American Journal of Computational Linguistics *Microfiche 45*

SYNTAX

IN

AUTOMATIC SPEECH UNDERSTANDING

MADELEINE BATES

Boston University

and

Bolt Beranek and Newman Inc.

50 Moulton Street

Cambridge, Massachusetts 02138

This research was principally supported by the Advanced Research Projects Agency of the Department of Defense (ARPA Order No. 2904) and was monitored by ONR under Contract No. N00014-75-c-0533. Partial support of the author by NSF grant GS-39834 to Harvard University is gratefully acknowledged.

Table of Contents

Section 1

Introduction

Understanding speech is an extremely complex process which requires the use of many types of knowledge, one of which is syntax. This report presents a system called SPARSER which is designed to provide and use the syntactic knowledge necessary to support an artificial speech understanding system. (We will assume for the remainder of this paper that unless explicitly stated otherwise "speech" means grammatical speech spoken at a moderate rate with natural inflections and pauses, spontaneously produced but similar to the type of speech produced by reading text.)

We will make the following assumptions about the characteristics of speech and a speech processor:

1. There is not enough information in the spech signal to uniquely identify the phonemes or words in a normally spoken utterance.
2. The acoustic processing component of any artificial speech understanding system will introduce additional errors and ambiguity as it attempts to identify the phonemes or words in the utterance.
3. As a consequence of 1 and 2, when an utterance is scanned to try to identify the words, it is reasonable to suppose that a number of (perhaps overlapping) candidates will be found.

This is illustrate in Figure 1.1 by a structure called a word lattice which shows schematically that many words may initially appear to be present. In this representation, the numbers along the horizontal scale are segment boundary points in the utterance which roughly correspond to points in time. This word lattice was produced by the lexical retrieval component of the BBN speech understanding system from an utterance which had been segmented and labeled by hand under conditions designed to simulate the performance of an automatic segmenter and labeler.

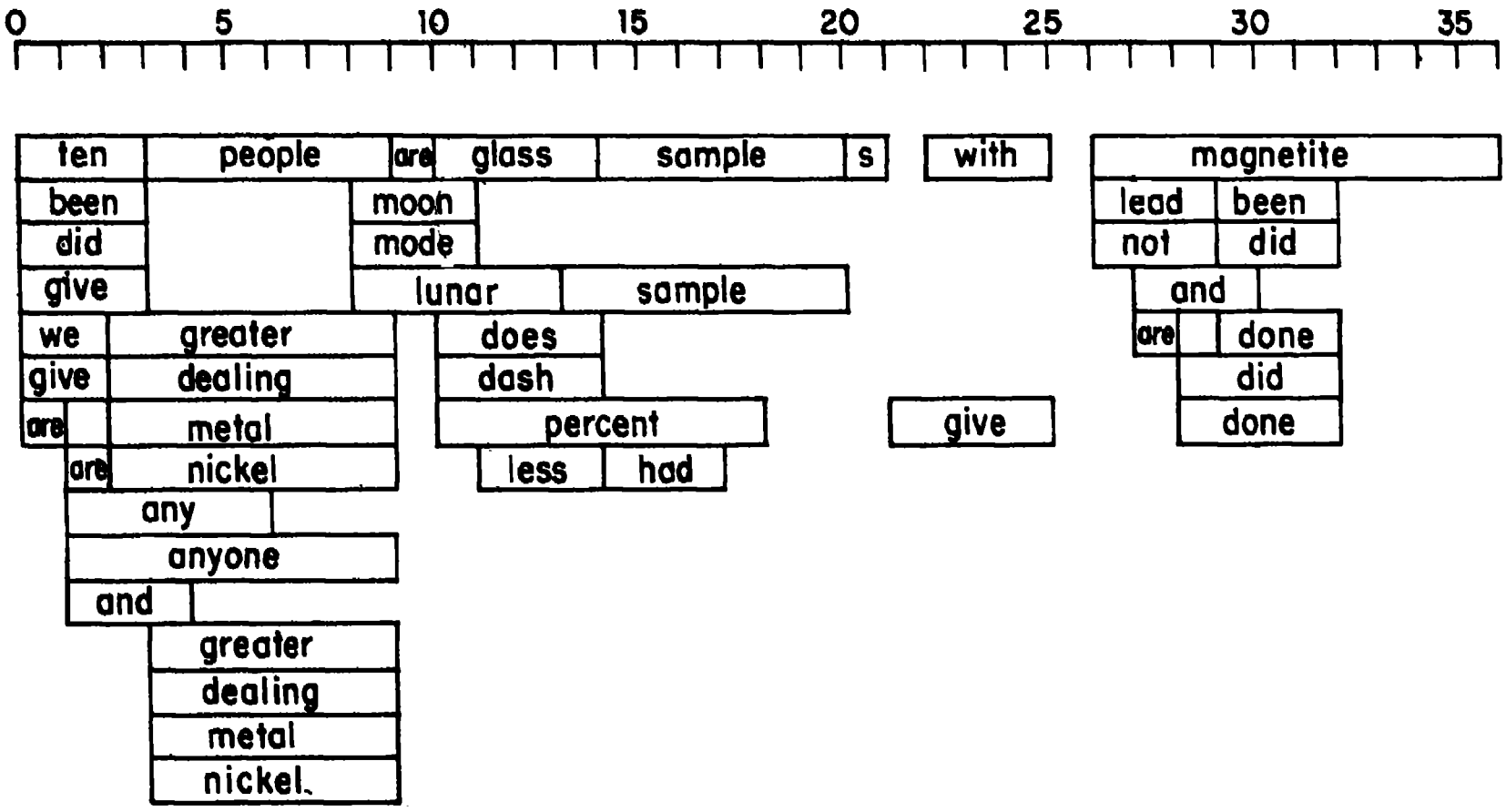

Figure 1.1

A Word Lattice
Sentence: Give me all glass samples with magnetite.

In the system described here, such a word lattice can be represented by a collection of word matches, each of which is composed of a word, the boundary points at the left and right ends of the portion of the utterance where it appears to match well, and a score indicating how well it matches the ideal phonemic representation of the word.

