to build semantic interpretation and is stored in the branch as part of its semantic processing result.

The marker passing paradigm provides a conceptually simple method to find a path between two concepts. But this simple method is plagued by the reality that there are too many spurious collisions(paths) collected during marker passing. This results in a serious problem when attempting to use the marker passing paradigm in natural language

processing. A mechanism called *secondary marker passing* has been developed to solve this problem. It has been found that the secondary marker passing method removes most of the spurious collisions efficiently. The key idea behind the secondary marker passing method is to utilize the information that can be provided by the preposition or pseudo-preposition supplied in the suggestion. Along with this mechanism, three heuristics have been developed to find only the best paths for a suggestion.

Another important task for the semantic processing component is to compare and filter the branches according to the semantic processing result at the semantic comparison stage. Each branch keeps the set of the knowledge base paths found for the suggestions issued by the rules of the branch. The problem of comparing the branches is reduced to comparing the paths. How to compare the two paths to determine the better one has been studied in this chapter. This is one way to solve the problem of scoring the alternative rules and choosing the one with the best score. Instead of producing the absolute score for the rules, the branches are rank-ordered according to the comparison of the paths and the branch with the highest rank is chosen. Using this scheme, the problem of producing an absolute score can be avoided.

CHAPTER 7

WORD SENSE DISAMBIGUATION

It is recognized that word sense disambiguation is one of the several natural language understanding problems that are hard to solve completely. Since any natural language understanding system should have some mechanism to do word sense disambiguation to achieve a reasonable degree of understanding, researchers in natural language understanding have devoted much research to this problem. It seems that we are far from achieving a complete solution to the problem. However humans seem to have no trouble in selecting the correct sense of a word in a sentence. Psycholinguists have tried to find the mechanism that humans use to achieve this high performance, but they are also far from completely understanding this mechanism.

There are three kinds of ambiguity related to a word (so-called lexical ambiguity) : homonymy, polysemy and categorical ambiguity. *Homonymous* words (for example, ball/toy-ball, ball/dance-party) have multiple meanings(senses) that are not related. *Polysemous* words (for example, "go") have several meanings that are related. The senses of a polysemous word have a meaning in common. A polysemous word is also called a vague word. A word is *categorically* ambiguous when it can be used in more than one syntactic category (for example, rock/shake, rock/stone).

There were two recent research efforts in word sense disambiguation that used knowledge bases built with *frame*-based knowledge representation. Hirst's(1984, 1986) research is largely related to the disambiguation of homonymous words. He used what he called *Polaroid Words(PW)* to realize his idea about the gradual disambiguation of ambiguous words. A PW is created and assigned to each word as it is input to the parser. PWs can be considered to be communicating processes that cooperate together.

A PW first starts with all possible senses of its word. The PW gradually eliminates the senses that are inappropriate to the surrounding context by communicating with other PWs. When a PW eliminates some of its senses, it announces this. Then other PWs use this information to try to eliminate some of their senses. This elimination and announcement process goes on until all words have been read and the exchanging of information stabilizes. Two major sources of disambiguation information that are used by PWs are selectional restriction and semantic association. The use of semantic association is achieved by incorporating marker passing in his system.

Lytinen(1984, 1988) put his focus on the disambiguation of vague words. He used the concept refinement process for this purpose. A vague word is initially represented by a concept that is high in his hierarchically organized knowledge base. As more words are read in and more contextual information is available, a set of concept refinement rules are used to refine the concept to a more specific concept in the knowledge base. But it is not clear how the homonymous words (especially nouns) are disambiguated using Lytinen's strategy.

In this chapter, we will explain a word sense disambiguation method used in the SYNSEM parser. Influenced by the work of Charniak(1983a, 1986), SYNSEM has been built based on the marker passing mechanism. SYNSEM keeps track of the semantic paths which reflect the process of semantic interpretation of an input sentence. If a word has multiple senses and it has not been disambiguated yet, there can be false paths being kept due to the inappropriate senses of the ambiguous word. If a word has a vague word sense, then it will be represented by an abstract concept until it is disambiguated to a more specific concept. As the parsing goes on, these false paths will be eliminated and abstract concepts will be replaced by concrete concepts. All these disambiguation strategies adopted by SYNSEM are made possible by the use of the knowledge base that contains the system's knowledge about the world.

7.1 Use of Semantic Paths for Word Sense Elimination

To explain several aspects of the disambiguation method used in SYNSEM, the knowledge base shown in Figure 7.1 will be used throughout this chapter.

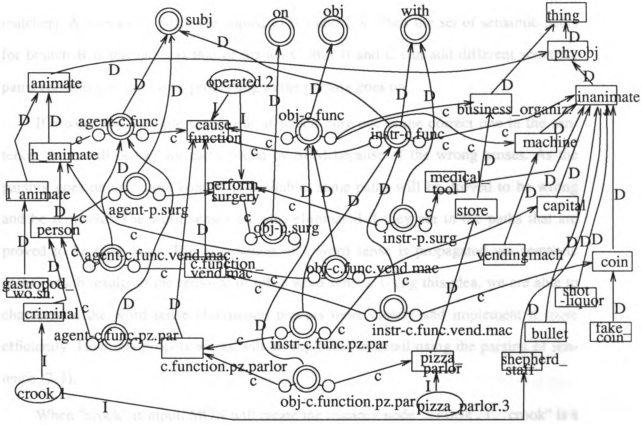


Figure 7.1 Knowledge Base for Examples

Let's use the example sentence (7-1) (Hirst, 1983) to show how the disambiguation goes on.

(7-1) The crook operated the pizza parlor.

In this section, the use of semantic paths for word sense disambiguation will be described. The mechanism is based on the word sense elimination strategy proposed by Hirst. But instead of using the complex interaction among Polaroid Words, we utilize the semantic paths that are found during the analysis of the sentence. As we saw in the previous chapter, the best path in the KB is found for each suggestion sent from SYN to SEM. The best path(or paths) for the suggestion is stored in a set in the branch of

parsing. This set represents the semantic interpretation of the sentence (in the loose sense). Each branch has its own set of paths that represent its semantic interpretation. Let's call this set of the best paths the set of *semantic paths*. When a branch, say branch A, is forked into two branches, say B and C (because of two rules found by the pattern matcher), A's semantic paths are copied to B's and C's. Then the set of semantic paths for branch B is the same as that of branch C. But B and C can add different semantic paths according to their own processing as the parsing goes on.

If a word has multiple senses out of which only one is the correct one in the sentence, there will be wrong paths found by SEM because of the wrong senses. As the parsing goes on and more context is available, some paths will be proved to be wrong and be removed. The word senses are also eliminated if they are in the paths that are proved to be false later. The elimination of a word sense is propagated via semantic paths, which results in the removal of other word senses. Using this idea, we are able to characterize the word sense elimination process more clearly and implement it more efficiently. How this process works will be explained in detail using the parsing of sentence (7-1).

When "crook" is input, MON will create the instance node `crook.1`. "crook" is a homonymous word that has two senses: criminal and a staff used by the shepherds. The node `crook.1` has I-links pointing to *all* senses of the word as shown in Figure 7.2. Each sense is represented by an absolute node in the KB.

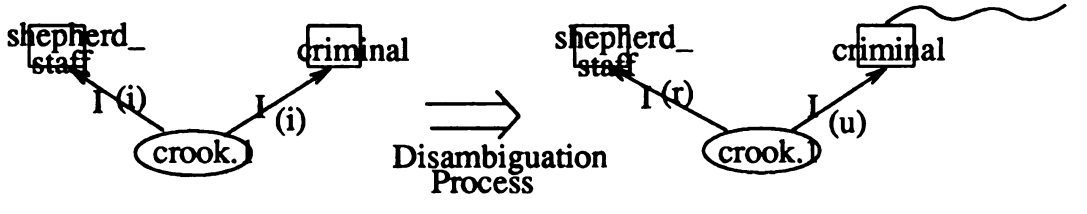


Figure 7.2. Representation of Multiple Word Senses

This means that SYNSEM supports the all-readings hypothesis for the word sense access (Swinney, 1979). To each I-link, a status flag is attached to indicate the status of

the link (the status of an I-link is written in the parentheses in Figure 7.2). A status flag can be one of the following values: "i", "u", and "r". "i" means that the link is in the initial state; In other words, it has not been processed yet since its creation. "u" indicates that the I-link is currently being used in a semantic path and the corresponding sense is still a candidate for the correct sense of the word. An I-link for the sense that has been eliminated is tagged with "r"(removed). When a word has been disambiguated fully, there should be only one I-link with the status flag "u" as illustrated in the left side of Figure 7.2.

Here, it is necessary to introduce a complicated matter that SYNSEM has to handle. As explained before, each branch of parsing has its own set of semantic paths. Therefore the word sense disambiguation status of a word may be different among the branches, which means that each branch should have its own representation of the disambiguation status of the words. This problem becomes more complicated with marker passing and semantic priming (that will be explained later). For example, consider Figure 7.3. Note the possible connections between Z and the instance for the new word. If branch-i is selected instead of branch-ii, then the sense of the new word should be the concept "V" in the figure. But if branch-ii is chosen instead of branch-i, then the concept "U" should be the correct sense of the new word.

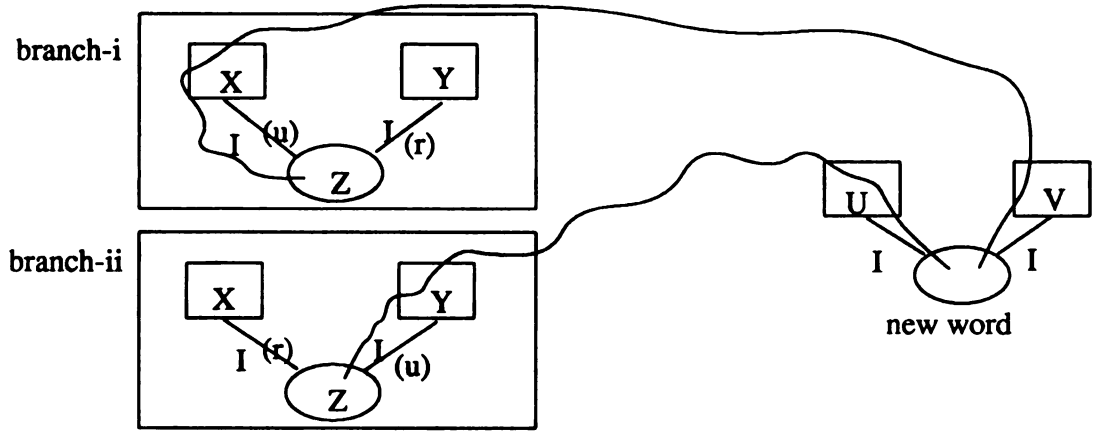


Figure 7.3 Different Word Senses for Different Branches

For the input word "crook", the MP passes markers starting from its instance node. The next word "operated" is input which is represented by the instance node `operated.2`. This word has two possible senses: to cause something to function and to perform surgery. Marker passing is done for this word, too. After syntactic analysis, SYN will send a suggestion `(crook.1 operated.2 subj)` to SEM according to the rule main-verb2. SEM will find all possible best paths as shown in Figure 7.4. There should be only one best path for a suggestion, but if one of the words related to the suggestion has multiple senses as in this example, more than one best path can be found by SEM. It is important for SEM not to remove any legitimate good path.

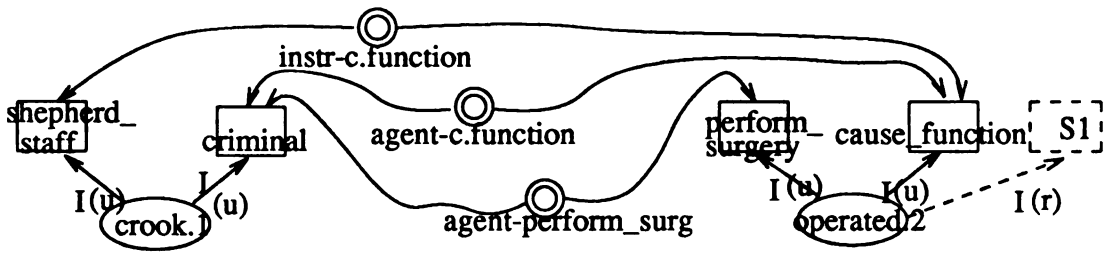


Figure 7.4 Paths between Two Instance Nodes

Three paths have been found in this example as in Figure 7.4. These paths are stored in the set of semantic paths. Note that `shepherd_staff` has no path to `perform_surgery` because the KB in figure 7.1 defines that the instrument of performing surgery should be some kind of medical tool and the shepherd's staff is not one of them. At this point of paring, the two senses of "crook" are still used in the semantic paths. Thus, it is not fully disambiguated yet. The word "operated" has not been disambiguated either. If we assume that the word "operated" has another sense `S1` (the dotted box in the figure) and it is not used in any path found, then the sense `S1` will be removed by attaching the flag "r" to the corresponding I-link. For a suggestion, usually one path is found. But three paths have been found because of the ambiguous words in this case. As the parsing goes on, the two false paths will be removed.

Now the next input is the canned phrase "pizza parlor" which is represented by the single concept `pizza_parlor`. Let's assume a canned phrase can be analyzed. After marker passing is done for this word, SYN will send the suggestion `(operated.2 pizza_parlor.3 obj)` to SEM by the **object** rule. For this suggestion, there is only one path (d) found as shown in Figure 7.5.

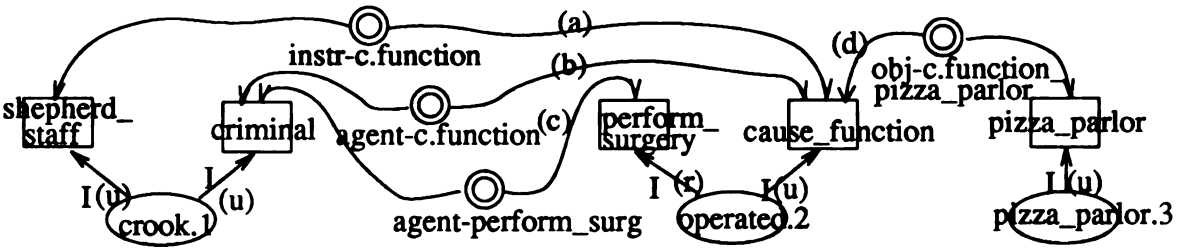


Figure 7.5 Effect of Reading a New Word

Note that the sense `perform_surgery` of "operated" is not used in this path. This means that `perform_surgery` cannot be the sense of the word "operated". The sense `perform_surgery` is removed by attaching the flag "r" to the corresponding I-link. The removal of the sense `perform_surgery` makes SEM remove the path (c)(in the figure) from the set of semantic paths, which results in the set of semantic paths shown in Figure 7.6.

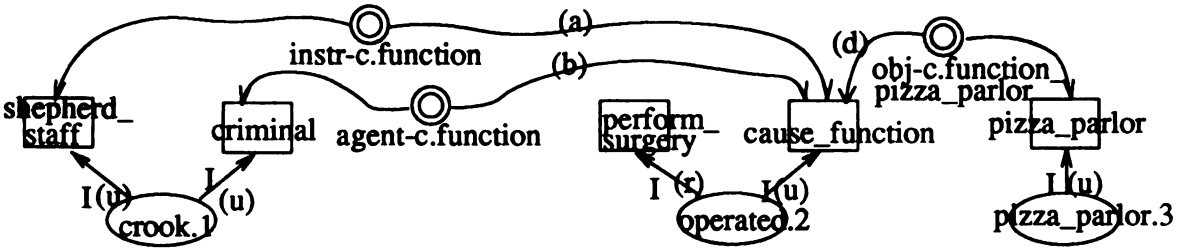


Figure 7.6 Paths after Removing a Sense and a Path

At this point of parsing, the word "operated" has only one sense with "u" flag, which means that it has been fully disambiguated. But there are still two senses with "u" flag for the word "crook". Thus, it has still word sense ambiguity.

If there were not path (b) between `crook.1` and `operated.2`, then the removal of the path (c) would result in the removal of the sense `criminal` of the word "crook" because `criminal` would not be used in any path connecting `crook.1` and `operated.2`. As soon as a sense gets an "r" flag, it will never be considered by SEM during the path-finding process.

At this point, let us consider a general case which can be illustrated in Figure 7.7. W.8, Y.1 and X.4 are instance nodes for the words already processed by the parser. W.8 has two senses which are P and Q and these two senses are still candidate senses. A and B are candidate senses for Y.1. D and E are candidate senses for X.4. At this point, the new word Z.6 has just been input. The suggestion between Z.6 and X.4 resulted in only one path (e) as shown in Figure 7.7. The sense D of X.4 is not used in this path (e). Thus, the sense D can not be the sense of X.4 and it is removed. The removal of the sense D causes the path (d) to be removed from the set of semantic paths. The removal of (d) causes the sense B to be removed from the possible senses of Y.1. The removal of the sense B again causes the removal of the path (a) from the set of semantic paths. This again causes the removal of the sense P from the possible senses of W.8. P is not connected to any other paths and thus the removal operation stops here. From this process, we can notice that the removal of a word sense recursively propagates through the KB via the semantic paths. We call this operation the *recursive word sense removal*(RWSR) operation.

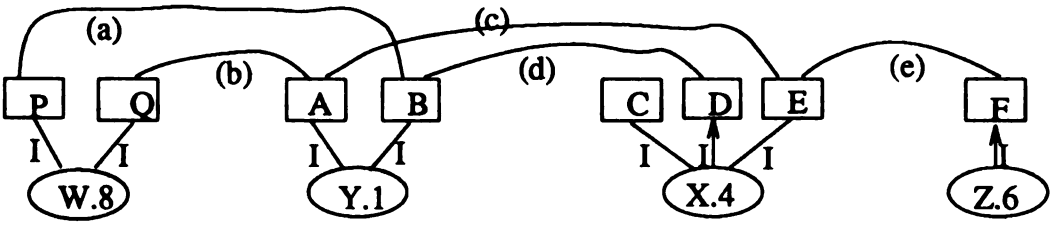


Figure 7.7. RWSR Operation

But there is a case which requires more careful consideration. Consider the case shown in Figure 7.8 which has one more path, (f), added to the case in Figure 7.7.

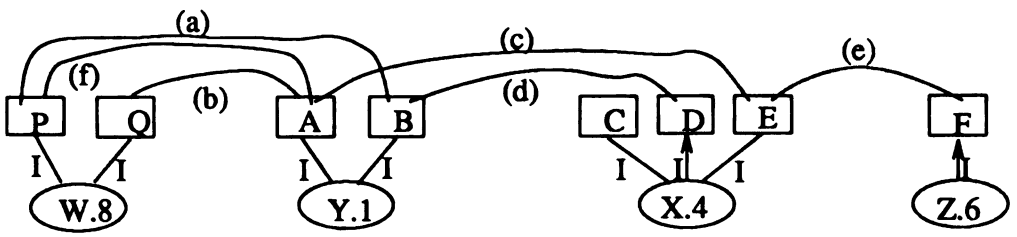


Figure 7.8 Consideration for the Set of Non-removable Senses

During the RWSR operation invoked by the removal of D of X.4, P will be considered for removal after the removal of the path (a). Note that P is also used in the path (f). Thus, P should not be removed. The sense P is an example of the so-called non-removable senses. After the path finding process for a suggestion has been done, the SEM computes the set of non-removable senses before the word sense disambiguation process starts. SEM computes the set of non-removable senses as follows: Starting from the word senses of the new word that are included in any semantic path found for the suggestion, find all word senses that can be reached via the semantic paths. In Figure 7.8, this set will be (F E A Q P). During the RWSR operation, any word sense that is in the set of non-removable senses is not removed.

In this section, we have been considering the word sense disambiguation process that is initiated by each suggestion sent by SYN. This operation starts from the set of semantic paths that is found by SEM for each suggestion from SYN. Let's assume that S is the set of semantic paths found for a suggestion having `old-instance` and `new-instance` as two origins. Then the word sense disambiguation is done as follows:

(1) Remove the senses of `new-instance` that are not in any of the paths in S(for example, the node C in Figure 7.9).

(2) Compute the set of non-removable senses. (In Figure 7.9, all senses that are reachable from A or B via semantic paths are non-removable senses.)

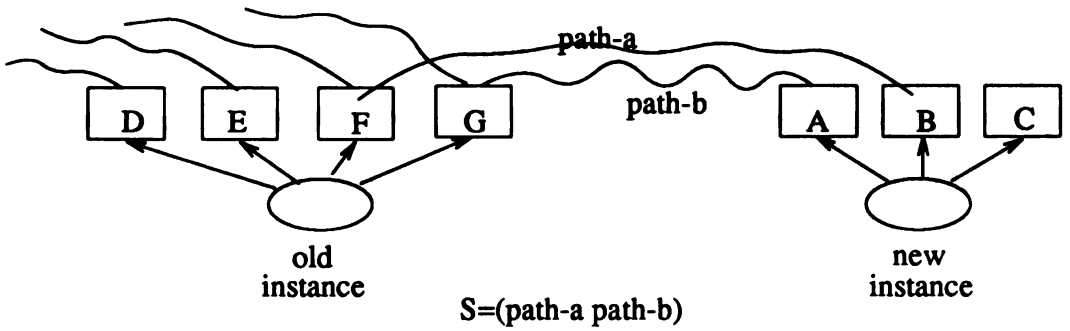


Figure 7.9 Starting the Removal Operation

(3) For each of the senses of `old-instance` having "u" flag, if the sense is not used in any path in S, call the RWSR operation passing this sense and `old-instance` as the arguments.

RWSR operation routine is shown in Figure 7.10.

```
Procedure RWSR_operation (sense instance)
 begin
  if sense is in non-removable-senses
     then exit
     else
       begin
          remove sense form the possible senses of instance;
          find all (sense instance) pairs that can be reached from
            (instance sense) pair via any path in the set of semantic
            paths and has not been visited yet. Insert the pairs into
            the set "reachable";
          remove any path used in the step 1 from the set of semantic paths;
          call RWSR_operation recursively for each element in the set
            "reachable" passing the element(which is pair) as argument;
       end;
 end;
```

Figure 7.10 RWSR Operation Routine

7.2 Use of Concretion

Another mechanism used by SYNSEM for word sense disambiguation is the *concretion* operation. A sense of a word is changed to a more specialized sense if a path with a special shape connecting the two instances of the suggestion are detected. Let's call this special shape *the concretion shape*. The concretion shape is shown in Figure 7.11.

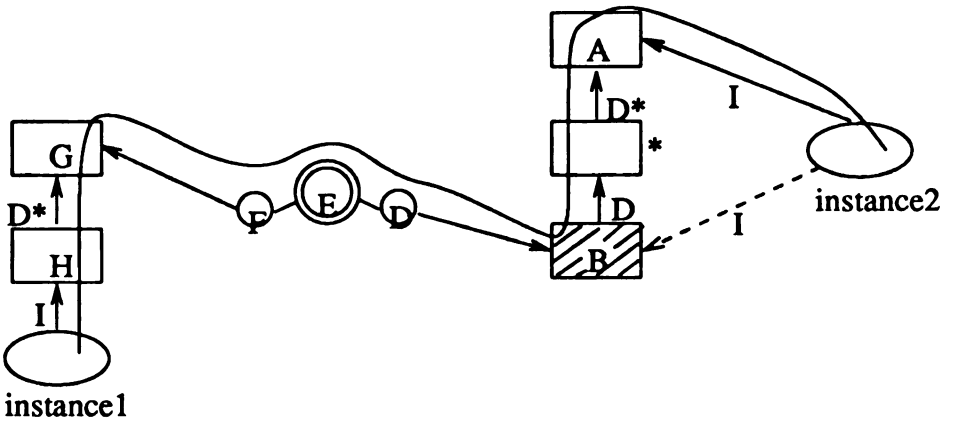


Figure 7.11 Shape of Concretion Path

In Figure 7.11, the path `(instance2 A B D E F G H instance1)` is of the concretion shape. Note that there can be zero or more absolutes between `G` and `instance1` and between `A` and `B` in the figure (as indicated with *). According to this concretion path, the sense of `instance2` can be specialized from `A` to `B`. The reason for this is as follows: The conceptual association between `instance1` and `instance2` is via `B` and `A`; But `B` is a specialization of `A`; Thus, `instance2` can actually imply the meaning `B` (which is a specialization of `A`). The concretion path in Figure 7.11 is the best path found for the suggestion `(instance1 instance2 prep)`.

The detection of the concretion path is done in a special way. Note that markers from `instance1` and `instance2` can not create a two-way collision at the relation node `E` because a marker from `instance2` can not flow in the reverse direction of D links and thus can not reach the node `E`. But it is necessary to get a two-way collision at

E so that a three-way collision can occur at E during secondary marker passing and thus the path can be found as the best path for the suggestion. This problem is handled in the following way. Note that the marker starting from instance1 will reach node A via the course of the concretion path (i.e. via H, G, F, E, D, and B). The two-way collision occurs at the node A when a marker from instance2 reaches the node A. For every two-way collision, the MP checks if the trace of one marker of the collision is of the concretion shape and the trace for another marker of the collision is of the form (collision-node I-link instance-node). Then a dummy two-way collision is put on the relation node which is in the middle of the trace (the node E in Figure 7.11). We add another case in which a three-way collision can occur in the case explained in Chapter 6. The dummy collision node is also used to form a three-way collision when it collides with a secondary marker. This is the way that the concretion path is found for a suggestion. When a concretion path is found for a suggestion, the sense of an instance is specialized. In the current example, the sense of instance2 is changed from A to B.

Let's look at the ongoing example that is using (7-1). In the previous section, it was explained that the path (d) in Figure 7.6 was found for the suggestion (operated.2 pizza_parlor.3 obj). Actually this path is a path of the concretion shape as shown in Figure 7.12.

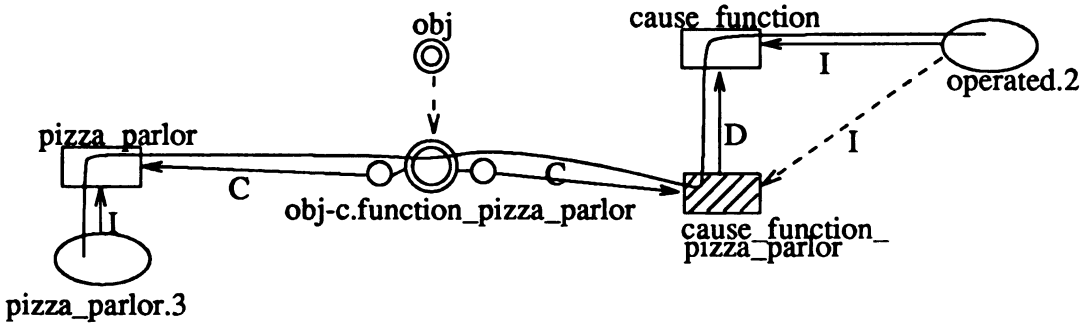


Figure 7.12 Concretion for the Example

Noticing that (d) in Figure 7.6 is a concretion path, the sense of "operated" is changed from cause_function to cause_function_pizza_parlor. This is called

the *concretion operation*. But this concretion operation requires the system to adjust the semantic paths that goes through the old word sense (`cause_function` in this example). For example, the path (a) and (b) in Figure 7.6 goes through `cause_function` to get to `operated.2` from `crook.1`. Thus these paths should be adjusted so that they can go through the concreted sense, `cause_function_pizza_parlor`, instead of the old sense. This is called *the path adjustment operation* after concretion. It is important to recognize that the path adjustment operation after concretion is also a source of word sense disambiguation. To see this, let's consider how the path adjustment operation is done in the ongoing example.

The new adjusted path should satisfy the following condition: the relation node of the new path is the specialization of(or same as) the relation node of the old path and the end absolutes of the old path is an ancestor of(or same as) the end absolutes of the new path. If a new path satisfying this condition can be found, the new path is added to the set of semantic paths in place of the old path. If the new adjusted path can not be found, the old path is simply thrown away.

The adjustment of the path (b) in Figure 7.6 is shown in Figure 7.13. The new adjusted path

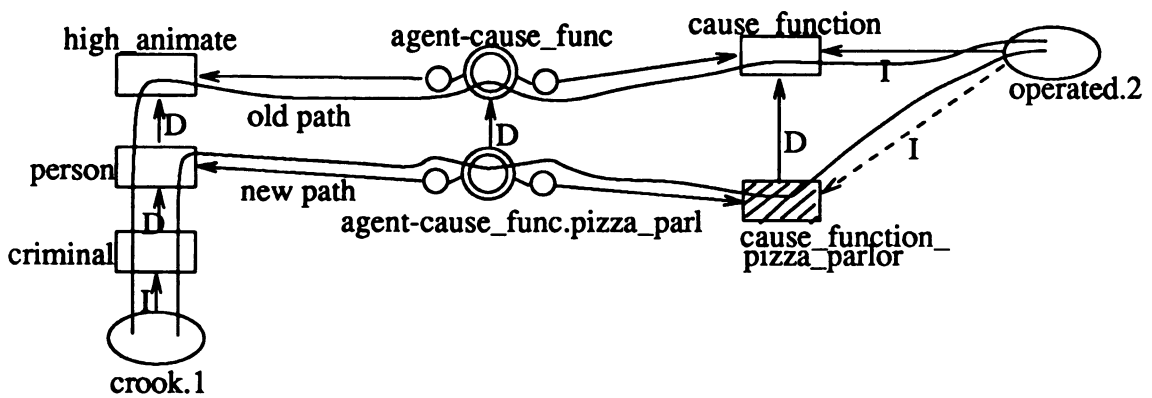


Figure 7.13 Success of Path Adjustment

shown in this figure satisfies the condition stated above. In the case of the adjustment operation for the path (a) in Figure 7.6, the new adjusted path can not be found as

illustrated in Figure 7.14. The reason is twofold: `shepherd_staff` is not an ancestor of `capital` and there is no connection between `capital` and `crook.1`.

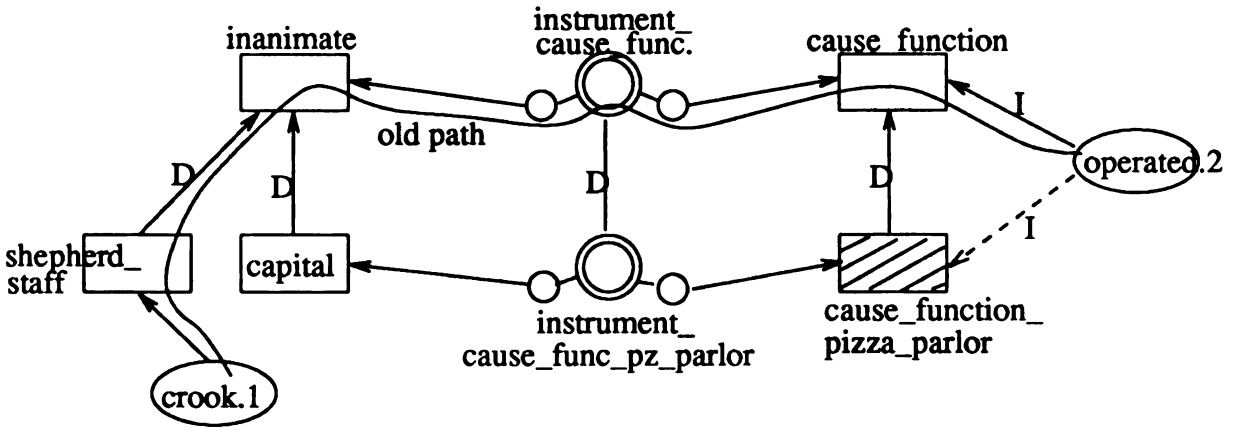


Figure 7.14 Failure of Path Adjustment

Therefore, the path (a) in Figure 7.6 is removed from the set of semantic paths, which results in the removal of `shepherd_staff` from the sense of `crook.1`. At this point, the word "crook" has only one sense `criminal` and is fully disambiguated.

The method of concretion used in this section is similar to the concept refinement process (specifically the slot-filler specialization rule) of Lytinen(1984). But the path adjustment operation after concretion was not used in Lytinen's method and is a new source of disambiguation information developed in SYNSEM. The disambiguation information from the path adjustment operation may lead to the elimination technique explained in section 7.1 to remove other senses, which means that the different sources of disambiguation cooperate with each other.

7.3 Use of Semantic Association

Semantic association is one of the sources of information that is useful for word sense disambiguation. It seems that Quillian(1968) is the first researcher who used it. Semantic association can be modeled naturally in a semantic network. Two concepts have a semantic association if there is a path connecting them in the network. A concept is said to be semantically primed by another concept if the access of the latter concept

facilitates the process(or access) of the former concept. In a marker passing system, nodes X and Y have semantic association if markers originating from them collide at a node. If X gets a marker whose origin is Y, it can be said that X is semantically primed by Y.

SYNSEM uses these notions for the disambiguation. When the MP begins to do marker passing for a new input word having multiple senses, it checks if there is a sense of this word that has been primed by a previous word. Then the MP removes the other senses of the new word by tagging the corresponding I-links with "prime-removed" (this is an additional possible value of the status flag). The MP does not pass markers from the senses with the "prime-removed" tag, which implies that these senses are not accessed. This is to follow the research in Seidenberg & etc.(1982) which found that semantic priming allows only the primed sense to be used. An ambiguous word which still has more than one sense remaining can be disambiguated by a new input word if there is a *strong* semantic association (detected by the MP) between one sense of the ambiguous word and one sense of the new word. Strong semantic association is defined as follows in SYNSEM: there should be a path that does not involve more than two relation nodes and the end absolutes of the path should not be promiscuous concepts. The MP removes the senses of the ambiguous words that are not involved in the strong semantic association by tagging them with "prime-removed". This process can be explained using the input sentence (Hirst, 1984):

(7-2)The slug operated the vending machine.

We assume that the possible senses of "slug" are: a gastropod without shell, a fake coin, a bullet or a shot of liquor. After "operated" is processed, the following three paths are found and registered :

(7-3)
```
(slug.1  fake_coin  coin  inanimate  instr-c.function
    cause_function operated.2)
```

(7-4) (slug.1 bullet inanimate instr-c.function cause_function operated.2)

(7-5) (slug.1 shot_liquor inanimate instr-c.function cause_function operated.2)

Note that perform_surgery is removed from the possible candidates for the sense of "operated" because it is not used in the paths. The sense gastropod_without_shell is not used in any path and thus it is removed from the possible senses of "slug". But the three senses(fake_coin, bullet, and shot_liquor) are still possible candidates for the correct sense of "slug". After "the vending machine" is input, the MP finds the following path indicating strong semantic association between slug.1 and vending_machine.3 through the sense fake_coin:

(7-6) (vending_machine.3 vending_machine obj-c.function_vendingmachine cause_function_vendingmachine instr-c.function_vendingmachine coin fake_coin slug.1)

Thus fake_coin is chosen as the sense of "slug", and other senses and their corresponding paths((7-4) and (7-5)) are removed. Word sense disambiguation using semantic association is graphically shown in Figure 7.15. Note that the removal of the sense A in the figure invokes the RWSR operation explained in the above section.

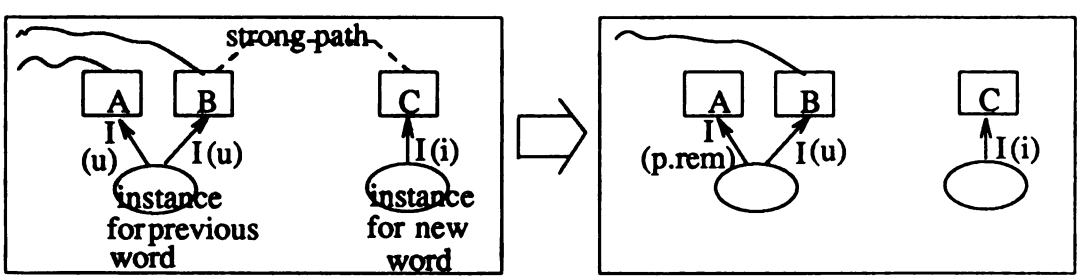


Figure 7.15 Use of Semantic Association

7.4 Use of Information about Syntactic Category

Another source of word sense disambiguation is syntactic categories of words assigned during syntactic analysis. Let's consider the sentence (7-7) to illustrate how disambiguation is done using the categorical information.

(7-7)The granite rocks near the shore.

The word "rocks" has two possible senses: *stone* and *shake*. But *stone* is used as the sense of "rocks" only when the word is interpreted as a noun and *shake* is used as the sense only when the word has verb as its part of speech. As soon as the syntactic analysis component determines that "rocks" is a verb, it can be determined that its word sense is *shake*. If it is decided that "rocks" should have the syntactic category of noun, its sense automatically becomes *stone*.