We also make a number of assumptions about the nature of the speech understanding process and the characteristics of a system to carry out that process:

1. People can understand a speaker even when the speech is fairly ungrammatical, so a syntax-driven system which would accept only input meeting rigid syntactic requirements would not be adequate for natural, converstional speech.
2. Since a number of word candidates are likely to be found throughout the utterance, it may be fruitful to be able to select a subset of them on semantic, pragmatic, or prosodic grounds as well as syntactic, depending on which cues seem most robust.
3. Syntax must interact with semantics in order to cut down the combinatorial explosion of syntactically correct but meaningless subsets of the utterance. Even in the small word lattice of Figure 1.1 it can be seen that there are numerous short sequences which are syntactically but not semantically valid (e.g. "Ten people are glass samples with magnetite", "glass samples give magnetite", "lunar samples give magnetite", "samples give lead", "people are percent", etc.).
4. The input to a speech parser will be similar to the word lattice described above, thus the parser will have to face not only the problem that one or more words in its input might be incorrect, but that gaps may appear in the input as well.
5. The parser will have to have the ability to predict words and syntactic classes which are consistent with partial hypotheses about the content of the sentence in order to help fill gaps in the lattice.
6. Because of the combinatorial explosion of syntactic

alternatives which occurs when all syntactic possibilities are explored for small sections of an utterance, the syntactic component must limit the number of such alternatives which are actually generated, or at least factor them or treat them implicitly rather than explicitly. One way of partially solving this problem is to order the alternatives in such a way that only the best alternatives are extended.

Section 2

The BBN Speech Understanding System

In the past few years there has been a flurry of activity in the field of automatic speech understanding, resulting in a number of different systems. For surveys of a number of these systems the reader is recommended to Wolf [31], Bates[4], and Hyde [10]. For more specific details on some of the individual systems, see [1, 2, 7, 8, 16, 19, 20, 21, 22, 28, 29, 33, 35]. Since SPARSER was implemented as part of a speech understanding system called SPEECHLIS which is under development at Bolt Beranek and Newman Inc., that system is briefly described here and is further documented in [3, 4, 5, 6, 15, 23, 24, 26, 33, 35]. SPEECHLIS has used two task domains; that of the LUNAR text question-answering system [36] which deals with chemical analyses of Apollo 11 moon rocks and one dealing with travel budget management.

The overall design of the system is illustrated in Figure 2.4. The acoustics component analyzes the acoustic signal to extract features and segment the utterance into a lattice of alternative possible sequences of phonemes (Schwartz and Makhoul [26]), phonological rules augment the output of the acoustic component to include sequences of phonemes which could have resulted in the observed phonemes; the lexical retrieval component retrieves words from the lexicon on the basis of this information (Rovner, et.al. [24]); the word matcher determines the degree to which the ideal phonetic spelling of a given word matches the acoustic analysis at a particular location [24]. All of these components structure their output in such a way as to represent the ambiguity which is inherent in their analyses. For example, they can be used to produce word lattices such as that which was shown in Figure 1.1.

The syntactic component is SPARSER, the system comprising the body of this paper (see also Bates [3, 4]). Acceptable utterances are not restricted to context-free syntax, since the grammar which SPARSER uses is a modified ATN grammar, capable of handling a large, natural subset of English. The remaining sections of this thesis detail the structure and operation of SPARSER.

The semantic component uses a semantic network to associate semantically related words and to judge the meaningfulness of a hypothesized interpretation (See Nash-Webber [15]). This semantic formalism is very-general although a new network must be constructed for each new task domain.

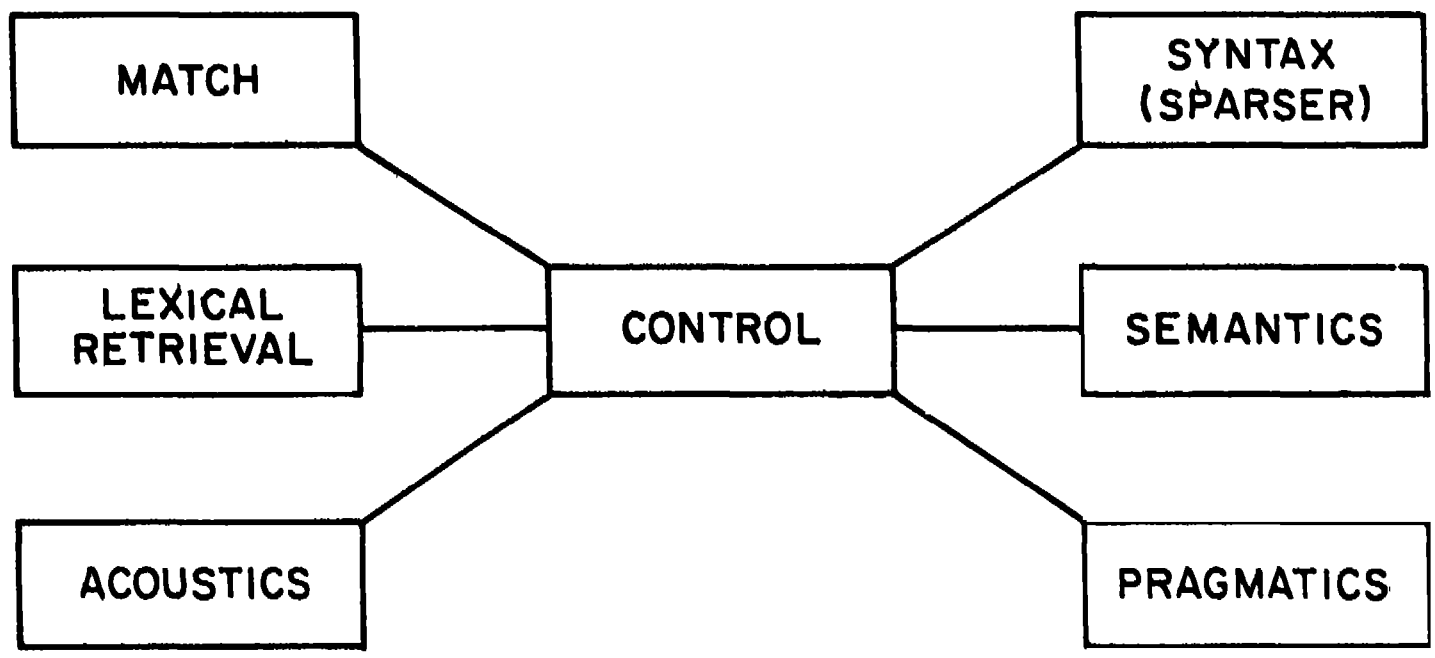

Figure 2.4

Design of BBN SPEECHLIS

The pragmatics component is currently being implemented, but is projected to contain information about the past dialogue, a model of the user, and other pragmatic data (see Bruce [6]).

control component contains an overall strategy for employing the other components in order to obtain an interpretation of an utterance (see Rovner, et al [23]). It decides which component is to be called, what input it is to be given, and what is to be done with the output. It sets thresholds on word match quality. It combines the scores produced by the other components in order to rank competing hypothesies, and is the primary interface to all other components.

Section 3

The Grammar

We have chosen the Augmented Transition Network formalism [32] for the grammar which drives SPARSER because it is a representation which allows merging of common portions of the analysis, it is amenable to both bottom up and top down parsing techniques, it fairly clearly separates the use of local information from information which was obtained from a distant portion of the utterance and, the author s previous experience with a large ATN grammar for parsing text laid the groundwork for the development of a similar grammar for speech.

We have tried as much as possible to keep the formalism which was developed by Woods intact, but some changes have been necessary or desirable to make the grammar more amenable to the speech parser. We call the formalism a Modified Augmented Transition Network (MATN), and assert that it has the same power as the original ATN formalism. The changes are briefly indicated here. For a fuller discussion, see Bates [4].

Every arc of an ordinary ATN has a test component, which may be any predicate. It is usually a boolean combination of tests on the current input word (its features, etc.) and the contents of registers which have been set by actions on previous arcs. In the MATN formalism, the test component of each arc is, on all but the PUSH arc, a list of two tests. The first is a test on the

current word and its features, i.e. a local, context-free test. The second is a test on the register contents, i.e. a context-sensitive test. Both tests must succeed for the arc to be taken.

The reason for splitting up the tests in this way is that register checking tests cannot be made unless the registers are set, and in many situations in the speech environment there may not be enough left context to guarantee that the proper registers would be set. Thus it is useful to be able to evaluate the context-free test on an arc at a different time in the parsing process from the context-sensitive one.

On PUSH arcs, there are three types of tests which are used. It is useful and efficient to test the next word of input before actually doing the PUSH, to see, for example, if the next word can begin a constituent of the type being PUSHed for. This test is called a look-ahead test, and takes the place of the normal context-free test in the test component of the arc. There is also the usual context-sensitive test on registers which were set before the PUSH arc was encountered. And finally, when the PUSH arc returns with a constituent, another context-free test may be done on the structure of the entire constituent. Therefore, the test component of a PUSH arc is a list of the three tests just described.

SENDR s were an efficient mechanism for text parsing because they allowed tests to be made on a lower level which involved information obtained somewhere (possibly far) to the left in the input string -- information which would normally be inaccessible

because it would be hidden on the stack during the parsing of sub-constituents.

There are several reasons for not allowing this mechanism in the speech parser Suppose, in the input that looks like "... the person who travels ...", the word "person" is not the word which was really uttered. If it were allowed to be passed down it would become an integral part of the analysis at the lower level, and if another word were to be hypothesized in its place, the lower level the analysis would have to be redone even if none of the words in the relative clause had been changed. This is a process which would be extremely wasteful, especially in the speech environment where one wants to be able to take as much advantage as possible of information which was gained at one point and slightly altered at another. In particular, it is advantageous to consider as constituents such constructions as relative clauses so that they can be placed in a well-formed-substring table for use by other processes.

Another reason is that some types of verifications (semantic, prosodic, and pragmatic, at least) can be done most conveniently on portions of an utterance which have been assigned a syntactic structure, i.e. on constituents. If a portion of an utterance is parsed (e.g. "that I gave you" from the complete utterance "The book that I gave you") but does not form a complete constituent because it is missing a piece of information from a higher constituent to the left which would have been sent down had it been available, then these verifications may not be made until the missing word or words are identified. Yet it may

be important to build and verify the constituent in order to predict the missing word to the left. Therefore, it is better to allow constituents to be built without information which would normally have been passed down When parsing possibly incorrect fragments with little or no left context, it is best to keep constituents as small and as independent as possible.

The conversion process from an ATN grammar to a MATN grammar with regard to SENDR s is straightforwardand infolves the use of a dummy symbol which is used in the construction of the lower level constituent. When the structure is popped, the PUSH arc examines it for agreement and may replace the dummy node by the appropriate item which would have been sent down. The structure returned by the PUSH for a relative clause on the fragment "that I gave you" might look like Figure 3.1 (where the structure is shown in both the usual tree diagram form and a corresponding form more amenable to computer output).

```
S REL
  S NP PRO I
       FEATS NU SG
    AUX TNS PAST
    VP V GIVE
       NP **NP**
       PP PREP TO
        NP PRO YOU
             FEATS NU SG
```