This disambiguation method is also used in SYNSEM. When an instance node is created for the word "rocks", it attaches the part of speech to the I link of each sense as shown in Figure 7.17. The MP flows markers through all senses of

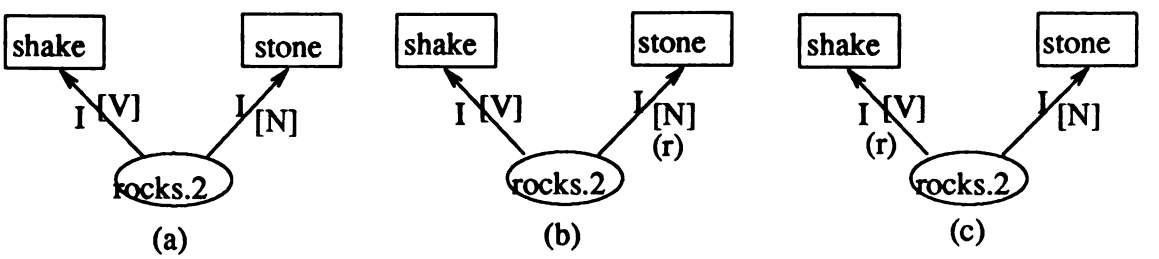


Figure 7.17 Use of Categorical Information

the instance because it does not know yet which sense is the correct sense of the word. Then three rules(noun-parse3, noun-parse4 and noun-parse5) are found by the pattern matcher as shown in Section 6.4.2. Three branches are created for the three rules and they run in parallel as described in that section. The branch for noun-parse3 analyzes "rocks" as a verb. (Note that each branch keeps its own copy of I links for ambiguous words.) In this branch, the I link for the sense `stone` is removed so that it can never be used during the process of finding the best path for the suggestion in the

branch as shown in Figure 7.17(b). Thus the paths to `rocks.2` can only go through the sense `shake`.

The branch for noun-parse4 and the branch for noun-parse5 set the I links for `rocks.2` as shown in Figure 7.17(c) so that the link for the sense `shake` can never be used for finding the paths for their suggestions (because they analyze "rocks" as a noun).

When one branch is finally chosen over the others, the correct sense of "rocks" is also automatically chosen which conforms to the syntactic category of the word. In this example, the branch for noun-parse3 is the one that the parser finally chooses and thus the sense `shake` is automatically chosen as the correct sense of "rocks".

7.5 Conclusion

A new method for applying the word sense elimination mechanism has been proposed. We have shown that a knowledge base encoded with the KODIAK knowledge representation language can be used to facilitate word sense disambiguation in natural language parsing. The basic strategy is to make the parser keep track of a set of semantic paths corresponding to the semantic interpretation of the input sentence. These are the paths that connect the instance concepts of input words. Senses of a word that are not used in the semantic paths found for a suggestion from the syntactic analysis component are removed. The removal of a sense at one end of a registered path results in the removal of the sense at the other end of that path. The removal operation is applied recursively.

A concretion mechanism similar to the concept refinement process is also incorporated into the disambiguation method in the SYNSEM parser. The idea of the path adjustment operation is newly employed to extend the concretion mechanism. The use of marker passing enables the parser to use semantic association and semantic priming as another source of disambiguation information. The syntactic analysis helps the parser eliminate word senses that do not correspond to the correct part of speech.

A problem occurs if semantic priming leads the parser to select an incorrect word sense. Further research is required to handle this problem. The fundamental problem that waits for additional research is how more diverse kinds of knowledge such as goals, intentions, discourse structures, and more global context (spanning to several sentences) can be applied to the word sense disambiguation problem.

CHAPTER 8

CONCLUSION

The parsing model with improved interleaved semantic processing developed in the thesis research will be summarized first and then it will be argued that the proposed approach of parsing is an integrated parsing approach. Then the achievements that have been made in this thesis will be outlined. Finally the problems that need more research related to the thesis will be enumerated and discussed.

8.1 Proposed Approach to Parsing

The objective of the research has been to develop a natural language parser that can utilize both syntax and semantics as early and fully as possible. To achieve this objective, we have taken the approach of *interleaved semantic processing* which is based on the syntax-oriented analyzer using semantics to get intermittent feedback. As suggested in several parsing models in psycholinguistics which proposed the parallelism, all syntactic alternatives of a structural ambiguity are considered in parallel. Each alternative invokes a new branch of parsing. The branches run in parallel until all branches need to read the next input word. At this time, the semantic comparison stage is invoked, during which the semantic processing results of the branches are compared and only those with best results are kept alive while the others are thrown away. This is a source of structural disambiguation that uses semantic information.

The branches out of the semantic comparison stage run again in parallel until the next semantic comparison stage comes. The semantic results collected during this interval between the two semantic comparison stages are used again for filtering the branches. The branches run asynchronously in parallel, but they are synchronized

periodically by the reading of words. Some branches might be removed if they reach a dead-end because the following input material does not agree with their analyses, which is realized by the fact that no rules are found during the pattern matching step and the situation is determined to be a dead-end. The removal of branches here is a source of structural disambiguation that is based on syntactic information. Another type of ambiguity that is abundant in natural languages is word sense ambiguity. Much effort on the word sense disambiguation problem has also been made in this research. The theoretical support for the parsing approach taken in this thesis can be found in Kurtzman's parsing model that advocated for immediate parallel analysis(IPA) with strong parallelism and conceptual selection hypothesis.

A parser that had a similar motivation as SYNSEM is Lytinen's MOPTRANS parser. But he approached the problem of building a parser with good use of syntax and semantics from the integrated parsers such as the conceptual analyzers. MOPTRANS adopted the control in the sense of Crain and Steedman's strong interaction between syntax and semantics, which means that, in the MOPTRANS parser, the selection of the syntactic rules is influenced by semantics. Actually only one rule that corresponds to the best semantic interpretation is chosen and other alternatives are not even considered. Lytinen's approach is directly against most of the parsing models proposed in psycholinguistics. In addition to the difference in general approach, SYNSEM and MOPTRANS are different in the inside detail of the parsing. It is argued in this thesis that SYNSEM's parsing approach constitutes a new distinct approach that utilizes syntax and semantics as early and fully as possible.

8.2 Argument for the Proposed Approach

It was pointed out that the parsers with interleaved semantic processing are not good enough in utilizing semantics. We will show that SYNSEM has improved on this point over the previous parsers. (The exact paths in the KB corresponding to the

suggestions will not be provided in this discussion; We will also sometimes use the terminologies such as "slot" of a frame representation language for the ease of explanation.) Let us assume that SYNSEM parses sentence (8-1) and SYN has just determined *Mary* to be a NP:

(8-1)John gave Mary a book.

At this point, two rules match the syntactic working memory(SWM): **object** and **ind-object**. The **ind-object** rule matches the SWM because the verb *gave* is a dative verb. SYN sends the suggestion (`gave.2 Mary.3 obj`) according to the **object** rule. Another suggestion (`gave.2 Mary.3 to`) is sent to SEM according to the **ind-object** rule. Here `gave.2` is the instance node of the concept `give`, and `Mary.3` is the instance node of the concept `female`. The path for the suggestion (`gave.2 Mary.3 to`) is better than the path for the suggestion (`gave.2 Mary.3 obj`) because `person` fits better into the RECIPIENT slot(implied by `to`) than the PATIENT slot (implied by `obj`) of `give` (more precisely, because `high_animate`, the constraint of the relation `recipient-give`, is a more specific concept than `phy_object`, the constraint of the relation `patient-give`). Thus the branch for the **ind-object** rule is selected over that of the **object** rule. This shows that SYNSEM correctly interprets "gave Mary" in (8-1) as an integrated parser does.

Another illustration can be provided by using the sentences from (8-2) to (8-4):

(8-2)The teachers taught by the Berlitz method passed the test.

(8-3)The students taught by the Berlitz method passed the test.

(8-4)The horse raced past the barn fell.

Lytinen(1984) argued that an integrated parser should be able to explain why (8-2) is a garden path sentence while (8-3) is not, even though they have the same syntactic structures as (8-4) which is a well known garden path sentence. After SYNSEM inputs "taught" in (8-2), three syntactic rules(**main-verb2**, **np-vpp1**, and **np-vpp2**) will match

the SWM because "taught" can be either a "past active" or a "past participle". The main-verb2 rule analyzes *the teachers* as the subject of the past active "taught". The np-vpp1 rule analyzes "the teachers" as the direct object, and the np-vpp2 rule analyzes "the teachers" as the indirect object of the past participle "taught". Each of the three rules creates a branch. These branches send three suggestions to SEM: (`teacher.1 taught.2 subj`) for the main-verb2 rule, (`teacher.1 taught.2 obj`) for the np-vpp1 rule, and (`teacher.1 taught.2 to`) for the np-vpp2 rule. For the suggestion of each branch, the best path connecting the two instance concepts in the suggestion will be found. Then at the semantic comparison stage, the three paths of the three branches will be compared and the best one will be selected. The path for the main-verb2 rule is best because the concept `teacher` can be a prototype for the AGENT slot but not for the RECIPIENT or SUBJECT-MATTER slot of the concept `teach`. (Note that the "P"(prototype) link is added in addition to the "C"(constraint) link between an absolute and an aspectual in our version of KODIAK.) Thus, the branch for the main-verb2 rule is selected which analyzes *the teacher* as the syntactic subject and thus a garden path phenomenon occurs in (8-2). But in the case of (8-3), the path for the suggestion of the np-vpp2 rule (that determines *the students* as the indirect object) is best among the three paths for the three suggestions of the three rules because the concept `student` is a prototype of the RECIPIENT slot of the concept `teach`. Therefore a garden path phenomenon does not occur in sentence (8-3) even if it has the same syntactic structure as sentence (8-4). So our parser behaves in the same way that Lytinen expected for an integrated parser.

It can be shown that the integrated parsing approach of MOPTRANS which performs semantic processing before syntactic processing in each step of parsing is more inefficient in some cases than the approach of SYNSEM. MOPTRANS tries to connect two semantic objects without using the information that syntactic knowledge can provide. Let's consider (8-5) to illustrate this point:

(8-5)The man attacks a bear with a boy.

After inputing *attacks*, MOPTRANS will try to find all possible connections between the semantic objects in memory: MAN and ATTACK. It will find the three possible semantic connections: MAN can be the AGENT of the action ATTACK; MAN can be the PATIENT of the action ATTACK; and finally MAN can be the PARTICIPANT of the event ATTACK. Then it will try to find the best one among the three.

In the case of SYNSEM, only one syntactic rule, main-verb2, will match the SWM. Therefore the relation node `patient-of-attack` and the relation node `participant-of-attack` will never be considered because these two nodes will never get marked during the secondary marker passing that starts from the node `subj`. The three-way collision will occur only at the node `agent-of-attack`. SYNSEM will have only one branch for one rule and never consider the semantic connections that should be considered by MOPTRANS. In a large and complex knowledge base, there can be many possible connections between semantic objects which may make the MOPTRANS' parsing approach inefficient.

From these examples, we argue that SYNSEM utilizes syntactic and semantic knowledge as fully and efficiently as any existing integrated parser.

8.3 Observations on Implementation and Test

The SYNSEM parser has been implemented on a SUN 3/280 system using Common Lisp. Appendix C lists the test sentences used to test the parser developed to demonstrate the working of the theory explained in this thesis. Appendix D shows the total output of the parser to parse a sentence, which is useful to track the course of parsing. (The comments are shown in italics. The sentences which the parser could not handle satisfactorily are preceded by an asterisk.)

The size of the parser is reflected in the following data:

Number of absolutes : 172

Number of relations : 145

Number of aspectuals: 290

Number of syntactic rules : 80

Number of Lisp lines : 7400 (including comment lines)

As far as the speed is concerned, the prototype of the SYNSEM parser is not good. It is slow, and needs some analysis to find out ways to increase the speed. For sentences with around 10 words, it takes about 3 minutes for parsing. But it would be useful to report what has been observed during the test:

(a) Most of the parsing time is spent in the steps of doing marker passing for open words (it has been called primary marker passing in this thesis). From three or four open words in the input string, it sometimes takes more than a minute to do primary marker passing. It should be mentioned that much portion of this time is for doing garbage collections by the lisp interpreter. I conjecture that the reason for the so many garbage collections is due to the use of a recursive function for finding paths for each collision. It will be interesting to see what will happen in the speed if the recursive function is replaced by the corresponding non-recursive version. The time taken for marker passing is about 90% of the total parsing time.

(b) The syntactic analysis component is fast and is not affected much by the increase of the number of words in the dictionary. The size of the rule base did not change much after the initial writing of the rules. With the rule base that has most of the rules that can do the clause level analysis, it has been found that the time taken for syntactic analysis is only a fraction of the total parsing time (around 4% of the total parsing time).

(c) Time taken for finding the best paths for suggestions with a preposition is only a fraction of the total parsing time and it did not vary much with the increase

of the knowledge base. This is due to the efficiency provided by the secondary marker passing mechanism (around 6% of the total parsing time).

(d) For suggestions which do not contain a preposition, the time to find the best path is too long (even much longer than the time for marker passing.) Thus it is really necessary to develop some method to speed up this process.

(e) Based on the above observations, it is concluded that it is necessary to develop a mechanism using some parallelism to speed up the marker passing process. (This is true regardless of the availability of the efficient method of path finding.)

8.4 What has been achieved

In this section, the items that have been achieved during the project will be listed and summarized:

- A new integrated parsing has been achieved by updating interleaved semantic processing. The details have already been summarized and discussed in the previous two sections in this chapter.
- A rule-based syntactic analyzer with the following new features has been developed. The base pattern is used to check the state of parsing which results in the removal of packets or states from the parser. A method of "delaying the decision until enough context appears" has been created and implemented. This enables the parser to have the same power as a parser using lookahead and to use simple and uniform processing instead of the attention shifting mechanism.
- A simple and clean interface between syntax and semantics has been developed. This is achieved by using *suggestions* that are sent from the syntactic processing component to the semantic processing component. But it seems that this method can not cover everything. The use of the semantic comparison stage provides a way of using semantic information to control the syntactic processing.

- A mechanism called secondary marker passing has been developed to facilitate the use of the marker passing paradigm in natural language parsers. This provides an efficient way of finding the best paths between two concepts in the knowledge base. Without this mechanism, we should tolerate the high computational cost or we should give up using the marker passing paradigm. As Charniak(1986) pointed out, there is yet no better alternative to marker passing.
- Along with the secondary marker passing mechanism, several heuristics have been developed to find the best paths between the concepts. It has been observed that usually the secondary marker passing mechanism and the H-heuristic are powerful enough to find the best paths. The S-heuristic is necessary in a complex situation such as the existence of word sense ambiguity.
- The critical problem of scoring the alternative analyses has been circumvented by the method of rank-ordering the branches by comparing the knowledge base paths. A method of comparing the paths has been developed under the context of the SYNSEM parser. The basic idea was to use the specificity of the concepts (these were called the end absolutes) that constrain the first and last relations in the path.
- Attacking the disambiguation problem of structural ambiguity is not enough because word sense ambiguity is another major source of ambiguity. Broad and deep treatment of this problem has been sought in this research. It has been suggested that managing the set of knowledge base paths (called the semantic paths) and propagating the elimination of word sense via these paths is an efficient method to remove the false senses. The path adjustment operation related to the concretion of a word sense has been proposed as another source of word sense disambiguation.

8.5 Future Research Items

In this section, limitations and the research items identified during the development of the SYNSEM parser will be listed:

- The first and most important problem on the list seems to be the problem of knowledge representation. Presently semantic information that can be used by SYNSEM is limited to the kind of knowledge that can be encoded using KODIAK. KODIAK is one of the most recent knowledge representation languages but it can not satisfy the users fully. For example, how can we represent the fact that "eating on the train" is more plausible than "an apple on the train"? (for the sentence "The man ate an apple on the train.") How can we represent the fact that "discussing running as a sport" is more plausible than "discussing running as an act of engaging in running"? Another problem related to knowledge representation is the use of other diverse kinds of knowledge sources such as knowledge about discourse structure, user modeling, goals and intentions, common sense knowledge such as time and unexpected happenings, etc. Some of these might need a totally new kind of knowledge representation language. To achieve the full use of semantics the system should be able to utilize these various knowledge sources. It is clear that the current version of KODIAK can not easily accommodate all of these knowledge structures in a reasonably easy way (even if it can). It is not clear that the present architecture of the parser can remain unchanged when these other knowledge sources are eventually used.

- The current grammar in SYNSEM needs to be extended so that more complex syntactic structures can be analyzed, for example, topicalization, particles, etc. But no major obstacles are anticipated in doing this.

- Parsers which use semantics early have an advantage in handling ill-formed inputs. It is worthwhile to extend the parser to be able to handle ill-formed inputs. The basic strategy can be an attempt to find the semantic connections between the

chunks that can not be linked by syntactic relations. Finding the paths based on marker passing can be useful for this purpose.

- Marker passing in SYNSEM requires a large amount of computation. Part of the reason is that detection of the concretion paths and strong semantic association paths related to word sense disambiguation should be done during marker passing. It seems interesting to consider the use of massive parallelism in marker passing to increase the speed.
- The branches running in parallel build their own copies of syntactic representation. It might be the case that the same structures are built in several branches. Intuitively, the human parsing system does not seem to have multiple copies of the same structure. It is better to share the data structures among the branches as far as possible.
- SYNSEM has no efficient way of finding the best path between two nouns that appear in the nominal compound. It just uses the brute force method to compare every pair of paths to find the best path and thus it takes much time. The problem is that there is no cue to facilitate the search (such as syntactic relation or preposition). It is necessary to develop a better scheme to find the best semantic connection among the nouns in the nominal compound.

APPENDICES

Appendix A

Structure of Noun Phrase(NP)

Figure A-1 is assumed to be the structure for an NP in SYNSEM. It is not claimed that this structure is the optimal one of the NP. But this NP structure is introduced here for the ease of following the grammar rules and examples appearing in the thesis. Note that the post NP modifiers such as the PP or the relative clause, etc. are not shown in the figure. Instead, only the portion of the NP up to the head noun is shown in the figure.

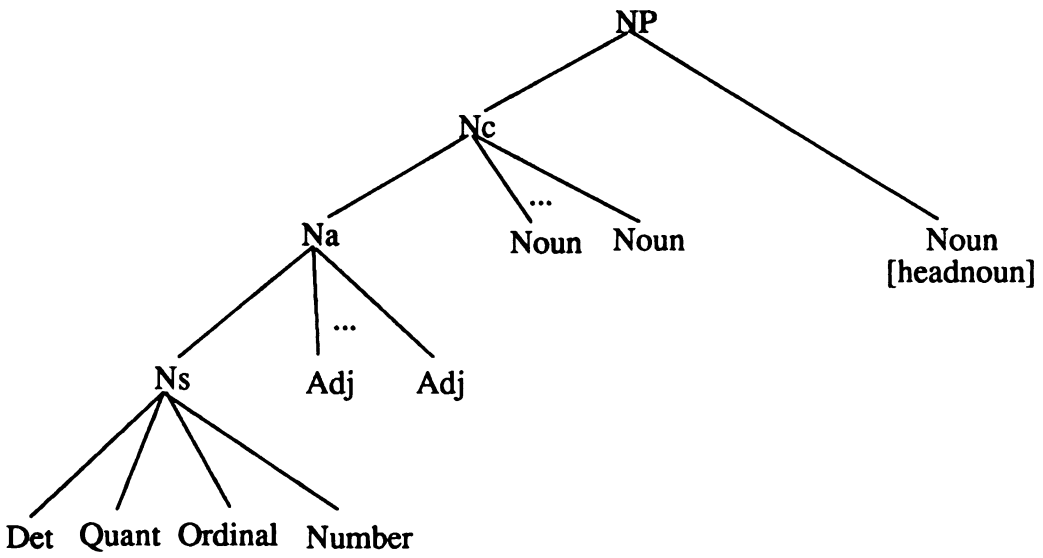


Figure A-1 Structure of an NP

Appendix B

Determining Dead-end Situation

When the pattern matcher finds no rules that match the syntactic working memory(i.e. CLIST), the parser should decide if the situation is the dead-end situation or just the situation that requires a new input (called read-again situation). The parser uses the routine called "justify-situation" for the decision. This routine is called by the parser with no input argument(this actually use CLIST for its working). This routine is explained below.