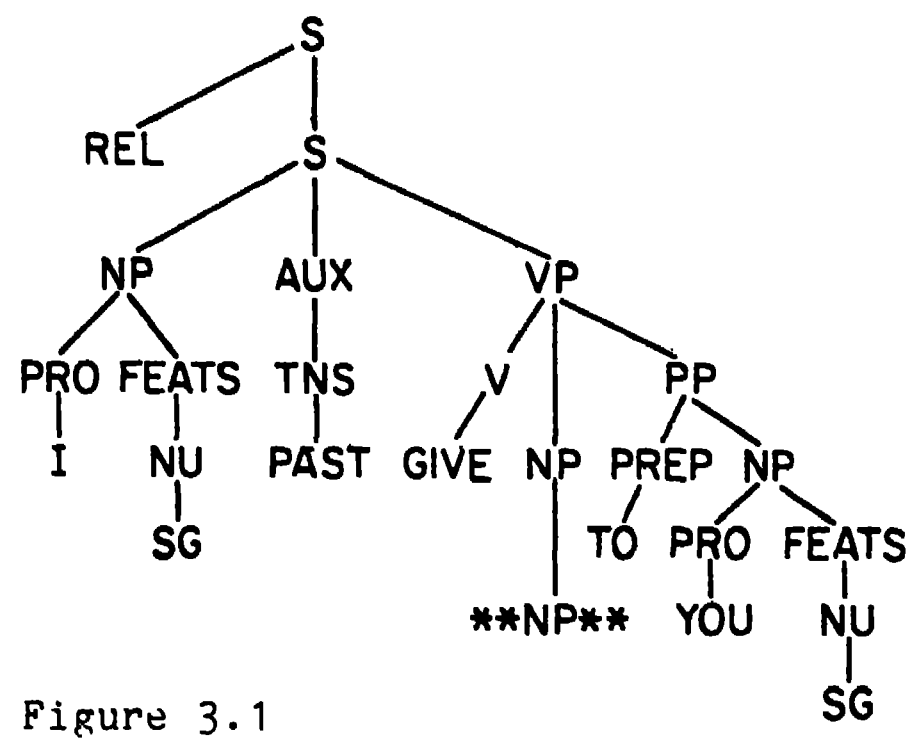

Figure 3.1

Two Representations of a Parse Tree

The fourth element of every arc in a MATN is a small integer which is called the weight of the arc. This weight was originally conceived of as a rough measure of either (a) how likely the arc is to be taken when the parser is in that state or (b) how much information is likely to be gained from taking this arc, i.e. whether the parse path will block quickly if the arc is wrong. That these two schemes are not equivalent can be seen by the following example. In a given state, say just after the main verb of the sentence has been found, the arc which accepts a particle may be much less likely than the arc which jumps to another state to look for complements. However if a particle which agrees with the verb is found in the input stream at this point, then the particle arc is more likely to be correct. Since it is not at all clear how to measure or even intuit how much information is likely to be gained from taking an arc, it was decided that the weights would reflect relative likelihoods. The actual weights which have been used in the speech grammar reflect an intuitive, though experienced guess as to how likely the arc is to be correct if it is taken, assuming the state itself is on the correct path.

Two grammars which will figure predominantly in the remainder of this paper have been written in the MATN formalism. One is an extensive grammar which can handle many questions, declaratives, noun phrase utterances, imperatives, active and passive forms, relative clauses (reduced and unreduced), complements, simple quantifiers, noun-noun modifiers, verb-particle constructions, numbers, and dates (but not conjunctions). It began as a modification of the grammar for the

LUNAR system [36] but has been considerably adapted and expanded. This grammar is called SPEECHGRAMMAR, and is listed in [4]. Examples are given below which were produced using this grammar.

For some illustrative purposes, SPEECHGRAMMAR is too big and complex, so we have produced a MINIGRAMMAR which will be used to show the basic operation of the speech parser. A detailed listing is given in Appendix I, but the diagram in Figure 3.2 probably shows the structure more clearly. The serious reader is encouraged to sketch a copy of this grammar for reference later on.

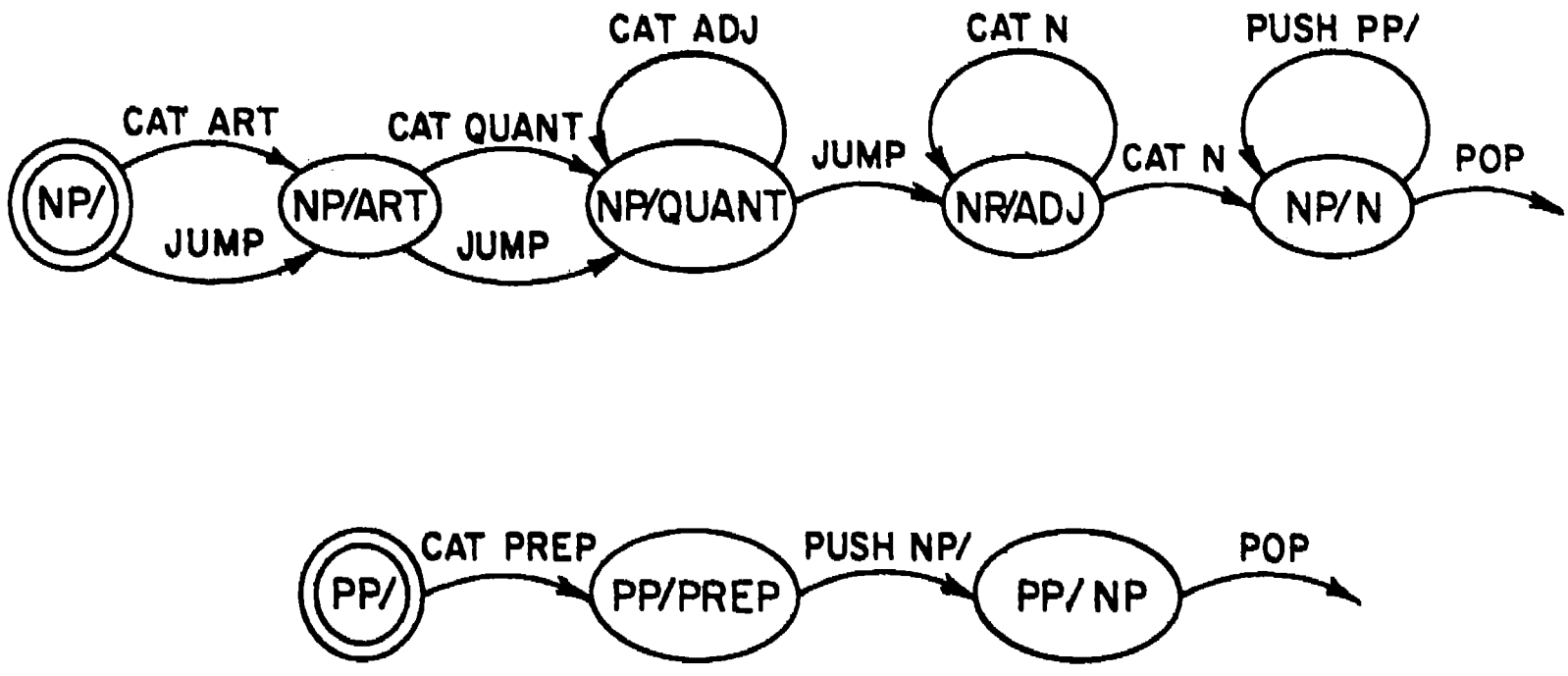

Figure 3.2

MINIGRAMMAR

Since the work reported here was finished, the author has written another grammar, called SMALLGRAM which uses the RATN formalism but which embodies a great deal of semantic and pragmatic information specific to the domain of discourse currently being used by the BBN speech understanding project.

In or for the parser der to move from right to left (to predict what could precede that first given word), it must be able to determine for any state which arcs can enter it, and for any arc which state it comes from. Since the grammar is organized for normal parsing in just the opposite fashion, i.e. for any state one can determine what arcs leave it and for any arc (except POP) one can determine which state it terminates on, it was necessary to build an index into the grammar. This index consists of a number of tables containing pre-computed informationwhich in effect inverts the grammar.

Section 4

Overview of SPARSER

The input to SPARSER is assumed to be a set of words together with their boundary points (which may or may not be related to points in time). A word together with its boundaries is termed a word match. A word match also includes a score which indicates how well the ideal phonemic representation of the word matched the acoustic analysis of the utterance (but as we shall see the parser has little need of this information). Since the same word may match at several sets of boundary points or may match in several ways between the same boundary points, each word match is also given a unique number to help identify it. Thus

the structure for a basic word match is:

(number word leftboundary rightboundary lexicalscore)

e.g. (4 TRAVEL 5 11 94), or (4 TRAVEL 5 11 (94 110)) where the score is given as a pair of numbers representing the actual and maximum scores, or (4 TRAVEL 5 11) where the score is omitted.