Let's call a tree in CLIST Ci where "i" indicates the position of the tree from the right side of CLIST. C1 is the rightmost tree, C2 is the second rightmost tree, etc. Let's call the raw patterns of rules RPi where i indicates the position of the raw pattern. Thus RP1 is the rightmost raw pattern, RP2 the second rightmost raw pattern(if it exists), etc.

C1 might lead to the construction of a basic nodes such as an NP or a PP. For example, DET can be considered to be the leading edge of an NP. Let's use SC1(starting edge of C1) to specify the basic node of which C1 is a leading edge. SC1 of det, ns, na, or nc is NP. SC1 of prep is PP. Except these, SC1 of C1 is set to nil to indicate that C1 does not lead to the basic node of NP or PP.

The basic scheme used in the routine "justify-situation" is that if the right side of CLIST can match the pattern part of any rule by using SC1 or the initial portion of the pattern part of any rule(indicating that the rule can be matched in the near future after more context is built in CLIST) with or without using SC1, then it is decided that the situation is a read-again situation.

This routine is shown in more detail below:

```
Routine justify-situation:

{ The return with yes means that the situation is decided to be a read-again
situation and the return with no indicates the dead-end situation.
The length of a rule is the number of raw patterns in the rule.          }

(1) if CLIST contains only one tree then return with yes;
(2) if C1 is a conjunction node(and, or, but), then return with yes;
(3) if C2 is of type S then
      if SC1 =/ nil
         then begin assume CLIST whose C1 is replaced with SC1;
                    call justify-situation with this new CLIST;
              end
         else begin if there is a rule of length 2 whose RP2 is C1
                       then return with yes;
                    if there is a rule of length 3 whose RP3 is C1
                       then return with yes;
                    return with no;
              end;
(4) if SC1 is non-nil and there is a rule of length 2
       whose RP1=SC1 and RP2=C2, then return with yes;
  {So far, checking with the rules of length 1 or 2 has been finished.
 From now, try the rules with length 3.}
(5) if there is a rule whose RP3=C2 and RP2=C1 then return with yes;
(6) if there is a rule whose RP3=C2 and RP2=SC1 then return with yes;
(7) if there is a rule whose RP3=C3, RP2=C2 and RP1=SC1,
       then return with yes;
(8) return with no;
```

Appendix C

Test Sentences

Diverse kinds of sentences have been used to test the features of the SYNSEM parser. The test sentences are listed under the categories which indicates the feature of ambiguity to be tested. The asterisk in front of a sentence indicates that SYNSEM can not handle the sentence satisfactorily.

(1) **pp-attachment ambiguity**

The man ate a cake with a fork.

The boy ate the pizza with the man.

The man ate the cake with frosting.

The man ate an apple from the orchard.

The employees of the company with the disease were examined by the doctor.

*The women discussed the dogs on the beach. ; *Kurtzman(1985). SYNSEM does not have enough conceptual expectation(described in Section 2.2.3) to handle this sentence.*

I bought the book for the girl in the bookstore. ; *the concept bookstore-buying has a slot of location to which bookstore is the prototype. Thus "in the bookstore is attached to the VP.*

The man ate in the restaurant with Tom.

The man ate the cake with Tom.

The man ate with Tom.

The spy saw the cop with the binoculars. ;*Ferreira & Clifton(1986)*

The spy saw the cop with the revolver. ;*Ferreira & Clifton(1986)*

(2) **Active past/past participle ambiguity**

The teacher taught by the pool passed the test. ; *Crain & Steedman(1985)*

The present wrapped by Judy was given to Tom. ; *Lytinen(1984)*

The horse raced past the barn fell. ; *Crain & Steedman(1985)*

The students taught by the pool passed the test. ; *Crain & Steedman(1985)*

(3) **Case determination ambiguity**

*The man ate the cracker with the dip. ;*the fact that "with" can indicate the co-object of some verbs should be encoded in the parser, which needs more consideration.*

I told the story to John.

I told John the story.

I told John the story was the last thing I wanted to hear. ; *Barton & Berwick(1985)*

The person gave the cat to Mary.

Tom broke the window with a rock.

The rock broke the window.

The rock broke. *Syntactic analysis is fine. But it misanalyzes semantically("rock" is analyzed as the instrument. Currently there is no way to detect it and reanalyze it later.*

(4) **relative clause attachment ambiguity**

The man bit the apple on the desk that the boy ate.

The man bit the apple on the desk that the boy painted.

*The man bit the apple on the desk that the boy bought. ;*It is hard to determine where to attach the relative clause. It is necessary to have some semantic information that it is likely for boys to buy an apple but not a desk.*

The girl saw the lion in the field which Tom shot in the body.

(5) **missing "that" ambiguity** Tom knew the man.

Tom knew the man ate the apple.

I told John the story was the last thing I wanted to hear. ;*At first "the story" is analyzed both as the object of "tell" and as the subject of the complement clause. As soon as "was" is seen, the object analysis is thrown away.*

(6) **Conjunction**

The boy and the girl ate an apple.

The man bit and ate the apple.

The man bit the apple and ate the pear.

The man ate an apple and the girl bit the pear.

The man ate the apple and the pear.

The boy ate and the girl saw the apple.

(7) **Wh-phrase sentences**: Wh-question and relative clause

Which apple did the man eat ?

I ate the apple which the man bit.

I ate the apple that the man bit.

He ate the apple the man bit.

Who do you want to visit?

Who do you want to eat the apple? *The equi-np deletion analysis and the Wh-analysis("who" fills the gap between "want" and "to") goes on in parallel until one is rejected.*

Who do you want to give the apple to? ; *Marcus(1980)*

What did you give Tom the book for? ; *Hirst(1984)*

I gave the boy who you gave the apples to three books.

(8) **Equi-np deletion**

The man wanted to eat the apple.

Tom wanted the man to eat an apple.

The man wanted the book.

(9) **Semantic anomaly**

The boy drank the apple. ;*the analysis fails because of semantic anomaly("apple" can not be object of action "drink".*

*The boy ate the green apple. ;*The semantic oddness of eating unripe apple can not be detected in SYNSEM*

(10) **Word sense ambiguity**

The slug operated the vending_machine. ; *Hirst(1984)*

The man operated the machine with the tool.

The granite rocks near the shore. ; *Milne(1982)*

The crook operated the pizza_parlor. ; *Hirst(1984)*

*The astronomer married a star. ; *Charniak(1983). "star" is at first disambiguated to be a concept of "star in the sky" but later the parser gets into trouble because the object of "marry" should be of type "person".*

Tom fixed the washer. ; *Lytinen(1984)*

Tom fixed the game. ; *Lytinen(1984)*

(11) **Metaphor**: *SYNSEM can not handle this type of sentence.*

*The car drank the gasoline. ; *Small(1983)*

(12) **Multiple part of speech ambiguity**

The granite rocks near the shore. ; *Milne(1982)*

*The prime number few. ; *Milne(1982). SYNSEM can not handle this one because of the lack of necessary semantic information.*

(13) **Plain sentences**

The apple is big.

The boy said that Tom had stolen the watch.

The girl sleeping in the room wanted to visit Tom.

(14) **Ambiguity from present participle**

*The man eating tiger growls. ; *Small(1983). SYNSEM currently can not handle this sentence because it does not have appropriate syntactic rules.*

The man eating shrimp growls. ; *Small(1983)*

(15) **Vague word sense ambiguity**

I took a picture.

The man took aspirin.

The man got the goods. ; *"got" is analyzed to be "buying".*

(16) **Semantic association between nouns in a nominal compound**

The cotton clothing is made of grows in Mississippi. ; *Marcus(1980). SYNSEM can not model the garden pathing effect of this sentence shown by humans. Two possible analyses are sought in parallel in SYNSEM.*

Appendix D

Output of Parsing

The following system output of SYNSEM shows how the parser works during the parsing of sentence "the teacher taught by the pool passed the test" which was explained extensively in Chapter 4 and 6.

```
;;; Dribble file #P"/usr.MC68020/users/staff/ra/nlp/out-taught" started
T
> (mon)

RRRRRRRRRead next word : THE

A word read in read-word: word=@
Come into rule-matching part....
rule succeeded:S-START
Return from rule-matching routine. rules= (S-START)
Only one rule selected = S-START

 ==== Trees in CLIST ====
  node:S2  pos:(S)  name:NIL  range:(1 1) semsug:NIL
  parent:NIL  features:((S (MAIN))) trace:NIL
     --
     node:SS1  pos:(SS)  name:@  range:(1 1) semsug:NIL
     parent:S2  features:((SS)) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------

A word read in read-word: word=THE
Come into rule-matching part....
Return from rule-matching routine. rules= (PARSE-DET)
Only one rule selected = PARSE-DET

 ==== Trees in CLIST ====
  node:S2  pos:(S)  name:NIL  range:(1 1) semsug:NIL
  parent:NIL  features:((S (MAIN))) trace:NIL
     --
     node:SS1  pos:(SS)  name:@  range:(1 1) semsug:NIL
     parent:S2  features:((SS)) trace:NIL
  node:NS4  pos:(NS)  name:NIL  range:(2 2) semsug:NIL
  parent:NIL  features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:DET3  pos:(DET)  name:THE  range:(2 2) semsug:NIL
     parent:NS4  features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : TEACHER

passing markers. instance=<abso TEACHER.1>
Bump into node carrying same ori mark at cnode=<asp TEACHER.TEACHER-TEACH>
Loop detected at cnode=<abso PERSON>
Loop detected at cnode=<abso PLACE>
```

```
A word read in read-word: word=TEACHER
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE1)

Only one rule selected = NOUN-PARSE1

==== Trees in CLIST ====
 node:S2 pos:(S) name:NIL range:(1 1) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS1 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S2 features:((SS)) trace:NIL
 node:NC6 pos:(NC) name:NIL range:(2 2) semsug:NIL
 parent:NIL features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
   --
   node:NS4 pos:(NS) name:NIL range:(2 2) semsug:NIL
   parent:NC6 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:DET3 pos:(DET) name:THE range:(2 2) semsug:NIL
     parent:NS4 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
 node:NOUN5 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
 parent:NIL features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : TAUGHT

passing markers. instance=<absо TAUGHT.2>
this is regular collision at <absо TEACH>.
this is regular collision at <absо TRSACT>.
Bump into node carrying same ori mark at cnode=<absо PERSON>
this is regular collision at <absо TT>.
Loop detected at cnode=<asp REC.REC-TEACH>

A word read in read-word: word=TAUGHT
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE3)
Only one rule selected = NOUN-PARSE3

==== Trees in CLIST ====
 node:S2 pos:(S) name:NIL range:(1 1) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS1 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S2 features:((SS)) trace:NIL
 node:NP8 pos:(NP) name:NIL range:(2 3) semsug:NIL
 parent:NIL features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
   --
   node:NC6 pos:(NC) name:NIL range:(2 2) semsug:NIL
   parent:NP8 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NS4 pos:(NS) name:NIL range:(2 2) semsug:NIL
     parent:NC6 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:DET3 pos:(DET) name:THE range:(2 2) semsug:NIL
       parent:NS4 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
   --
   node:NOUN5 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
   parent:NP8 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= (SUBJECT-RULE)
Only one rule selected = SUBJECT-RULE

==== Trees in CLIST ====
 node:S2 pos:(S) name:NIL range:(1 3) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS1 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S2 features:((SS)) trace:NIL
   --SUBJ--
   node:NP8 pos:(NP) name:NIL range:(2 3) semsug:NIL
   parent:S2 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
     --
     node:NC6 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP8 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS4 pos:(NS) name:NIL range:(2 2) semsug:NIL
```

```
      parent:NC6 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:DET3 pos:(DET) name:THE range:(2 2) semsug:NIL
        parent:NS4 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NOUN5 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
    parent:NP8 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.
-------- justify finished.as success ----------
an object read in from cash = V7
Come into rule-matching part....
Return from rule-matching routine. rules= (MAIN-VERB2 NP-VPP NP-VPP1)

come into go_on_several_rules. rules= (NP-VPP1 NP-VPP MAIN-VERB2)
start of branch in go_several_rules = (NP-VPP1)
Suggestion received: dir=CONNECT o1=<abso TEACHER.1> o2=<abso TAUGHT.2> o3=TO whichasp= 2
two-way collisions betw <abso TAUGHT.2> and <abso TEACHER.1> = 31
# of paths for these two-way collisions = 1502
call into secondary marker passing
Number of original three-way collisions = 1
number of collisions before H-heuristics = 1
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<abso TEACH> <abso PERSON> (<abso TAUGHT.2> I <abso TEACH> C_INV <asp TEACH.REC-TEACH> A_INV
<rel REC-TEACH> A_INV <asp REC.REC-TEACH> C_INV <abso PERSON> D <abso TEACH_PROFESSIONAL> D
<abso TEACHER> I <abso TEACHER.1>))

==== Trees in CLIST ====
 node:S10 pos:(S) name:NIL range:(1 3) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS11 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S10 features:((SS)) trace:NIL
   --SUBJ--
   node:NP12 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S10 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC13 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP12 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS14 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC13 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET15 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS14 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN16 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP12 features:((NOUN (NUM 3S))) trace:NIL
     --
     node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
     parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
       --SUBJ--
       node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
       parent:S17 features:((NP (DUMMY))) trace:NIL
       --
       node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
       parent:S17 features:((AUX (PASSIVE))) trace:NIL
       --
       node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
       parent:S17 features:((VP)) trace:NIL
         --
         node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
         parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
         --IOBJ--
         node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
         parent:VP20 features:((NP (TRACE))) trace:NP12
node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
  --SUBJ--
  node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
  parent:S17 features:((NP (DUMMY))) trace:NIL
  --
  node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S17 features:((AUX (PASSIVE))) trace:NIL
```

```
  --
  node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
  parent:S17 features:((VP)) trace:NIL
    --
    node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> TO 2)
    parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
    --IOBJ--
    node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
    parent:VP20 features:((NP (TRACE))) trace:NP12

beginning of NEXT_round.

Come into rule-matching part....
No rules matched.(just after NEXT_ROUND1)
-------- justify finished.as success ----------
End of one rule-process for=(NP-VPP1) merit=(<absо TEACH> <absо PERSON> (<absо TAUGHT.2> I <absо TEACH> C_INV
<asp TEACH.REC-TEACH> A_INV <rel REC-TEACH> A_INV <asp REC.REC-TEACH> C_INV <absо PERSON> D
<absо TEACH_PROFESSIONAL> D <absо TEACHER> I <absо TEACHER.1>))

start of branch in go_several_rules = (NP-VPP)
Suggestion received: dir=CONNECT o1=<absо TEACHER.1> o2=<absо TAUGHT.2> o3=OBJ whichasp= 2
two-way collisions betw <absо TAUGHT.2> and <absо TEACHER.1> = 31
# of paths for these two-way collisions = 1502
call into secondary marker passing
Number of original three-way collisions = 1
number of collisions before H-heuristics = 1
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1
Filter paths having too_abstract_concept....
removed: (<absо TAUGHT.2> I <absо TEACH> D <absо TRSACT> C_INV <asp TRSACT.ACTEE-TRSACT> A_INV <rel ACTEE-TRSACT>
A_INV <asp ACTEE.ACTEE-TRSACT> C_INV <absо THING> D <absо PHY_OBJECT> D <absо ANIMATE> D <absо ANIMAL> D
<absо HIGH_ANIMATE> D <absо PERSON> D <absо TEACH_PROFESSIONAL> D <absо TEACHER> I <absо TEACHER.1>)
Final paths selected in pickup_interp: no paths found.

start of branch in go_several_rules = (MAIN-VERB2)
Suggestion received: dir=CONNECT o1=<absо TEACHER.1> o2=<absо TAUGHT.2> o3=SUBJ whichasp= NIL
two-way collisions betw <absо TAUGHT.2> and <absо TEACHER.1> = 31
# of paths for these two-way collisions = 1502
call into secondary marker passing
Number of original three-way collisions = 3
number of collisions before H-heuristics = 3
drop collision at filter1: cnode=<rel ACTOR-ACT>
drop collision at filter1: cnode=<rel ACTOR-TRSACT>
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<absо TEACH> <absо TEACH_PROFESSIONAL> (<absо TAUGHT.2> I <absо TEACH> C_INV <asp TEACH.TEACHER-TEACH>
A_INV <rel TEACHER-TEACH> A_INV <asp TEACHER.TEACHER-TEACH> P_INV <absо TEACH_PROFESSIONAL> D <absо
TEACHER> I <absо TEACHER.1>)

==== Trees in CLIST ====
 node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S36 features:((SS)) trace:NIL
   --SUBJ--
   node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
   parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
     --
     node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
   --
   node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
   parent:S36 features:((AUX)) trace:NIL
   --
   node:VP44 pos:(VP) name:NIL range:(4 4) semsug:NIL
```

```
    parent:S36 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
      --
      node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> SUBJ NIL)
      parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))

Come into rule-matching part....
No rules matched.(just after NEXT_ROUND1)
-------- justify finished.as success ----------

End of one rule-process for=(MAIN-VERB2) merit=(<abso TEACH> <abso TEACH_PROFESSIONAL> (<abso TAUGHT.2> I <abso TEACH>
C_INV <asp TEACH.TEACHER-TEACH> A_INV <rel TEACHER-TEACH> A_INV <asp TEACHER.TEACHER-TEACH> P_INV
<abso TEACH_PROFESSIONAL> D <abso TEACHER> I <abso TEACHER.1>))

start of run parallel till one is found. # of branches=2
# of sem branches= 2  # of no sem branches= 0

Backed up branches into *backup-stack* = (NP-VPP1)
Branches to continue to run =  (MAIN-VERB2)

FINAL One branch chosen after parallel execution: (MAIN-VERB2)

Come into rule-matching part....
Return from rule-matching routine. rules= NIL

-------- justify finished.as success ----------
RRRRRRRRRead next word : BY

A word read in read-word: word=BY
Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : THE

A word read in read-word: word=THE
Come into rule-matching part....
Return from rule-matching routine. rules= (PARSE-DET)

Only one rule selected = PARSE-DET

==== Trees in CLIST ====
 node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
    --
    node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
    parent:S36 features:((SS)) trace:NIL
    --SUBJ--
    node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
    parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
       --
       node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
       parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
          parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
             --
             node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
             parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
       parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
    --
    node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
    parent:S36 features:((AUX)) trace:NIL
    --
    node:VP44 pos:(VP) name:NIL range:(4 4) semsug:NIL
    parent:S36 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
       --
       node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> SUBJ NIL)
       parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
 node:PREP45 pos:(PREP) name:BY range:(5 5) semsug:NIL
 parent:NIL features:((PREP)) trace:NIL
 node:NS47 pos:(NS) name:NIL range:(6 6) semsug:NIL
 parent:NIL features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:DET46 pos:(DET) name:THE range:(6 6) semsug:NIL
    parent:NS47 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
```

```
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : POOL

passing markers. instance=<absо POOL.3>
this is regular collision at <absо PLACE>.
this is regular collision at <absо INANIMATE>.
this is regular collision at <asp ACT.ACTOR-ACT>.
Loop detected at cnode=<asp ACT.INSTR-ACT>
this is regular collision at <asp ACT.ACCOMPANIER-ACT>.
this is regular collision at <absо TRSACT>.
Loop detected at cnode=<asp POS.IDENTITY-REL>
Bump into node carrying same ori mark at cnode=<asp ACT.PURPOSE-ACT>
this is regular collision at <rel AG-EXAMINE>.

A word read in read-word: word=POOL
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE1)

Only one rule selected = NOUN-PARSE1

==== Trees in CLIST ====
 node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
    --
    node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
    parent:S36 features:((SS)) trace:NIL
    --SUBJ--
    node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
    parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
       --
       node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
       parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
          parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
             --
             node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
             parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
       parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
    --
    node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
    parent:S36 features:((AUX)) trace:NIL
    --
    node:VP44 pos:(VP) name:NIL range:(4 4) semsug:NIL
    parent:S36 features:((VP (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
       --
       node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> SUBJ NIL)
       parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
 node:PREP45 pos:(PREP) name:BY range:(5 5) semsug:NIL
 parent:NIL features:((PREP)) trace:NIL
 node:NC49 pos:(NC) name:NIL range:(6 6) semsug:NIL
 parent:NIL features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NS47 pos:(NS) name:NIL range:(6 6) semsug:NIL
    parent:NC49 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:DET46 pos:(DET) name:THE range:(6 6) semsug:NIL
       parent:NS47 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
 node:NOUN48 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
 parent:NIL features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : PASSED

passing markers. instance=<absо PASSED.4>
this is regular collision at <absо PASS>.
this is regular collision at <absо TRSACT>.
this is regular collision at <absо PHY_OBJECT>.
Loop detected at cnode=<asp ACTOR.ACTOR-ACT>
Loop detected at cnode=<asp ACCOMPANIER.ACCOMPANIER-ACT>
Bump into node carrying same ori mark at cnode=<absо TT>
this is regular collision at <rel PAT-WRAP>.
Loop detected at cnode=<absо PLACE>
```

```
Come into syntax-semantics.

A word read in read-word: word=PASSED
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE3)

Only one rule selected = NOUN-PARSE3

==== Trees in CLIST ====
 node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S36 features:((SS)) trace:NIL
   --SUBJ--
   node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
   parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
     --
     node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
   --
   node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
   parent:S36 features:((AUX)) trace:NIL
   --
   node:VP44 pos:(VP) name:NIL range:(4 4) semsug:NIL
   parent:S36 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
     --
     node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> SUBJ NIL)
     parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
 node:PREP45 pos:(PREP) name:BY range:(5 5) semsug:NIL
 parent:NIL features:((PREP)) trace:NIL
 node:NP51 pos:(NP) name:NIL range:(6 7) semsug:NIL
 parent:NIL features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
   --
   node:NC49 pos:(NC) name:NIL range:(6 6) semsug:NIL
   parent:NP51 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NS47 pos:(NS) name:NIL range:(6 6) semsug:NIL
     parent:NC49 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:DET46 pos:(DET) name:THE range:(6 6) semsug:NIL
       parent:NS47 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
   --
   node:NOUN48 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
   parent:NP51 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= (build-PP)

Only one rule selected = build-PP

==== Trees in CLIST ====
 node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S36 features:((SS)) trace:NIL
   --SUBJ--
   node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
   parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
     --
     node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
```

```
  --
  node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S36 features:((AUX)) trace:NIL
  --
  node:VP44 pos:(VP) name:NIL range:(4 4) semsug:NIL
  parent:S36 features:((VP (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
    --
    node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> SUBJ NIL)
    parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
node:PP52 pos:(PP) name:NIL range:(5 7) semsug:NIL
parent:NIL features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
  --
  node:PREP45 pos:(PREP) name:BY range:(5 5) semsug:NIL
  parent:PP52 features:((PREP)) trace:NIL
  --
  node:NP51 pos:(NP) name:NIL range:(6 7) semsug:NIL
  parent:PP52 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
    --
    node:NC49 pos:(NC) name:NIL range:(6 6) semsug:NIL
    parent:NP51 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NS47 pos:(NS) name:NIL range:(6 6) semsug:NIL
      parent:NC49 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:DET46 pos:(DET) name:THE range:(6 6) semsug:NIL
        parent:NS47 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NOUN48 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
    parent:NP51 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= (VP-PP)

Only one rule selected = VP-PP
Suggestion received: dir=CONNECT o1=<absо TAUGHT.2> o2=<absо POOL.3> o3=BY whichasp= NIL
two-way collisions betw <absо POOL.3> and <absо TAUGHT.2> = 27
# of paths for these two-way collisions = 562
call into secondary marker passing
Number of original three-way collisions = 2
number of collisions before H-heuristics = 2
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<absо PLACE> <absо TEACH> (<absо POOL.3> I <absо POOL> D <absо PLACE> C_INV <asp LOC.LOC-TEACH> A_INV
<rel LOC-TEACH> A_INV <asp TEACH.LOC-TEACH> C_INV <absо TEACH> I <absо TAUGHT.2>))

==== Trees in CLIST ====
node:S36 pos:(S) name:NIL range:(1 4) semsug:NIL
parent:NIL features:((S (MAIN))) trace:NIL
  --
  node:SS37 pos:(SS) name:@ range:(1 1) semsug:NIL
  parent:S36 features:((SS)) trace:NIL
  --SUBJ--
  node:NP38 pos:(NP) name:NIL range:(2 3) semsug:NIL
  parent:S36 features:((NP (NUM 3S) (MODIFIABLE) (SUBJ))) trace:NIL
    --
    node:NC39 pos:(NC) name:NIL range:(2 2) semsug:NIL
    parent:NP38 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NS40 pos:(NS) name:NIL range:(2 2) semsug:NIL
      parent:NC39 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:DET41 pos:(DET) name:THE range:(2 2) semsug:NIL
        parent:NS40 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NOUN42 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
    parent:NP38 features:((NOUN (NUM 3S))) trace:NIL
  --
  node:AUX43 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S36 features:((AUX)) trace:NIL
  --
  node:VP44 pos:(VP) name:NIL range:(4 7) semsug:NIL
  parent:S36 features:((VP (DATIVE-PREP TO) (DATIVE) (TRANSITIVE) (MAIN) (TENSE PAST) (EN)))
    --
    node:V35 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> SUBJ NIL)
    parent:VP44 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
    --
    node:PP52 pos:(PP) name:NIL range:(5 7) semsug:NIL
```

```
    parent:VP44 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
      --
      node:PREP45 pos:(PREP) name:BY range:(5 5) semsug:NIL
      parent:PP52 features:((PREP)) trace:NIL
      --
      node:NP51 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
      parent:PP52 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
        --
        node:NC49 pos:(NC) name:NIL range:(6 6) semsug:NIL
        parent:NP51 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:NS47 pos:(NS) name:NIL range:(6 6) semsug:NIL
          parent:NC49 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
            --
            node:DET46 pos:(DET) name:THE range:(6 6) semsug:NIL
            parent:NS47 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:NOUN48 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
        parent:NP51 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
an object read in from cash = V50
Come into rule-matching part....
Return from rule-matching routine. rules= (NP-VPP)

Only one rule selected = NP-VPP
Suggestion received: dir=CONNECT o1=<abso POOL.3> o2=<abso PASSED.4> o3=OBJ whichasp= 2
two-way collisions betw <abso PASSED.4> and <abso POOL.3> = 17
# of paths for these two-way collisions = 356
call into secondary marker passing
Number of original three-way collisions = 1
number of collisions before H-heuristics = 1
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1
Filter paths having too_abstract_concept....
Final paths selected in pickup_interp: no paths found.

Come into backup...
Come into backup...original rule = (NP-VPP1)

==== Trees in CLIST ====
 node:S10 pos:(S) name:NIL range:(1 3) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS11 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S10 features:((SS)) trace:NIL
   --SUBJ--
   node:NP12 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S10 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC13 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP12 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS14 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC13 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET15 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS14 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN16 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP12 features:((NOUN (NUM 3S))) trace:NIL
     --
     node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
     parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
       --SUBJ--
       node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
       parent:S17 features:((NP (DUMMY))) trace:NIL
       --
       node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
       parent:S17 features:((AUX (PASSIVE))) trace:NIL
       --
       node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
       parent:S17 features:((VP)) trace:NIL
         --
         node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
```

```
          parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) (NUM 1S 2S 3S 1P 2P 3P)))
          --IOBJ--
          node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
          parent:VP20 features:((NP (TRACE))) trace:NP12
 node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
 parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
   --SUBJ--
   node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
   parent:S17 features:((NP (DUMMY))) trace:NIL
   --
   node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
   parent:S17 features:((AUX (PASSIVE))) trace:NIL
   --
   node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
   parent:S17 features:((VP)) trace:NIL
     --
     node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
     parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
     --IOBJ--
     node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
     parent:VP20 features:((NP (TRACE))) trace:NP12

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
Save backup point for sec clause:(S17 S10)

A word read in read-word: word=BY
No rules are found.

-------- justify finished.as success ----------
A word read in read-word: word=THE
Come into rule-matching part....
Return from rule-matching routine. rules= (PARSE-DET)

Only one rule selected = PARSE-DET
This rule executed.

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
A word read in read-word: word=POOL
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE1)

Only one rule selected = NOUN-PARSE1
This rule executed.

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
A word read in read-word: word=PASSED
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE3)

Only one rule selected = NOUN-PARSE3
This rule executed.

Come into rule-matching part....
Return from rule-matching routine. rules=build-PP
Only one rule selected = build-PP

==== Trees in CLIST ====
node:S10 pos:(S) name:NIL range:(1 3) semsug:NIL
parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS11 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S10 features:((SS)) trace:NIL
   --SUBJ--
   node:NP12 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S10 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC13 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP12 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
```

```
        node:NS14 pos:(NS) name:NIL range:(2 2) semsug:NIL
        parent:NC13 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:DET15 pos:(DET) name:THE range:(2 2) semsug:NIL
          parent:NS14 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NOUN16 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
      parent:NP12 features:((NOUN (NUM 3S))) trace:NIL
      --
      node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
      parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
        --SUBJ--
        node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
        parent:S17 features:((NP (DUMMY))) trace:NIL
        --
        node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
        parent:S17 features:((AUX (PASSIVE))) trace:NIL
        --
        node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
        parent:S17 features:((VP)) trace:NIL
          --
          node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
          parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
          --IOBJ--
          node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
          parent:VP20 features:((NP (TRACE))) trace:NP12
node:S17 pos:(S) name:NIL range:(4 NIL) semsug:NIL
parent:NP12 features:((S (SEC) (STRUC-OK))) trace:NIL
  --SUBJ--