How is the input to the parser to be constructed? We assume that acoustic processing and lexical scanning components can operate on a digitized waveform to produce a number of word matches such as previously shown in the word lattice of Figure 1.1. (That this is possible has been demonstrated by Woods [33]). Allowing the parser to operate unrestricted on the entire word lattice would probably not be fruitful because of the large number of locally syntactically correct combinations of words, but one possibility for input to the parser would be to take a set of the best-matching, non-overlapping word matches in the lattice, such as those in Figure 4.1.

A set of non-overlapping word matches is a hypothesis about the content of the utterance. In order to avoid creating large numbers of such sets which are put together combinatorially with no basis except local acoustic match, semantic or pragmatic processes can be used to group word matches based on what is meaningful or likely to be heard. For example, if a dialogue has been about various nickel compounds, the combination "nickel analyses" may be more likely than "chemical analyses" even though the word match for "chemical" has a higher score than that for "nickel"". We will not attempt to detail here how this semantic grouping could be done and how the sets could be scored, since it

has been described elsewhere [15].

```
DO MANY PEOPLE DONE CHEMICAL ANALYSES      ROCK
0  2      6        11     14          22          30    35    38

GIVE EIGHTY PEOPLE DONE   TEN   MODAL  DETERMINATION ROCK
0    3        6        11     14 15   18 21     26                35    38

WERE ANY PEOPLE          METAL            SEVEN
0     3    6        11        17      21          27      32
```

Figure 4.1

Sample Word Match Sets

Using more terminology from the BBN speech system, the word theory to denotes a set of word matches such as we have just described together with (possibly empty) slots for information from each of the possible knowledge sources in the system. From the point of view of SPARSER, usually only the word match portion of a theory is of interest, hence we shall fall into the habit of using the word "theory" to refer to the word match set it contains. When speaking of the syntactic component of a theory, however, we are refering to the information slot for syntax which accompanies each word match set.

Theories have the following characteristics:

1) They contain a set of basic, non-overlapping word matches.

2) They tend at first to contain long content words and not many short function words. This is because long words are more reliably acoustically verified and content words are easier to

relate semantically and pragmatically. Since small words such as "am", "do", "the", "one", "have", "of", "in", etc. may be represented by very little acoustic information, they would tend to match at many places in the utterance where they do not really occur. Consequently they are not searched for by the initial word match scan, nor are they proposed in the semantic stages of hypothesis formation.

3) They need not (and generally do not) completely span the utterance, but have numerous gaps of varying sizes (e.g. for the function words).

4) They tend to contain some sequences of contiguous word matches. Such a sequence is called an island.

That such a set of theories can be created has been demonstrated by the BBN SPEECHLIS system. The syntactic component, SPARSER, is expected to process these theories one at a time. In certain circumstances which will be detailed later, the input to SPARSER will be a theory together with one or more word matches which are to be added in order to create a new larger theory which is then to be syntactically analyzed.

We will assume that there exists a control component which presents SPARSER with theories to process and to which SPARSER can communicate predictions and results.

Preliminaries

Given a theory, what is to be done with it? We begin by considering a subset of the question: Given an island of word matches, what is to be done with it? The answer is to create one

or more parse paths through the island and to predict what words or syntactic classes could surround the island. A parse path is the sequence of arcs in the grammar which would be used by a conventional ATN parser to process the words in the island, if the island were embedded in a complete sentence.

For example, consider the way a parser might process an island of word matches such as (1 CHEMICAL 14 22) (2 ANALYSES 22 30) using the MINIGRAMMAR of the previous section. Beginning in state NP/ of the grammar (omitting for the moment the problem of how it is known that NP/ is the right place to begin) the sequence of arcs which would be taken to parse "chemical analyses" as a noun phrase is that shown below in Figure 4.2.

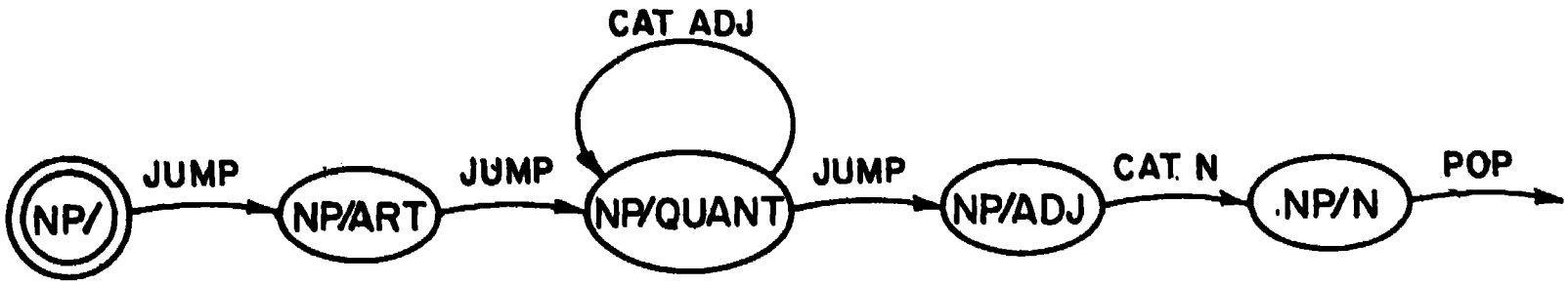

Figure 4.2

Portion of MINIGRAMMAR needed to parse "chemical analyses"

Let us define a <u>configuration</u> to be a representation of the parser being in a given state (say NP/QUANT) at a given point in the utterance (say 14). We will write configurations as STATE:POSITION in text (e.g. NP/QUANT:14) and schematically as a box within which are written the state and the position. If a configuration represents a state which is either the initial state of the grammar or a state which can be PUSHed to (i.e a state which can begin the parsing of a constituent), it is called

an initial configuration, and is indicated schematically by a filled-in semi-circle attached to the left edge of the box. Note that a configuration NP/QUANT:14 is quite distinct from a configuration NP/QUANT:22 since they are at different positions in the input. In SPARSER, each configuration is also assigned a unique number which is a convenient internal pointer.