  node:NP18 pos:(NP) name:NIL range:NIL semsug:NIL
  parent:S17 features:((NP (DUMMY))) trace:NIL
  --
  node:AUX19 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S17 features:((AUX (PASSIVE))) trace:NIL
  --
  node:VP20 pos:(VP) name:NIL range:(4 NIL) semsug:NIL
  parent:S17 features:((VP)) trace:NIL
    --
    node:V9 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
    parent:VP20 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) )) trace:NIL
    --IOBJ--
    node:NP21 pos:(NP) name:NIL range:NIL semsug:NIL
    parent:VP20 features:((NP (TRACE))) trace:NP12
node:PP78 pos:(PP) name:NIL range:(5 7) semsug:NIL
parent:NIL features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
  --
  node:PREP71 pos:(PREP) name:BY range:(5 5) semsug:NIL
  parent:PP78 features:((PREP)) trace:NIL
  --
  node:NP77 pos:(NP) name:NIL range:(6 7) semsug:NIL
  parent:PP78 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
    --
    node:NC75 pos:(NC) name:NIL range:(6 6) semsug:NIL
    parent:NP77 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NS73 pos:(NS) name:NIL range:(6 6) semsug:NIL
      parent:NC75 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:DET72 pos:(DET) name:THE range:(6 6) semsug:NIL
        parent:NS73 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NOUN74 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
    parent:NP77 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= (VP-PP PASSIVE-BY)

come into go_on_several_rules. rules= (PASSIVE-BY VP-PP)
start of branch in go_several_rules = (PASSIVE-BY)
Suggestion received: dir=CONNECT o1=<abso TAUGHT.2> o2=<abso POOL.3> o3=SUBJ whichasp= 2
two-way collisions betw <abso POOL.3> and <abso TAUGHT.2> = 27
# of paths for these two-way collisions = 562
call into secondary marker passing
Number of original three-way collisions = 0
No three way collision found.....
This branch is removed.

start of branch in go_several_rules = (VP-PP)
Suggestion received: dir=CONNECT o1=<abso TAUGHT.2> o2=<abso POOL.3> o3=BY whichasp= NIL
two-way collisions betw <abso POOL.3> and <abso TAUGHT.2> = 27
# of paths for these two-way collisions = 562
```

```
call into secondary marker passing
Number of original three-way collisions = 2
number of collisions before H-heuristics = 2
drop collision at filter1: cnode=<rel LOC-ACT>
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<abso PLACE> <abso TEACH> (<abso POOL.3> I <abso POOL> D <abso PLACE> C_INV <asp LOC.LOC-TEACH> A_INV
<rel LOC-TEACH> A_INV <asp TEACH.LOC-TEACH> C_INV <abso TEACH> I <abso TAUGHT.2>))

==== Trees in CLIST ====
 node:S107 pos:(S) name:NIL range:(1 3) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS108 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S107 features:((SS)) trace:NIL
   --SUBJ--
   node:NP109 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S107 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC110 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP109 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS111 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC110 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET112 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS111 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN113 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP109 features:((NOUN (NUM 3S))) trace:NIL
     --
     node:S114 pos:(S) name:NIL range:(4 NIL) semsug:NIL
     parent:NP109 features:((S (SEC) (STRUC-OK))) trace:NIL
       --SUBJ--
       node:NP115 pos:(NP) name:NIL range:NIL semsug:NIL
       parent:S114 features:((NP (DUMMY))) trace:NIL
       --
       node:AUX116 pos:(AUX) name:NIL range:NIL semsug:NIL
       parent:S114 features:((AUX (PASSIVE))) trace:NIL
       --
       node:VP117 pos:(VP) name:NIL range:(4 7) semsug:NIL
       parent:S114 features:((VP)) trace:NIL
         --
         node:V118 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
         parent:VP117 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
         --IOBJ--
         node:NP119 pos:(NP) name:NIL range:NIL semsug:NIL
         parent:VP117 features:((NP (TRACE))) trace:NP12
         --
         node:PP100 pos:(PP) name:NIL range:(5 7) semsug:NIL
         parent:VP117 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
           --
           node:PREP101 pos:(PREP) name:BY range:(5 5) semsug:NIL
           parent:PP100 features:((PREP)) trace:NIL
           --
           node:NP102 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
           parent:PP100 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
             --
             node:NC103 pos:(NC) name:NIL range:(6 6) semsug:NIL
             parent:NP102 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
               --
               node:NS104 pos:(NS) name:NIL range:(6 6) semsug:NIL
               parent:NC103 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
                 --
                 node:DET105 pos:(DET) name:THE range:(6 6) semsug:NIL
                 parent:NS104 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
             --
             node:NOUN106 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
             parent:NP102 features:((NOUN (NUM 3S))) trace:NIL
 node:S114 pos:(S) name:NIL range:(4 NIL) semsug:NIL
 parent:NP109 features:((S (SEC) (STRUC-OK))) trace:NIL
   --SUBJ--
   node:NP115 pos:(NP) name:NIL range:NIL semsug:NIL
   parent:S114 features:((NP (DUMMY))) trace:NIL
   --
   node:AUX116 pos:(AUX) name:NIL range:NIL semsug:NIL
   parent:S114 features:((AUX (PASSIVE))) trace:NIL
```

```
--
node:VP117 pos:(VP) name:NIL range:(4 7) semsug:NIL
parent:S114 features:((VP)) trace:NIL
  --
  node:V118 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
  parent:VP117 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
  --IOBJ--
  node:NP119 pos:(NP) name:NIL range:NIL semsug:NIL
  parent:VP117 features:((NP (TRACE))) trace:NP12
  --
  node:PP100 pos:(PP) name:NIL range:(5 7) semsug:NIL
  parent:VP117 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
    --
    node:PREP101 pos:(PREP) name:BY range:(5 5) semsug:NIL
    parent:PP100 features:((PREP)) trace:NIL
    --
    node:NP102 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
    parent:PP100 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
      --
      node:NC103 pos:(NC) name:NIL range:(6 6) semsug:NIL
      parent:NP102 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:NS104 pos:(NS) name:NIL range:(6 6) semsug:NIL
        parent:NC103 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:DET105 pos:(DET) name:THE range:(6 6) semsug:NIL
          parent:NS104 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NOUN106 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
      parent:NP102 features:((NOUN (NUM 3S))) trace:NIL

beginning of NEXT_round.

Come into rule-matching part....
No rules matched.(just after NEXT_ROUND1)
-------- justify finished.as success ----------
End of one rule-process for=(VP-PP) merit=(<abso PLACE> <abso TEACH> (<abso POOL.3> I <abso POOL> D <abso PLACE> C_INV
<asp LOC.LOC-TEACH> A_INV <rel LOC-TEACH> A_INV <asp TEACH.LOC-TEACH> C_INV <abso TEACH> I <abso TAUGHT.2>))

start of run parallel till one is found. # of branches=1

# of sem branches= 1  # of no sem branches= 0

Backed up branches into *backup-stack* =
Branches to continue to run =  (VP-PP)

FINAL One branch chosen after parallel execution: (VP-PP)
only one branch chosen in parallel.

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
Save backup point for sec clause:(S114 S107)

an object read in from cash = V120
Come into rule-matching part....
Return from rule-matching routine. rules= (NP-VPP)

Only one rule selected = NP-VPP
Suggestion received: dir=CONNECT o1=<abso POOL.3> o2=<abso PASSED.4> o3=OBJ whichasp= 2
two-way collisions betw <abso PASSED.4> and <abso POOL.3> = 17
# of paths for these two-way collisions = 356
call into secondary marker passing
Number of original three-way collisions = 1
number of collisions before H-heuristics = 1
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation  filtering = 1
Number of full paths after hierarchy of end absos = 1
Filter paths having too_abstract_concept....
Final paths selected in pickup_interp: no paths found.

Come into backup...

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
```

```
an object read in from cash = V141
Come into rule-matching part....
Return from rule-matching routine. rules= (MAIN-VERB2)

Only one rule selected = MAIN-VERB2
Suggestion received: dir=CONNECT o1=<absо TEACHER.1> o2=<absо PASSED.4> o3=SUBJ whichasp= NIL
two-way collisions betw <absо PASSED.4> and <absо TEACHER.1> = 30
# of paths for these two-way collisions = 1048
call into secondary marker passing
Number of original three-way collisions = 3
number of collisions before H-heuristics = 3
drop collision at filter1: cnode=<rel ACTOR-ACT>
drop collision at filter1: cnode=<rel ACTOR-TRSACT>
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<absо PASS> <absо ANIMAL> (<absо PASSED.4> I <absо PASS> C_INV <asp PASS.PASSER-PASS> A_INV <rel PASSER-PASS>
A_INV <asp PASSER.PASSER-PASS> C_INV <absо ANIMAL> D <absо HIGH_ANIMATE> D <absо PERSON> D
<absо TEACH_PROFESSIONAL> D <absо TEACHER> I <absо TEACHER.1>))

==== Trees in CLIST ====
 node:S121 pos:(S) name:NIL range:(1 8) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
   --
   node:SS122 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S121 features:((SS)) trace:NIL
   --SUBJ--
   node:NP123 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S121 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC124 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP123 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS125 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC124 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET126 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS125 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN127 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP123 features:((NOUN (NUM 3S))) trace:NIL
     --
     node:S128 pos:(S) name:NIL range:(4 NIL) semsug:NIL
     parent:NP123 features:((S (SEC) (STRUC-OK) (CLOSED))) trace:NIL
       --SUBJ--
       node:NP129 pos:(NP) name:NIL range:NIL semsug:NIL
       parent:S128 features:((NP (DUMMY))) trace:NIL
       --
       node:AUX130 pos:(AUX) name:NIL range:NIL semsug:NIL
       parent:S128 features:((AUX (PASSIVE))) trace:NIL
       --
       node:VP131 pos:(VP) name:NIL range:(4 7) semsug:NIL
       parent:S128 features:((VP)) trace:NIL
         --
         node:V132 pos:(V) name:TAUGHT range:(4 4) semsug:(<absо TEACHER.1> <absо TAUGHT.2> TO 2)
         parent:VP131 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
         --IOBJ--
         node:NP133 pos:(NP) name:NIL range:NIL semsug:NIL
         parent:VP131 features:((NP (TRACE))) trace:NP12
         --
         node:PP134 pos:(PP) name:NIL range:(5 7) semsug:NIL
         parent:VP131 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
           --
           node:PREP135 pos:(PREP) name:BY range:(5 5) semsug:NIL
           parent:PP134 features:((PREP)) trace:NIL
           --
           node:NP136 pos:(NP) name:NIL range:(6 7) semsug:(<absо TAUGHT.2> <absо POOL.3> BY NIL)
           parent:PP134 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
             --
             node:NC137 pos:(NC) name:NIL range:(6 6) semsug:NIL
             parent:NP136 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
               --
               node:NS138 pos:(NS) name:NIL range:(6 6) semsug:NIL
               parent:NC137 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
                 --
                 node:DET139 pos:(DET) name:THE range:(6 6) semsug:NIL
                 parent:NS138 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
             --
```

```
                node:NOUN140 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
                parent:NP136 features:((NOUN (NUM 3S))) trace:NIL
    --
    node:AUX147 pos:(AUX) name:NIL range:NIL semsug:NIL
    parent:S121 features:((AUX)) trace:NIL
    --
    node:VP148 pos:(VP) name:NIL range:(8 8) semsug:NIL
    parent:S121 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (TRANSITIVE) (EN) (TENSE PAST) (MAIN))) trace:NIL
      --
      node:V141 pos:(V) name:PASSED range:(8 8) semsug:(<absо TEACHER.1> <absо PASSED.4> SUBJ NIL)
      parent:VP148 features:((V (MAIN) (TENSE PAST) (EN) (TRANSITIVE) (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : THE

A word read in read-word: word=THE
Come into rule-matching part....
Return from rule-matching routine. rules= (PARSE-DET)

Only one rule selected = PARSE-DET
This rule executed.

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : TEST

passing markers. instance=<absо TEST.5>
this is regular collision at <absо GENERAL_TEST>.
Bump into node carrying same ori mark at cnode=<absо STATE>
this is regular collision at <rel STEAL-ER>.
this is regular collision at <rel AG-ACQUIRE>.
this is regular collision at <rel AG-DISCUSS>.

A word read in read-word: word=TEST
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE1)

Only one rule selected = NOUN-PARSE1
This rule executed.

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
RRRRRRRRRead next word : @.

A word read in read-word: word=@.
Come into rule-matching part....
Return from rule-matching routine. rules= (NOUN-PARSE3)

Only one rule selected = NOUN-PARSE3

==== Trees in CLIST ====
 node:S121 pos:(S) name:NIL range:(1 8) semsug:NIL
 parent:NIL features:((S (MAIN))) trace:NIL
    --
    node:SS122 pos:(SS) name:@ range:(1 1) semsug:NIL
    parent:S121 features:((SS)) trace:NIL
    --SUBJ--
    node:NP123 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
    parent:S121 features:((NP (NUM 3S) (SUBJ))) trace:NIL
      --
      node:NC124 pos:(NC) name:NIL range:(2 2) semsug:NIL
      parent:NP123 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:NS125 pos:(NS) name:NIL range:(2 2) semsug:NIL
        parent:NC124 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:DET126 pos:(DET) name:THE range:(2 2) semsug:NIL
          parent:NS125 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NOUN127 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
      parent:NP123 features:((NOUN (NUM 3S))) trace:NIL
```

```
    --
    node:S128 pos:(S) name:NIL range:(4 NIL) semsug:NIL
    parent:NP123 features:((S (SEC) (STRUC-OK) (CLOSED))) trace:NIL
      --SUBJ--
      node:NP129 pos:(NP) name:NIL range:NIL semsug:NIL
      parent:S128 features:((NP (DUMMY))) trace:NIL
      --
      node:AUX130 pos:(AUX) name:NIL range:NIL semsug:NIL
      parent:S128 features:((AUX (PASSIVE))) trace:NIL
      --
      node:VP131 pos:(VP) name:NIL range:(4 7) semsug:NIL
      parent:S128 features:((VP)) trace:NIL
        --
        node:V132 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
        parent:VP131 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
        --IOBJ--
        node:NP133 pos:(NP) name:NIL range:NIL semsug:NIL
        parent:VP131 features:((NP (TRACE))) trace:NP12
        --
        node:PP134 pos:(PP) name:NIL range:(5 7) semsug:NIL
        parent:VP131 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
          --
          node:PREP135 pos:(PREP) name:BY range:(5 5) semsug:NIL
          parent:PP134 features:((PREP)) trace:NIL
          --
          node:NP136 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
          parent:PP134 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
            --
            node:NC137 pos:(NC) name:NIL range:(6 6) semsug:NIL
            parent:NP136 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
              --
              node:NS138 pos:(NS) name:NIL range:(6 6) semsug:NIL
              parent:NC137 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
                --
                node:DET139 pos:(DET) name:THE range:(6 6) semsug:NIL
                parent:NS138 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
            --
            node:NOUN140 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
            parent:NP136 features:((NOUN (NUM 3S))) trace:NIL
  --
  node:AUX147 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S121 features:((AUX)) trace:NIL
  --
  node:VP148 pos:(VP) name:NIL range:(8 8) semsug:NIL
  parent:S121 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (TRANSITIVE) (EN) (TENSE PAST) (MAIN))) trace:NIL
    --
    node:V141 pos:(V) name:PASSED range:(8 8) semsug:(<abso TEACHER.1> <abso PASSED.4> SUBJ NIL)
    parent:VP148 features:((V (MAIN) (TENSE PAST) (EN) (TRANSITIVE) (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
node:NP154 pos:(NP) name:NIL range:(9 10) semsug:NIL
parent:NIL features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
  --
  node:NC152 pos:(NC) name:NIL range:(9 9) semsug:NIL
  parent:NP154 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NS150 pos:(NS) name:NIL range:(9 9) semsug:NIL
    parent:NC152 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:DET149 pos:(DET) name:THE range:(9 9) semsug:NIL
      parent:NS150 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
  --
  node:NOUN151 pos:(NOUN) name:TEST range:(10 10) semsug:NIL
  parent:NP154 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= (OBJECT-RULE)

Only one rule selected = OBJECT-RULE
Suggestion received: dir=CONNECT o1=<abso PASSED.4> o2=<abso TEST.5> o3=OBJ whichasp= NIL
two-way collisions betw <abso TEST.5> and <abso PASSED.4> = 8
# of paths for these two-way collisions = 171
call into secondary marker passing
# of collisions after H-heuristic: 1
Number of full paths bef # rel filtering = 1
number of full paths after num. relation filtering = 1
Number of full paths after hierarchy of end absos = 1

Final paths selected in pickup_interp.
<<>> (<abso GENERAL_TEST> <abso PASS> (<abso TEST.5> I <abso TEST> D <abso GENERAL_TEST> C_INV <asp
PASSEE.PASSEE-PASS> A_INV <rel PASSEE-PASS> A_INV <asp PASS.PASSEE-PASS> C_INV <abso PASS> I <abso PASSED.4>))

==== Trees in CLIST ====
```

```
node:S121 pos:(S) name:NIL range:(1 8) semsug:NIL
parent:NIL features:((S (MAIN) (STRUC-OK))) trace:NIL
  --
  node:SS122 pos:(SS) name:@ range:(1 1) semsug:NIL
  parent:S121 features:((SS)) trace:NIL
  --SUBJ--
  node:NP123 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
  parent:S121 features:((NP (NUM 3S) (SUBJ))) trace:NIL
    --
    node:NC124 pos:(NC) name:NIL range:(2 2) semsug:NIL
    parent:NP123 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NS125 pos:(NS) name:NIL range:(2 2) semsug:NIL
      parent:NC124 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:DET126 pos:(DET) name:THE range:(2 2) semsug:NIL
        parent:NS125 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --
    node:NOUN127 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
    parent:NP123 features:((NOUN (NUM 3S))) trace:NIL
    --
    node:S128 pos:(S) name:NIL range:(4 NIL) semsug:NIL
    parent:NP123 features:((S (SEC) (STRUC-OK) (CLOSED))) trace:NIL
      --SUBJ--
      node:NP129 pos:(NP) name:NIL range:NIL semsug:NIL
      parent:S128 features:((NP (DUMMY))) trace:NIL
      --
      node:AUX130 pos:(AUX) name:NIL range:NIL semsug:NIL
      parent:S128 features:((AUX (PASSIVE))) trace:NIL
      --
      node:VP131 pos:(VP) name:NIL range:(4 7) semsug:NIL
      parent:S128 features:((VP)) trace:NIL
        --
        node:V132 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
        parent:VP131 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO) ))
        --IOBJ--
        node:NP133 pos:(NP) name:NIL range:NIL semsug:NIL
        parent:VP131 features:((NP (TRACE))) trace:NP12
        --
        node:PP134 pos:(PP) name:NIL range:(5 7) semsug:NIL
        parent:VP131 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
          --
          node:PREP135 pos:(PREP) name:BY range:(5 5) semsug:NIL
          parent:PP134 features:((PREP)) trace:NIL
          --
          node:NP136 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
          parent:PP134 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
            --
            node:NC137 pos:(NC) name:NIL range:(6 6) semsug:NIL
            parent:NP136 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
              --
              node:NS138 pos:(NS) name:NIL range:(6 6) semsug:NIL
              parent:NC137 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
                --
                node:DET139 pos:(DET) name:THE range:(6 6) semsug:NIL
                parent:NS138 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
            --
            node:NOUN140 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
            parent:NP136 features:((NOUN (NUM 3S))) trace:NIL
  --
  node:AUX147 pos:(AUX) name:NIL range:NIL semsug:NIL
  parent:S121 features:((AUX)) trace:NIL
  --
  node:VP148 pos:(VP) name:NIL range:(8 10) semsug:NIL
  parent:S121 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (TRANSITIVE) (EN) (TENSE PAST) (MAIN))) trace:NIL
    --
    node:V141 pos:(V) name:PASSED range:(8 8) semsug:(<abso TEACHER.1> <abso PASSED.4> SUBJ NIL)
    parent:VP148 features:((V (MAIN) (TENSE PAST) (EN) (TRANSITIVE) (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --OBJ--
    node:NP154 pos:(NP) name:NIL range:(9 10) semsug:(<abso PASSED.4> <abso TEST.5> OBJ NIL)
    parent:VP148 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
      --
      node:NC152 pos:(NC) name:NIL range:(9 9) semsug:NIL
      parent:NP154 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:NS150 pos:(NS) name:NIL range:(9 9) semsug:NIL
        parent:NC152 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:DET149 pos:(DET) name:THE range:(9 9) semsug:NIL
          parent:NS150 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
```

```
        node:NOUN151 pos:(NOUN) name:TEST range:(10 10) semsug:NIL
        parent:NP154 features:((NOUN (NUM 3S))) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------
an object read in from cash = FPUNC153
Come into rule-matching part....
Return from rule-matching routine. rules= (S-CLOSE-AFF)

Only one rule selected = S-CLOSE-AFF

==== Trees in CLIST ====
; This tree is the final syntactic representation of the input sentence.
 node:S121 pos:(S) name:NIL range:(1 11) semsug:NIL
 parent:NIL features:((S (MAIN) (STRUC-OK))) trace:NIL
   --
   node:SS122 pos:(SS) name:@ range:(1 1) semsug:NIL
   parent:S121 features:((SS)) trace:NIL
   --SUBJ--
   node:NP123 pos:(NP) name:NIL range:(2 NIL) semsug:NIL
   parent:S121 features:((NP (NUM 3S) (SUBJ))) trace:NIL
     --
     node:NC124 pos:(NC) name:NIL range:(2 2) semsug:NIL
     parent:NP123 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
       --
       node:NS125 pos:(NS) name:NIL range:(2 2) semsug:NIL
       parent:NC124 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
         --
         node:DET126 pos:(DET) name:THE range:(2 2) semsug:NIL
         parent:NS125 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
     --
     node:NOUN127 pos:(NOUN) name:TEACHER range:(3 3) semsug:NIL
     parent:NP123 features:((NOUN (NUM 3S))) trace:NIL
     --
     node:S128 pos:(S) name:NIL range:(4 NIL) semsug:NIL
     parent:NP123 features:((S (SEC) (STRUC-OK) (CLOSED))) trace:NIL
       --SUBJ--
       node:NP129 pos:(NP) name:NIL range:NIL semsug:NIL
       parent:S128 features:((NP (DUMMY))) trace:NIL
       --
       node:AUX130 pos:(AUX) name:NIL range:NIL semsug:NIL
       parent:S128 features:((AUX (PASSIVE))) trace:NIL
       --
       node:VP131 pos:(VP) name:NIL range:(4 7) semsug:NIL
       parent:S128 features:((VP)) trace:NIL
         --
         node:V132 pos:(V) name:TAUGHT range:(4 4) semsug:(<abso TEACHER.1> <abso TAUGHT.2> TO 2)
         parent:VP131 features:((V (EN) (TENSE PAST) (MAIN) (TRANSITIVE) (DATIVE) (DATIVE-PREP TO)))
         --IOBJ--
         node:NP133 pos:(NP) name:NIL range:NIL semsug:NIL
         parent:VP131 features:((NP (TRACE))) trace:NP12
         --
         node:PP134 pos:(PP) name:NIL range:(5 7) semsug:NIL
         parent:VP131 features:((PP (MODIFIABLE) (NUM 3S))) trace:NIL
           --
           node:PREP135 pos:(PREP) name:BY range:(5 5) semsug:NIL
           parent:PP134 features:((PREP)) trace:NIL
           --
           node:NP136 pos:(NP) name:NIL range:(6 7) semsug:(<abso TAUGHT.2> <abso POOL.3> BY NIL)
           parent:PP134 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
             --
             node:NC137 pos:(NC) name:NIL range:(6 6) semsug:NIL
             parent:NP136 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
               --
               node:NS138 pos:(NS) name:NIL range:(6 6) semsug:NIL
               parent:NC137 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
                 --
                 node:DET139 pos:(DET) name:THE range:(6 6) semsug:NIL
                 parent:NS138 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
             --
             node:NOUN140 pos:(NOUN) name:POOL range:(7 7) semsug:NIL
             parent:NP136 features:((NOUN (NUM 3S))) trace:NIL
   --
   node:AUX147 pos:(AUX) name:NIL range:NIL semsug:NIL
   parent:S121 features:((AUX)) trace:NIL
   --
   node:VP148 pos:(VP) name:NIL range:(8 10) semsug:NIL
   parent:S121 features:((VP (NUM 1S 2S 3S 1P 2P 3P) (TRANSITIVE) (EN) (TENSE PAST) (MAIN))) trace:NIL
```

```
    --
    node:V141 pos:(V) name:PASSED range:(8 8) semsug:(<abso TEACHER.1> <abso PASSED.4> SUBJ NIL)
    parent:VP148 features:((V (MAIN) (TENSE PAST) (EN) (TRANSITIVE) (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
    --OBJ--
    node:NP154 pos:(NP) name:NIL range:(9 10) semsug:(<abso PASSED.4> <abso TEST.5> OBJ NIL)
    parent:VP148 features:((NP (NUM 3S) (MODIFIABLE))) trace:NIL
      --
      node:NC152 pos:(NC) name:NIL range:(9 9) semsug:NIL
      parent:NP154 features:((NC (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
        --
        node:NS150 pos:(NS) name:NIL range:(9 9) semsug:NIL
        parent:NC152 features:((NS (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
          --
          node:DET149 pos:(DET) name:THE range:(9 9) semsug:NIL
          parent:NS150 features:((DET (NUM 1S 2S 3S 1P 2P 3P))) trace:NIL
      --
      node:NOUN151 pos:(NOUN) name:TEST range:(10 10) semsug:NIL
      parent:NP154 features:((NOUN (NUM 3S))) trace:NIL
  --
  node:FPUNC153 pos:(FPUNC) name:@. range:(11 11) semsug:NIL
  parent:S121 features:((FPUNC)) trace:NIL

Come into rule-matching part....
Return from rule-matching routine. rules= NIL
No rules are found.

-------- justify finished.as success ----------

processing one sentence has finished.

> (print_semi)

Semantic Interpretation ----->
;;; This is a set of knowledge base paths found during the parsing.

(<abso TEST.5> I <abso TEST> D <abso GENERAL_TEST> C_INV <asp PASSEE.PASSEE-PASS> A_INV
<rel PASSEE-PASS> A_INV <asp PASS.PASSEE-PASS> C_INV <abso PASS> I <abso PASSED.4>)

(<abso PASSED.4> I <abso PASS> C_INV <asp PASS.PASSER-PASS> A_INV <rel PASSER-PASS> A_INV
<asp PASSER.PASSER-PASS> C_INV <abso ANIMAL> D <abso HIGH_ANIMATE> D <abso PERSON> D
<abso TEACH_PROFESSIONAL> D <abso TEACHER> I <abso TEACHER.1>)

(<abso POOL.3> I <abso POOL> D <abso PLACE> C_INV <asp LOC.LOC-TEACH> A_INV <rel LOC-TEACH>
A_INV <asp TEACH.LOC-TEACH> C_INV <abso TEACH> I <abso TAUGHT.2>)

(<abso TAUGHT.2> I <abso TEACH> C_INV <asp TEACH.REC-TEACH> A_INV <rel REC-TEACH> A_INV
<asp REC.REC-TEACH> C_INV <abso PERSON> D <abso TEACH_PROFESSIONAL> D <abso TEACHER> I <abso TEACHER.1>)

> (dribble)
```

Appendix E

List of Sample Rules

The rules that have been mentioned in Chapter 4 and 6 in this thesis will be listed in this section.

```
;
;  In the pattern part, the base pattern is indicated by the flag 1
;  and the raw pattern is indicated by the flag 0.
;  Because the base pattern should be able to specify a portion of a tree,
;  it might be necessary to use more that one node chain elements. For example,
;  in the "passive-by" rule, the base pattern consists of three node chain
;  elements, "[(np-1 s-1) 1 ()] [(aux-1 s-1) 1 (passive)] [(vp-1 s-1) 1 ()]".
;  This is to specify the following shape of a portion of a tree:
;
;                              S
;                            / | \
;                           /  |  \
;                         NP  AUX  VP
;
;  The first element of each rule is the rule name which should be unique.

;  This rule creates a PP.
 (build-pp ((l )(h dative-dobj-pp))
   [ ( [ (prep-1) 0 () ]
       [ (np-1) 0 () ] )
     ( ) ]
   [ (create pp-1)
     (attach prep-1 to pp-1)
     (attach np-1 to pp-1)
     (transfer-features np-1 to pp-1)
   ]
 )

; This rule analyzes the main verb of a clause.
 (main-verb2  ()
   [ ( [ (np-1 s-1) 1 ((link subj)) ]
       [ (v-1) 0 (main (not aux)) ] )
     ( [equal (feature-of num np-1) (feature-of num v-1)]
     ) ]
   [
     (create aux-1)
     (attach aux-1 to s-1)
     (create vp-1)
     (attach v-1 to  vp-1)
     (transfer-features v-1 to vp-1)
     (attach vp-1 to s-1)
     (sem connect np-1 v-1 subj)
     (if (have-feature intransitive in v-1)
         (add-feature struc-ok to s-1))
      ;; if *dummy_prop_instance* is set by the that-complement rule,
      ;; then the proposition instance in this var is connected to
      ;; the main verb of the secondary clause in the global var
      ;; called *connections_sec_clause*.
     (connect_main_sec2 v-1)
   ]
 )

; This rule analyzes the NP which starts with a noun.
;    (ex)  "apples"
 (noun-parse0 ()
```

```
   [ ( [ (noun-1) 0 () ]
     )
     ( [no-ns-na-nc-noun-2] )
   ]
   [ (push-to-cash-last-node) ; to put new nc in front of noun-1.
     (create nc-2 on clist) ; this will be dummy nc containing nothing.
     (bring-from-cash)      ; to get noun-1 after new nc.
   ]
 )

; This rule creates an NC when a noun directly follows the NS.
 (noun-parse1 ()
   [ ( [ (ns-1) 0 () ]
       [ (noun-1) 0 () ] )
     () ]
   [ (create-mother nc-1 of ns-1)
     (transfer-features ns-1 to nc-1)
      ]
 )

; This rule identifies the head noun of an NP by noticing that it is
; followed by some non-noun element.
;    (ex) "the book is..."
 (noun-parse3 ((1 which-diag2 parse-det pronoun-rule)(h ))
   [ ( [ (nc-1) 0 () ]
       [ (noun-1) 0 () ]  ; this noun becomes the end of np.
       [ (not noun) 0 ((word)) ] ) ; this gives the enough context.
     ( ) ]
   [ (push-to-cash-last-node)
     (create np-1 on clist)
     (attach nc-1 to np-1)
     (transfer-features nc-1 to np-1)
     (attach noun-1 to np-1)  ; noun-1 is the end of np.
     (sem connect nc-1 noun-1) ; nc-1 alone means the last noun under it.
     (sem connect (na nc-1) noun-1 adj-mod); na alone means the last adj underit.
     (transfer-features noun-1 to np-1)
     (if (have-feature wh in np-1)
         (bind-whcomp np-1) ; then
         (add-feature modifiable to np-1)) ; else
           ; deleted from clist by bind-whcomp. specific rules will do.
                                                          ]
 )

; This rule identifies the head noun of an NP by noticing that
; a noun with plural form should be the head noun.
;    (ex) "the books ..."
 (noun-parse33 ()
   [ ( [ (nc-1) 0 () ]
       [ (noun-1) 0 ((have-feature num 3p) (not sing-plu-same)) ]
         ;; sing-plu-same indicates that the noun can be both singular
         ;; and plural. (ex) fish, sheep.
     )
     () ]
   [ (create np-1 on clist)
     (attach nc-1 to np-1)
     (transfer-features nc-1 to np-1)
     (sem connect nc-1 noun-1)
     (sem connect (na nc-1) noun-1)
     (attach noun-1 to np-1)
     (transfer-features noun-1 to np-1)
     (if (have-feature wh in np-1)
         (bind-whcomp np-1) ; then
         (add-feature modifiable to np-1)) ; else
           ; deleted from clist by bind-whcomp. specific rules will do.
   ]
 )

; This rule analyzes a non-"head noun" in the nominal compound by noticing
; that a noun(of singular form) is immediately followed by another noun.
;    (ex) "the book shelf ..."
 (noun-parse4 ()
       ;;; accomodate noun-1 as part of the np being built.
   [ ( [ (nc-1) 0 () ]
       [ (noun-1) 0 ((have-feature num 3s)) ] ; becomes modifier of np.
       [ (noun-2) 0 () ] ; this will become part of the np being built.
                         )
     ()
   ]
   [ (sem connect nc-1 noun-1) ; it is not clear if this suggestion is
              ; necessary. more study of nominal compound is required.
     (attach noun-1 to nc-1)   ]
```

```
 )

; This rule determines that a noun followed by a plural noun can be
; the head noun of a NP and the plural noun will build another NP.
;     (ex) "the fish eggs ..."
 (noun-parse5 ()
   [ ( [ (nc-1) 0 () ]
       [ (noun-1) 0 ((have-feature num 3s)) ] ; this is the end of np built.
       [ (noun-2) 0 ((have-feature num 3p)) ] ; if it is not plural form,
                      ; an article(a, the) should come to lead relative cl.
                      ; noun-2 becomes a new np.
                    )
     () ]
   [ (push-to-cash-last-node)
     (create np-1 on clist)
     (attach nc-1 to np-1)
     (transfer-features nc-1 to np-1)
     (sem connect nc-1 noun-1)
     (sem connect (na nc-1) noun-1)
     (attach noun-1 to np-1) ; this is the head noun of np-1.
     (transfer-features noun-1 to np-1)
     (if (have-feature wh in np-1)
         (bind-whcomp np-1) ; this is then part.
         (add-feature modifiable to np-1)) ; this is else part.
   ]
 )

; This rule attaches a PP to an NP which immediately preceding it.
;     (ex) "a book on the desk"
 (np-pp  ((l )(h passive-by))
   [ ( [ (np-1 low-right) 1 (modifiable) ]
       [ (pp-1)       0 () ] )
     () ]
   [ (attach pp-1 to np-1)
     (if (have-feature trace in pp-1)
         (sem connect (np pp-1) np-1 (prep pp-1) 2) ; if wh object is under pp.
         (sem connect np-1 (np pp-1) (prep pp-1)) )
     (remove-feature modifiable from np-1) ]
 )

; This rule analyzes the reduced relative clause when an NP is followed by
; a verb with past participle form(the NP is analyzed as the direct object
; of the verb.
;     (ex) "the book torn ..."
 (np-vpp ()
   [ ( [ (np-1 low-right) 1 (modifiable) ]
       [ (v-1) 0 (en transitive) ]   ; has feature of past participle
                    )
     ( )  ]
   [ (create s-1 in clist)
    (add-feature sec to s-1)
    (attach s-1 to np-1 implicitly)
    (create np-2)
    (add-feature dummy to np-2)
    (attach np-2 to s-1 as subj)
    (create aux-1)
    (add-feature passive to aux-1)
    (attach aux-1 to s-1)
    (create vp-1)
    (attach v-1 to vp-1)
    (create np-3) ; this is the trace node for object.
    (add-feature trace to np-3)
    (make-pointer np-3 to np-1)
    (attach np-3 to vp-1 as obj)
    (sem connect np-1 v-1 obj 2) ; see order of w1 and w2 here.
    (attach vp-1 to s-1)
    (add-feature struc-ok to s-1)
    (remove-feature modifiable from np-1)
                                    ]
 )

; This rule applies for the same situation for the above "np-vpp" rule.
; But the NP is analyzed as the indirect object of the verb.
;     (ex) "the teacher taught by the professor ..."
 (np-vpp1 ()
   [ ( [ (np-1 low-right) 1 (modifiable) ]
       [ (v-1) 0 (en transitive dative) ]   ; has feature of past participle
                    )
     ( )  ]
   [ (create s-1 in clist)
    (add-feature sec to s-1)
```

```
    (attach s-1 to np-1 implicitly)
    (create np-2)
    (add-feature dummy to np-2)
    (attach np-2 to s-1 as subj)
    (create aux-1)
    (add-feature passive to aux-1)
    (attach aux-1 to s-1)
    (create vp-1)
    (attach v-1 to vp-1)
    (create np-3)
    (add-feature trace to np-3)
    (make-pointer np-3 to np-1)
    (attach np-3 to vp-1 as iobj)
    (sem connect np-1 v-1 (dative-prep v-1) 2) ; see order of w1 and w2 here.
    (attach vp-1 to s-1)
    (add-feature struc-ok to s-1)
    (remove-feature modifiable from np-1)
                                        ]
)

; This rule analyzes the direct object of a clause.
 (object-rule  ()
   [ ( [ (v-1 vp-1 s-1) 1 (transitive) ]
       [ (np-1)         0 ((not wh)) ] )
     () ]
   [ (attach np-1 to vp-1 as obj)
     (sem connect v-1 np-1 obj)
     (add-feature struc-ok to s-1) ]
)

; This rule analyzes the determiner(a, the) of an NP.
 (parse-det ((l )(h noun-parse3))
   [ ( [ (det-1) 0 () ]
                      )
     ( )  ]
   [ (create ns-1 on clist)
     (attach det-1 to ns-1)
     (transfer-features det-1 to ns-1) ]
)

; This rule analyzes the passive clause's real subject of the form "by NP".
;     (ex) "... eaten by the tiger"
 (passive-by ((l np-pp np-pp1)(h ))
   [ ( [ (np-1 s-1) 1 () ]
       [ (aux-1 s-1) 1 (passive) ]
       [ (vp-1 s-1) 1 () ]
       [ (pp-1)  0 () ] )
     ( [equal-exact (symbol by) (prep-of-pp pp-1)] ) ]
   [ (sem connect (v vp-1) (np pp-1) subj 2)
     (attach pp-1 to vp-1)
     ;;(replace np-1 with pp-1)
     ;;(remove-from-clist pp-1)
   ]
)

; This rule creates the start of a that-complement clause where
; "that" is missing.
;     (ex) "I know he passed the test."
 (reduced-that-comp ((l reduced-relative)(h )) ;;; reduced-relative has lower pri.;;;;
   [ ( [ (vp-1 s-1) 1 (thatcomp) ]
       [ (np-1) 0 ( (not wh) ) ]
                                    )
     ( ) ]
   [ (push-to-cash-last-node) ; move the np to cash.
     (create s-2)
     (add-feature sec to s-2)
     (add-feature comp to s-2)
     (create that-comp-1)
     (add-feature dummy to that-comp-1)
     (attach that-comp-1 to s-2) ; to use as a claudse starter.
     (attach s-2 to vp-1 as comp implicitly)
     (add-feature struc-ok to s-1)
       ; the following action push the instance of (v vp-1) to
       ; the global variable *connect_main_sec_proposition*.
       ; the instance for the verb of sec clause will be pushed
       ; later by the main verb processing rules.
     (connect_main_sec1 (v vp-1) )
       ]
)

; This rule analyzes the first NP of a clause to be the subject of the clause.
```

```
 (subject-rule ()
   [ ( [ (*clause-start* s-1) 1 () ] ; ss, relpro, comp, ........
       [ (np-1) 0 ((not wh)) ] )
     () ] ; no additional restrictions
   [ (attach np-1 to s-1 as subj)
     (add-feature subj to np-1) ]
 )

; This rule close an affirmative sentence by looking at the final
; punctuation mark ".".
 (s-close-aff ()
   [ ( [ (s-1) 1 ( (not question) main struc-ok ) ]
       [ (fpunc-1) 0 ( (not question) ) ])
     ()  ]
   [ (attach fpunc-1 to s-1)
     (set-sentence-finish) ]
 )

; This rule attaches a PP to the VP of a clause.
;     (ex) "eat with a fork"
 (vp-pp  ((l )(h wh-dative-np-pp))
   [ ( [ (vp-1 s-1) 1 () ]
       [ (pp-1) 0 () ] )
     () ]
   [ (attach pp-1 to vp-1)
     (if (have-feature trace in pp-1)
         (sem connect (np pp-1) (v vp-1) (prep pp-1) 2) ; if wh object is under pp.
         (sem connect (v vp-1) (np pp-1) (prep pp-1))  )

   ]
 )

; This rule starts the relative clause by noticing that
; "which" or "that" is directly following an NP.
;     (ex) "the book which the man read ..."
 (wh-relative ()     ; after this rule, subject rule will run.
   [ ( [ (np-1 low-right) 1 (modifiable) ]
       [ (relpro-1) 0 () ] )
     () ]
   [ (create s-1 on clist)
     (add-feature sec to s-1)
     (add-feature rel to s-1)
     (attach relpro-1 to s-1) ; relpro-1 will work as clause-starter
     (bind-whcomp np-1)    ; this means making whcomp reg point to np-1
     (remove-feature modifiable from np-1)
     (attach s-1 to np-1 as mod implicitly) ]  ; but s-1 should still be on clist.
 )
```

LIST OF REFERENCES

LIST OF REFERENCES

Akmajian, A. and Heny, F. W. (1975) *An Introduction to the Principles of Transformational Syntax.* Cambridge, Massachusetts: The MIT Press, 1975.

Allen, J. F. (1983) "Recognizing Intentions from Natural Language Utterances". in Brady, M. and Berwick, R. C. (editors). *Computational Models of Discourse.* Cambridge, MA: The MIT Press, 107-166.

Allen, J. F. and Hayes, P. J. (1985) "A common sense theory of time." *Proceedings of 9th International Joint Conference on Artificial Intelligence,* Los Angeles, 1985, 528-531.

Alterman, R. (1985) "A Dictionary Based on Concept Coherence." *Artificial Intelligence* 25(2), 1985, 153-186.

Anderson, J. (1983) *The Architecture of Cognition.* Harvard University Press, Cambridge, MA, 1983.

Barton, E. and Berwick, R. (1985) "Parsing with assertion Sets and Information Monotonicity." *Proceedings of the 9th International Joint Conference on Artificial Intelligence,* Los Angeles, California, 1985, 769-771.

Berwick, R. (1983) "A Deterministic Parser with Broad Coverage." *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany, 1983.

Birnbaum, L. and Selfridge, M. (1981) "Conceptual Analysis." in Schank and Riesbeck 1981, 318-353.

Bobrow, D. G. (1968) "Natural Language Input for a Computer Problem Solving System." in Minsky, M. L.(editor), *Semantic Information Processing.* Cambridge, Mass.:The MIT Press, 146-226.

Bobrow, D. G. and Winograd, T. (1977) "An Overview of KRL: A knowledge representation language." *Cognitive Science* 1(1), 1977, 3-46.

Bobrow, R. J. and Webber, B. L. (1980) "Knowledge Representation for Syntax/Semantic Processing." *Proceedings of the First Annual National Conference on Artificial Intelligence*. Stanford, August 1980, 316-323.

Brachman, R. J. (1979) "On the Epistemological Status of Semantic Networks." In *Associative Networks: Representation and Use of Knowledge by Computers*. N. V. Findler(editor), New York: Academic Press, 1979.

Brachman, R. J. (1983) "What IS-A is and isn't: An Analysis of Taxonomic Links in Semantic Nets." *Computer* 16(10) 1983, 30-36.

Brachman, R. J., Fikes, R. E. and Levesque, H. J. (1983) "Kripton: A Functional Approach to Knowledge Representation." *Computer* 16(10), 1983, 67-73.

Brachman, R. J. and Schmolze, J. G. (1985) "An overview of KL-ONE knowledge representation system." *Cognitive science*, 9(2), 1985, 171-216.

Bresnan, J. W. (editor) (1982) *The Mental Representations of Grammatical Relations*. Cambridge, Massachusetts: The MIT Press, 1982.

Burton, R. (1976) "Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems." Technical Report 3453, Bolt Beranek and Newman, Cambridge, Mass, 1976.

Carbonell, J. G. and Hayes, P. J. (1983) "Recovery Strategies for parsing Extragrammatical Language." *Computational Linguistics*, 9(3-4), 1983, 123-146.

Charniak, E. (1976) "Inference and Knowledge." in Charniak and Wilks 1976.

Charniak, E. (1978) "On the use of framed knowledge in language comprehension." *Artificial Intelligence," 11(3), 1978, 225-266.*

Charniak, E. (1981a) "A Common Representation for Problem Solving and Language Comprehension Information." *Artificial Intelligence*, 16(3), 1981, 225-255.

Charniak, E. (1981b) "The Case-Slot Identity Theory." *Cognitive Science,* 5(3), 1981, 285-292.

Charniak, E. (1981c) "Six Topics in Search of a Parser." *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouber, Canada, 1981, 1079-1087.

Charniak, E. (1982) "Context Recognition in Language Comprehension." in Lehnert and Ringle 1982, 435-454.