The process of traversing an arc of the grammar using a particular word is represented by a transition from one configuration to another. A transition can be made only if the arc type is compatible with the current item of input and if the context-free test on the arc is satisfied. (The context-sensitive tests are evaluated later.) A transition carries with it information about the arc which it represents and the item of input it uses. The item of input is usually the word match which the arc uses, but it is NIL in cases such as JUMP arcs which do not use input, and it is a complete constituent for PUSH arcs. A unique identifying number and the list of features, if any, which is associated with the input word or constituent are also recorded on the transition in SPARSER, but they are not shown schematically. A transition is represented schematically by an arrow from one configuration to another with an abbreviated form of the arc written above the arrow and the item of input under it.

The syntactic part of any theory which SPARSER processes contains, among other things, lists of the transitions and configurations which are created or used by the theory. Thus when we talk about creating a configuration or transition it is

implicitly understood that SPARSER also adds it to the appropriate list, and when we talk of adding an existing configuration or transition to a theory we mean adding it to the appropriate list. Therefore, removing a configuration or transition from a theory means removing it from the syntactic part of the theory, not removing it entirely from SPARSER s data base.

Like configurations, transitions are unique, so only one transition is ever constructed from point A to point B for arc X and input Y. We will frequently speak of creating a transition or a configuration, but the reader must bear in mind that if such a configuration or transition already exists, this fact will be recognized and the pre-existing configuration or transition will be used. (Timing measurements indicate that it takes about .052 seconds to create a configuration and only .01 seconds to test if a particular configuration already exists. For transitions, creation takes about .54 seconds and recognition .012 seconds.

The sequence of configurations and transitions which would parse the above example is displayed in Figure 4.3.

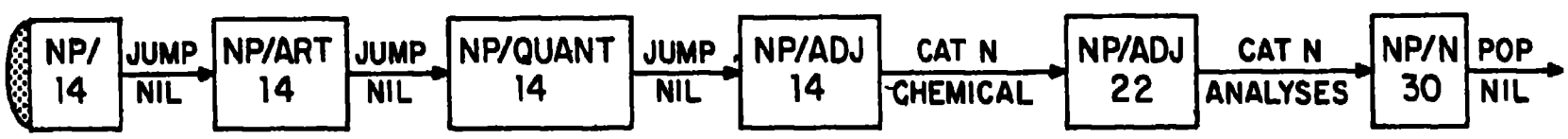

Figure 4.3

Path for parsing "chemical analyses"

A connected sequence of transitions and configurations is called a path. If the sequence begins with an initial configuration and ends with a transition representing a POP arc,

it is a complete path, otherwise it is a partial path. Paths are assumed to be partial unless otherwise specified.

Beginning to Parse an Island

SPARSER processes an island of words by beginning with the leftmost word and determining its possible parts of speech. Then the arcs of the grammar which can process the word are found (by looking in the previously constructed grammar index). For each arc, two configurations are constructed one for the state at the tail of the arc and one for the state at the head, using the left and right boundary positions of the word match, respectively, and a transition for that arc using the current word match is also built. Schematically, we have for our example a situation which looks like that of Figure 4.4 (such a display of all or some of the transitions and configurations which the parser has constructed is called a map). Notice that a configuration may have any number of transitions entering or leaving it.

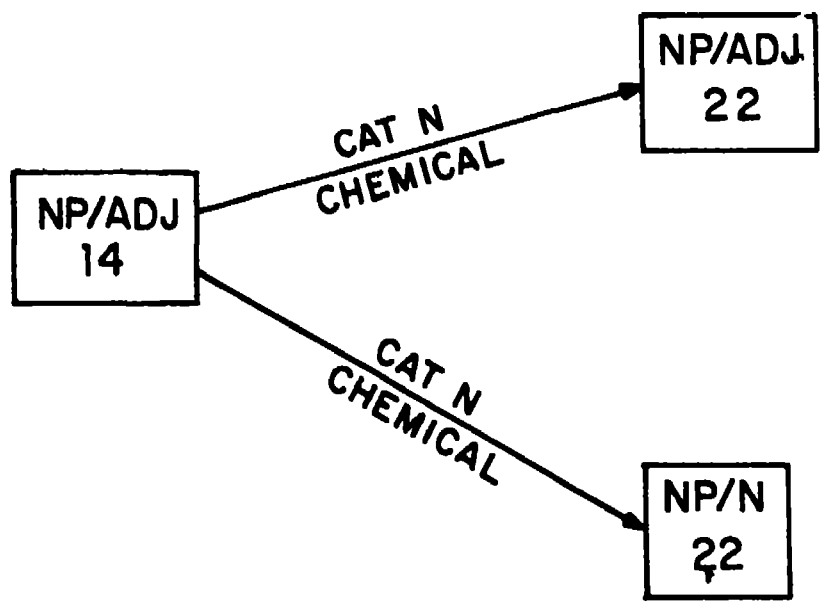

Figure 4.4

Initial map for parsing "chemical analyses"

The idea of this process is to begin to set up paths which may be used to parse the island. However it is not necessarily the case that the only configurations which could start paths through the island are those which have just been obtained, since it may be possible to create transitions which enter them via JUMP arcs or TST arcs. For each state, the sequence of arcs which can reach it without using the previous word of input have been be pre-calculated by the grammar indexing package so the appropriate configurations and transitions may be constructed. These transitions are called lead-in transitions. Thus the map becomes that in Figure 4.5

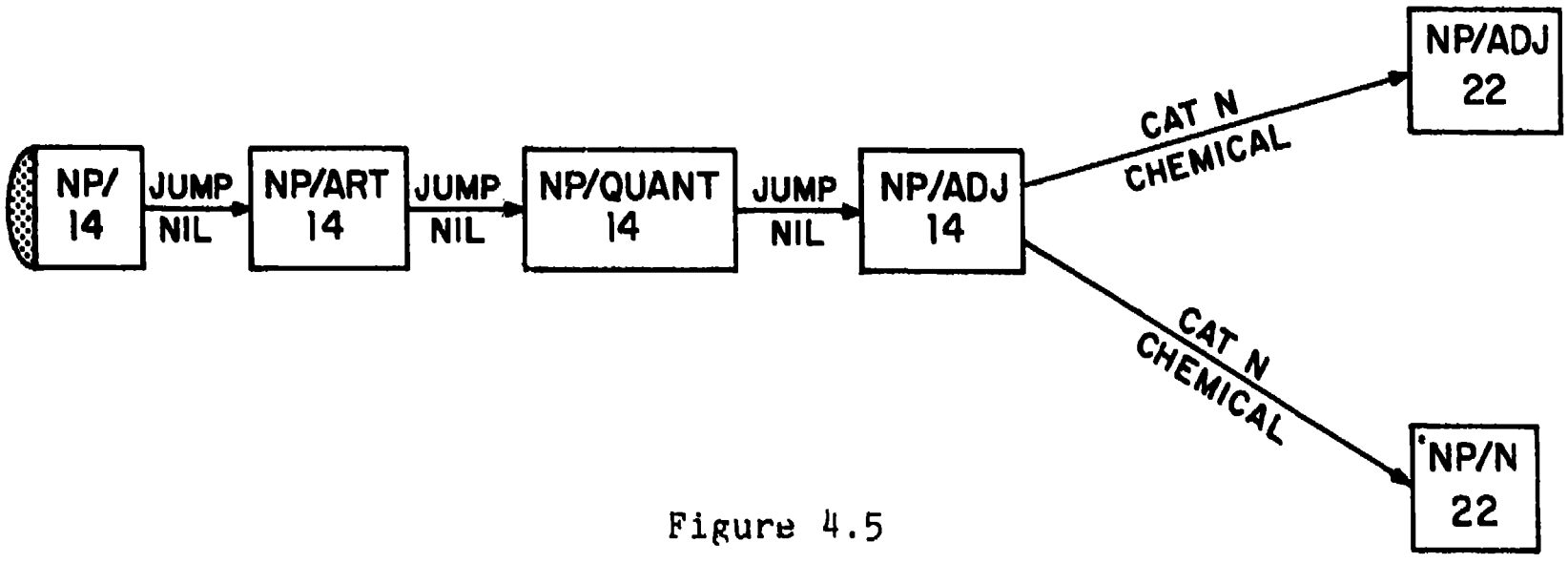

Figure 4.5

Lead-in transitions for parsing "chemical analyses"

Note that any of the configurations (except for NP/ADJ:22 and NP/N:22) could actually be the correct leftmost configuration for this island, depending upon what the (currently unknown) left context of the island is.

By looking in the grammar index, SPARSER can determine, for each configuration which could start the island, just what sort of left context could be appropriate. For example, the CAT ADJ arc in MINIGRAMMAR which enters state NP/QUANT implies that an

adjective could precede the island and, if it did, the transition which would process it would terminate on configuration NP/ADJ:14.

Because the initial configuration NP/:14 could start the island, anything which could precede a noun phrase could occur to the left; again the grammar index provides the information that the CAT PREP arc could lead to a configuration which could accept a noun phrase (via the PUSH NP/ arc), so a preposition could also prefix the island. If the index functions indicate that a constituent could be picked up by a PUSH arc which could terminate on the configuration under consideration, an indication is made in the WFST so that any time a constituent of the desired type is built which ends at the proper location, it may be tried here.

Because of the highly recursive nature of ATN grammars, it is very likely that as we chain back through the possible sequences of PUSHes which could lead to the beginning of the current constituent (or the sequence of POPs which could be initiated by the completion of the current constituent) a large number of predictions will be made. Rather than make all these predictions automatically, before we are even sure that there is in fact a constituent at the current level, the possible configurations which could make predictions on other levels are saved to be activated later if the predictions from the current set of active configurations are not sufficient.

The predictions which are made (not saved) are not acted upon at this time, but are kept internally by SPARSER until all the islands of the theory have been processed. We shall see below what then becomes of the predictions.

Parsing Through an Island

Once processing has proceeded this far, we can go back and consider the set of configurations which represent states the parser could be in just after processing the first word of the island. In our example, these are configurations NP/ADJ:22 and NP/N:22. Configurations such as these which are waiting to be extended to the right are called active configurations. SPARSER selects a subset of the set of active configurations (how this subset is selected will be discussed in the next section) and for each configuration tries to extend it by trying to parse the rest of the island beginning in that configuration. When the parser is considering a configuration at some position, the input pointer is set to the word match of the island, if any, which begins at the same position in the input.

The grammar associates with the state of the configuration a list of arcs which may be tested (using the arc type, the context free test on the arc, and the current input) to determine whether a transition can be made to extend the path. We will consider each type of arc in turn, since the effects of taking various types of arcs are different, and explain for each case what happens if the arc is taken. Whether just one transition, or several, or all possible transitions are made from an active

configuration is a matter to be discussed in Section Five.

Some JUMP arcs do not look at the current item, so they may be taken whether the input pointer is set to a word match or to NIL. The transition which results from taking an arc of this type has a null item associated with it, even if there is a word match in the theory at this point. The positions of the configurations at each end of the transition are the same; this corresponds to the fact that an ATN parser would not move the input pointer as a consequence of taking this arc.

Rarely, a JUMP arc may test the current item in some way, for example, to make a feature check. If there is no word match for input, an arc of this type cannot be taken. If there is a word match, it is noted on the transition wh ch is created, but the configurations at each end of the transition have the same position. (It is then the case that theinext input-using or input-consuming transition on the path including this transition must use the same word match.)

These are TST, CAT, and WRD arcs which end in a (TO nextstate) action. The operation is exactly the same as that above except that the configuration on which the transition terminates has the position of the right boundary of the current word match.

Taking a POP arc results in the creation of a transition which has a null final configuration and a null item, because POP arcs are not permited to look at input.

When a PUSH arc is encountered, a monitor is placed in the Well-Formed Substring Table (WFST) at the current position to await the occurrence of a constituent of the required type. If one or more such constituents are already in the table, then for each one there are three possibilities: it may be composed of word matches which are in the current theory, it may be composed of word matches some of which are not in the current theory but which could be added without violating the non-overlapping constraint, or it may be composed of word matches some of which are incompatible with the current theory.

In the first case a transition is set up using the constituent as the current word. The transition terminates on a configuration whose state is determined from the termination of the PUSH arc and whose position is that of the right boundary of the rightmost word match in the constituent.

In the second case, a notice is created and sent to the control component. A notice is a request that SPARSER be called to enlarge a theory by adding some new information, in this case, some additional word matches which form a constituent that the theory can use. SPARSER does not try to determine when (or even whether) the theory should be so enlarged. That is an issue for the main controller to decide (see Rovner, et.al. [23]). We will discuss below how SPARSER enlarges a theory if called upon to do so.

In the final case, if there are no usable constituents in the WFST, a new configuration is set up to start looking for one and is added to the list of active configurations. Its state is

the state specified by the PUSH arc and its position is the same as the current configuration.