Charniak, E. (1983a) "Passing markers: A Theory of Contextual Influence in Language Comprehension." *Cognitive Science*, 7(3), 1983, 171-190.

Charniak, E. (1983b) "A Parser with Something for Everyone." in King 1983, 117-149.

Charniak, E. (1986) "A Neat Theory of Marker Passing." *Proceedings of the 5th National Conference on Artificial Intelligence, 1986, 584-588.*

Charniak, E. and Mcdermott, D. V. (1986) *Introduction to Artificial Intelligence.* Addison Wesley, 1986.

Charniak, E., Riesbeck, C. K. and McDermott, D. V. (1980) *Artificial intelligence programming*. Hillsdale, N.J., Lawrence Erlbaum Associates, 1980.

Charniak, E. and Wilks, Y. A. (1976) *Computational Semantics: An Introduction to Artificial Intelligence and Natural Language Comprehension.* Amsterdam: North-Holland, 1976.

Chomsky, A. N. (1965) *Aspects of the Theory of Syntax.* Cambridge, MA: The MIT Press, 1965.

Church, K. (1980) "On memory limitations for natural language parsing." Technical Report MIT/LCS/TR-245, Laboratory for Computer Science, MIT, 1980.

Church, K. and Patil, R. S. (1982) "Coping with syntactic ambiguity or how to put the block in the box on the table." *American Journal of Computational Linguistics,* 8(3-4), 1982.

Colby, K. M., Faught B. and Parkinson R. (1974) "Pattern Matching Rules of the recognition of Natural Language Dialogue Expressions." Stanford AI Lab. Memo AIM-234, 1974.

Collins, A. M. and Loftus, E. F. (1975) "A spreading activation theory of semantic processing." *Psychological Review,* 82(6), 1975, 407-428.

Crain, S. and Steedman M. (1985) "On not Being Led up the Garden Path: The Use of Context by the Psychological Parser." in *Studies in Natural Language Processing*. Dowty, D., Karttunen, L. J. and Zwicky, A. M.(editors), Cambridge University Press, 1985.

Cullingford, Richard (1978). "Script Application: Computer Understanding of Newspaper Stories." Research Report No. 116, Department of Computer Science, Yale University, New Haven, Connecticut, 1978.

Dahlgren, K. and Mcdowel, J.(1986) "Using common sense knowledge to disambiguate prepositional phrase modifiers." *Proceedings of the 5th National Conference on Artificial Intelligence,* Philadelphia, 1986, 589-593.

Dowty, D. R., Wall, R. E. and Peters, S. (1981) *Introduction to Montague Semantics.* Dordrecht: D. Reidel, 1981.

Dyer, M. (1983) *In-Depth Understanding*. Cambridge, MA: The MIT Press, 1983.

Eiselt, K.P. (1985) "A Parallel-process Model of On-Line Inference Processing." *Proceedings of the 9th International Joint Conference on Artificial Intelligence,* Los Angeles, California, 1985. 863-869.

Fahlman, S. E. (1979) *NETL: A System for Representing and Using Real-World Knowledge*. MIT Press, Cambridge, 1979.

Feldman, J. A. and Ballard, D. H. (1982) "Connectionest models and their properties." *Cognitive Science*, 6(3), 1982, 205-254.

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

Fillmore, C. J. (1968) "The Case for Case." in *Universals in Linguistic Theory*. E. W. Bach and R. T. Harms(editors), Holt, Reinhart and Winston, New York, 1968, 1-88.

Finin, T. (1980) "Semantic interpretation of nominal compounds." *proceeding of The first annual national conference on artificial intelligence*, 1980, 310-312.

Fodor, J. A. (1983) *Modularity of Mind*, Cambridge, MA: MIT Press, 1983.

Fodor, J., Bever, T., and Garrett, M. (1974) *The Psychology of Language*. McGrawhill Book Co., New York, 1974.

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

Ford, M., Bresnan, J. W. and Kaplan, R. M. (1982) "A Competence-based Theory of Syntactic Closure." in Bresnan 1982, 727-796.

Forster, K. I. (1979) "Levels of processing and the structure of the language processor." In *Sentence processing: psycholinguistic studies presented to Merill Garrett*. W. E. Cooper and E. C. T. Walker (editors), Hillsdale, N.J.: Lawrence Erlbaum, 1979.

Frazier, L. (1978) *On Comprehending Sentences: Syntactic Parsing Strategies*. Ph. D. Dissertation, University of Connecticut, 1978.

Frazier, L. and Fodor, J. D. (1978) "The Sausage Machine: a New two Stage Parsing Model." *Cognition*, 6(4), 1978, 291-325.

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

Frazier, L. (1987) "Theories of sentence processing." In *Modularity in Knowledge Representation and Natural Language Understanding*. Garfield, J. L. (editor), Cambridge, MIT Press, 1987, 291-307.

Gawron, J. M., King, J. J, Lamping J. and Egon E., Paulson, E. A., Pullum, G. K., Sag, I. A. and Wasow, T. A. (1982)
"Processing English with a Generalized Phrase Structure Grammar." *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, Tronto, June 1982. 74-81.

Gazdar, G. (1981) "Unbounded dependencies and constituent structure." *Linguistic Inquiry*, 12, 1981, 155-184.

Gazdar, G. (1982) "Phrase Structure Grammar." in *The Nature of Syntactic Representation*. Jacobson, P. and Pullum, G. K. (editors), Dordrecht: D. Reidel, 1982.

Gorell, P. G. (1987) *Studies of human syntactic processing: Ranked-parallel versus serial models*. Ph. D. Dissertation, University of Connecticut, 1987.

Granger, R. H. (1983) "The NOMAD System: Expectation-based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-formed Text." *Computational Linguistics*, 9(3-4), 1983. 188-196.

Granger, R., Eiselt, K., and Holbrook, J. (1986) "Parsing with parallelism:a spreading-activation model of inference processing during text understanding." in *Memory, Experience and Reasoning*. J. Kolodner and C. Riesbeck(editors), Hillsdale, NJ:Erlbaum, 1986.

Grice, H. P. (1975) "Logic and conversation." in *Syntax and Semantics, Vol. III: Speech Acts*. P. Cole and J. L. Morgan(editor), Academics Press, New York, 1975.

Grosz, B. (1983) "TEAM: A transportable natural language interface system." *Proceedings of the Conference on Applied Natural Language Processing*. Santa Monica, 1983, 39-45.

Hendler, J. (1986) "Integrating Marker-Passing and Problem-Solving: A Spreading Activation Approach to Improved Choice in Planning." TR-1624, Department of Computer Science, University of Maryland, College Park, MD, 1986.

Hirst, G. (1984) *Semantic Interpretation Against Ambiguity*. Ph. D. Dissertation, Department of Computer Science, Brown University, 1984.

Hirst, G. (1986) *Semantic Interpretation and the Resolution of Ambiguity*. Cambridge University Press, Cambridge, England, 1986.

Joshi, A. K., Webber, B. L. and Sag, I. A. (editors) (1981) *Elements of Discourse Understanding*. Cambridge University Press, 1981.

Katz, J. J. and Fodor J. A. (1963) "The Structure of a Semantic Theory." *Language*, 39(2), 1963, 170-210.

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

King, M. (1983) *Parsing Natural Language*. London: Academic Press, 1983.

Kurtzman, H. (1985) *Studies in syntactic ambiguity resolution*. Ph. D. Dissertation, MIT, 1985, distributed by IULC.

Lebowitz, M. (1980) *Generalization and Memory in an Integrated Understanding System.* Ph. D. Dissertation, Department of Computer Science, Yale University. 1980.

Lehnert, W. G. and Ringle, M. H.(editors) (1982) *Strategies for Natural Language Processing.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Lesser, V. R., Fennel, R. D., Erman L. D. and Reddy, D. R.(1975) "Organization of the Hearsay II Speech Understanding System." *IEEE Transaction on Acoustics, Speech, and Signal Processing.* ASSP-23, 1975, 11-24.

Lorch, R. F. (1982) "Priming and Search Processes in Semantic Memory: A test of three models of spreading activation." *Journal of Verbal Learning and Verbal Behavior, 21, 1982, 468-492.*

Lytinen, S. L. (1984) *The Organization of Knowledge in a Multi-lingual Integrated Parser.* Ph. D. Dissertation, Department of Computer Science, Yale University, 1984.

Lytinen, S. L. (1986) "Dynamically Combining Syntax and Semantics in Natural Language Processing." *Proceedings of the 5th National Conference on Artificial Intelligence,* 1986, 574-578.

Lytinen, S. L. (1988) "Are Vague Words Ambiguous?" in *Lexical Ambiguity Resolution,* S. Small, G. Cottrell and M. Tanenhaus(editors), Morgan Kaufmann Publishers, California, 1988, 109-128.

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

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

Marr, D. (1977) Artificial Intelligence - A personal view", *Artificial Intelligence,* 9, 1977, 37-48.

Marslen-Wilson, W. D and Tyler, L. (1980) "The Temporal Structure of Spoken Language Understanding." *Cognition,* 8(1), 1980, 1-71.

McCarthy, J. and Hayes, P. J. (1969) "Some philosophical problems from the stand point of artificial intelligence." in *Machine Intelligence 4,* Meltzer and Michie(editors),

Edinburgh, 1969, 463-502.

McDermott, D. (1981) "Artificial Intelligence Meets Natural Stupidity." in *Mind Design*, J. Haugeland(editor), MIT Press, Cambridge, MA, 1981, 143-161.

Milne, R. (1980) "a Framework for Deterministic Parsing using Syntax and Semantics." DAI Working Paper, Department of Artificial Intelligence, University of Edinburgh, January 1980.

Milne, R. (1982) "Predicting Garden Path Sentences." *Cognitive Science*, 6, 1982, 349-373.

Milne, R. (1983) *Resolving Lexical Ambiguities in the Deterministic Parser*. Ph. D. Dissertation, Department of Artificial Intelligence, University of Edinburgh, Scotland, 1983.

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

Minsky, M. (1968) *Semantic Information Processing*. MIT Press, Cambridge, MA, 1968.

Minsky, M. (1975). "A Framework for Representing Knowledge." in *The Psychology of Computer Vision*. Winston, Patrick (editor), New York, McGraw-Hill, 1975, 211-277.

Norvig, P. (1983a) "Frame Activated Inferences in a Story Understanding Program." *Proceedings of the 8th International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany, 1983.

Norvig, P. (1983b) "Six Problems for Story Understanders." *Proceedings of 4th National Conference on Artificial Intelligence*, Washington, DC, 1983.

Norvig, P. (1986). *Unified theory of inference for text understanding*. Ph. D. thesis, Department of Computer Science, University of California: Berkeley, 1986.

Pazzani, M. K. and Engelman, C. (1983) "Knowledge-based Qusetion Answering." *Proceedings of the Conference on Applied Natural Language Processing,* Santa Monica, 1983, 73-80.

Pazzani M. K. (1984) "Conceptual Analysis of Garden Path Sentences." *Proceeding of International Conference on Computational Linguistics,* 1984, 486-490.

Perreira, F. C. N. and Warren, D. H. D. (1980) "Definite clause grammars for language analysis--- a survey of the formalism and a comparison with augmented transition networks." *Artificial Intelligence*, 13(3), 1980, 231-278.

Quillian, M. R. (1968) "Semantic Memory." in *Semantic Information Processing*. Minsky, M. L.(editor), Cambridge, MA: The MIT Press, 1968.

Quillian, M. R. (1969) "The teachable language comprehender: A simulation program and theory of language." *Communications of the ACM*, 1969, 459-476.

Ra, D. and Stockman, G. C. (1989a) "Use of Knowledge for Word Sense Disambiguation." *Proceedings of the Annual Conference for the International Association of Knowledge Engineers,* College Park, Maryland, 1989.

Ra, D. and Stockman, G. C. (1989b) "Integrated Natural Language Parsing Based on Interleaved Semantic Processing." *Proceedings of the Second International Symposium on Artificial Intelligence*, Monterrey, Mexico, 1989.

Riesbeck, C. K. (1975) "Conceptual Analysis." in Schank 1975, 83-156.

Riesbeck, C. K. and Martin, C. E. "Direct Memory Access Parsing." Technical Report 354, Department of Computer Science, Yale University, 1984.

Riesbeck, C. K. and Schank R. C. (1976) "Comprehension by Computer: Expectation-based analysis of Sentences in context." Research Report 78, Department of Computer Science, Yale University, 1976.

Rieger, C. (1976) "An organization of knowledge for problem solving and language comprehension." *Artificial Intelligence*, 7(2), 1976.

Ritchie, G. (1983) "Semantics in Parsing." in *Parsing Natural Language*, M. King (editor), Academic Press, London, 1983.

Rumelhart, D. (1975) "Notes on a schema for stories." in *Representation and Understanding*. D. G. Bobrow and A. Collins(editor), Academic Press, New York, 1975, 211-236.

Sabah, G. and Rady, M. (1983) "A deterministic syntactic-semantic parser." *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, August 1983.

Sachs, J. S. (1967) "Recognition Memory for Syntactic and Semantic Aspects of Connected Discourse." *Perception and Psychophysics*, 2, 1967, 437-442.

Sampson, G. R. (1983) "Deterministic Parsing." in King 1983, 90-116.

Schank, R. C. (1975) *Conceptual Information Processing*. Amsterdam: North-Holland, 1975.

Schank, R. C. (1982a) "Reminding and Memory Organization: An Introduction to MOPs." in *Strategies for Natural Language Processing*. W. G. Lehnert and M. H. Ringle(editors), Lawrence Erlbaum Associates, Hillsdale, N.J., 1982, 455-494.

Schank, R. C. (1982b) *Dynamic Memory: a Theory of Reminding and Learning in People and Computers*. Cambridge University Press, 1982.

Schank, R. C. and Abelson, R. P. (1977) *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1977.

Schank, R. C. and Birnbaum, L. (1980) "Memory, Meaning and Syntax." Research report No. 189, Department of Computer Science, Yale University, New haven, Connecticut. 1980.

Schank, R. C. and Riesbeck C.K. (1981) *Inside Computer Understanding: Five Programs plus Miniatures*. Lawrence Erlbaum Associates, Hillsdale, N.J., 1981.

Seidenberg M., Tanenhaus, M., Leiman, J., and Bienkowski, M. (1982) "Automatic access of the meanings of ambiguous words in context: Some limitation of knowledge-based processing." *Cognitive Psychology*, 14(4), 1982, 489-537.

Selfridge, M. (1986) "Interated Processing Produces Robust Understanding." *Computational Linguistics*, 12(2), 1986, 89-106.

Small, S. L. (1980) *Word Expert Parsing: a theory of distributed word-based natural Language Understanding*. Ph. D. Dissertation, Department of Computer Science, University of Maryland, September 1980.

Small, S. L. (1983) "Parsing as Cooperative Distributed Inference: Understanding through Memory Interactions." in King 1983.

Small, S. L., Cottrell, G. W. and Shastri, L. (1982) "Toward Connectionist parsing." *Proceedings of the National Conference on Artificial Intelligence,* 1982, 247-250.

Small, S. L. and Rieger, C. J. (1982) "Parsing and Comprehending with Word Experts(a theory and its realization)." in Lehnert and Ringle 1982, 89-147.

Sowa, J. F. (1984) *Conceptual Structures*. Addison-Wesley, 1984.

Stabler, E.(1983) "Deterministic and bottom-up parsing in prolog." *Proceedings of the National Conference on Artificial Intelligence,* 1983.

Swinney, D. A. (1979) "Lexical Access during Sentence Comprehension: (Re)Consideration of context effects." *Journal of Verbal Learning and Verbal Behavior*, 18(6), 1979.

Tomita, M. (1987) "An Efficient Augmented-context-free Parsing Algorithm." *Computational Linguistics*, 13(1-2), 1987, 31-46.

Tyler, L. and Marslen-Wilson, W. (1977) "The on-line Effects of Semantic Context on Syntactic Processing." *Journal of Verbal Learning and Verbal Behavior*, 16, 1977, 683-692.

Waltz, David L (1982). "The State of the Art of the Natural-language understanding." in Lehnert and Ringle 1982.

Waltz, D. L. and Pollack, J. B. (1984) "Massively parallel parsing: A strongly interactive model of natural language interpretation." Technical Report, Coordinated Science Laboratory, University of Illinois, Urbana, Ill, 1984.

Wanner, E. (1980) "The ATN and the sausage Machine: Which one is baloney?" *Cognition*, 8(2), 1980, 209-225.

Weischedel, R. M. and Ramshaw, L. A. (1986) "Reflections on the Knowledge Needed to Process Ill-Formed Language." Report No. 6264, BBN Laboratories Inc. Cambridge, Massachusetts, 1986.

Weischedel, R. M. and Sondheimer N. K. (1983) "Meta-Rules as a Basis for Processing Ill-formed Input." *Computational Linguistics*, 9(3-4), 1983, 161-177.

Wilensky, R. (1987) "Some Problems and Proposals for Knowledge Representation". Report No. UCB/CSD 87/351, Computer Science Division, University of California, Berkeley, 1987.

Wilensky, R., Arens, Y. and Chin, D. N. (1984) "Talking to Unix in English: An Overview of UC". *Communications of the ACM*, 27(6), 1984.

Wilensky, R., Mayfield, J., Albert, A., Chin, D., Cox, C., Luria, M., Martin, J. and Wu, D. (1986) "UC - A Progress Report". Report no. UCB/CSD 87/303, Computer Science Division(EECS), University of California, Berkeley, 1986.

Wilks, Y. A. (1975a) "An Intelligent Analyzer and Understander of English." *Communications of ACM*, 18(5), 1975, 264-274.

Wilks, Y. A. (1975b) "A Preferential, Pattern-Seeking Semantics for Natural Language Inference." *Artificial Intelligence*, 6, 1975, 53-74.

Winograd, T. (1972) *Understanding Natural Language*. New York, Academic Press, 1972.

Winograd, T. (1983) *Language as a Cognitive Process*. Addison-Wesley, 1983.

Winston, P. H. (1984) *Artificial Intelligence*. Second Edition, Addison-Wesley, 1984.

Woods, W. A. (1970) "Transition Network Grammars for Natural Language Analysis." *Communications of the ACM*, 13(10), 1970, 591-606.

Woods, W. A. (1972) "An experimental parsing system for transition network grammars." in *Natural Language Processing*, R. Rustin (editor), Algorithmics Press, New York, 1972.

Woods, W. A. (1975) "What's in a Link: Foundations for Semantic Networks." in *Representation and Understanding: Studies in Cognitive Science*. Bobrow, D. G. and Collins, A. M. (editors), New York: Academic Press, 1975, 35-82.

Woods, W. A. (1983) "What's Important about Knowledge Representation." *Computer*, 16(10), 1983, 22-29.

Woods, W. A., Kaplan, R. M. and Nash-Webber, B. L. (1972) "The Lunar Sciences Natural Language Information system: Final Report." Report 2378, Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts, 1972.