There is a considerable amount of processing that can happen any time one of the transitions just discussed is made. Whenever an initial configuration is constructed, this fact is recorded in the configuration. Whenever a transition is made from such a configuration, the information that there is a path from some initial configuration is recorded on the subsequent configuration. Similarly, whenever a POP transition is made, the configuration it emanates from and all previous configurations on any path which can terminate with the POP transition are marked to indicate that they can reach a POP transition. Whenever a transition is made which completes a path from an initial configuration to a POP transition, the path is executed, one transition at a time, and the register setting actions and context sensitive tests are executed. If a test fails or an arc aborts, the transitions and configurations of the path are removed from the list of configurations and transitions which are in the syntactic part of the current theory (unless they are used by another path in the theory) but not removed from the map. If the execution is successful, a deep structure tree is produced. That structure together with its features is given a score, which may include evaluations by other components such as semantics and prosodics, and is entered in the WFST.

It is quite important that sources of knowledge other than syntax be called upon to verify and to rank syntactic constituents. This is because there are likely to be many

combinations of plausible words from the word lattice which form syntactically reasonable constituents but which may be ruled out on other grounds. To allow immediate use of this information which syntax cannot provide alone, SPARSER has an interface to the semantic component so that constituents can be verified directly without going through the control component. It will be a trivial modification to insert verification calls to pragmatics and prosodics when they become available. In the meantime, even semantic knowledge can be turned off; if the parser gets no information from the call to semantics, it proceeds without it.

Placement of a constituent in the WFST causes a number of things to happen. First, any monitors which have been set by the current theory at that position are activated. That is, for each configuration which was waiting for this constituent, a PUSH transition is made which uses the constituent as its input item. If no monitors have been set which can use this constituent, it is treated exactly as if it were the first word of an island: all the PUSH arcs which can use it are found in the grammar index and appropriate configurations and transitions (including lead-in transitions, if appropriate) are set up. Next, if there are any monitors for other theories which can use the constituent, notices are created and output to Control as was described above in the section on PUSH transitions.

Figure 4.6 shows SPARSER s map after our example island has been completely processed. The parsing results in the creation of a CAT N transition to configuration NP/N:30 using the word "analyses" The PUSH PP/ arc at state NP/N would cause

configuration PP/:30 to be created. Similarly, PP/:22 would be created when the configuration NP/N:22 is picked up to be extended. The POP arc transitions from each of the configurations for state NP/N result in the formation of complete paths, resulting in the creation of two noun phrases ("chemical analyses" and "chemical"). Since there were no monitors for them, they result in the creation of configuration PP/PREP:14 and its subsequent paths.

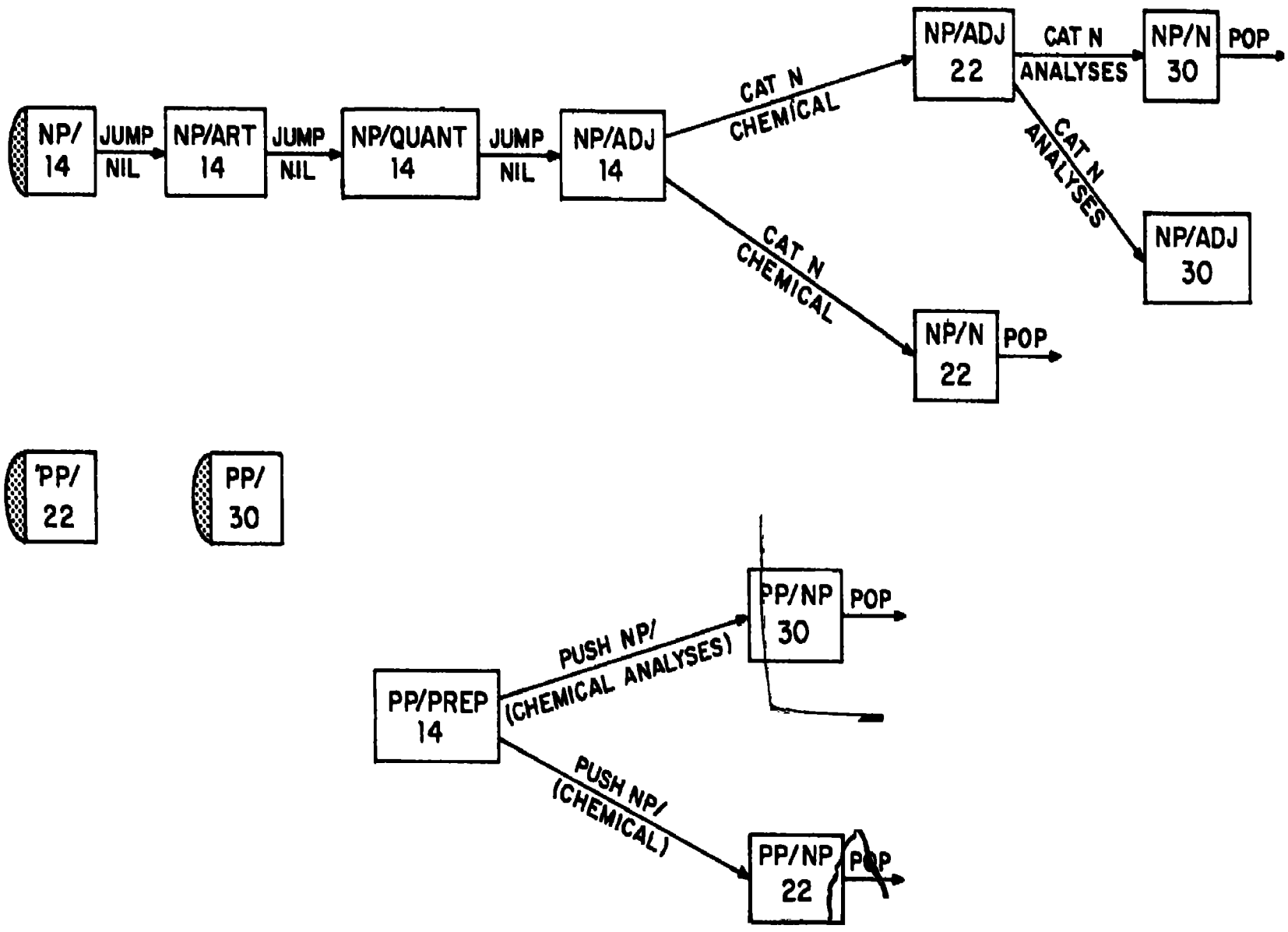

Figure 4.6

Map after processing island

Ending an Island

It may be the case that no path can be found from one end of an island to the other. (This would occur when all active configurations block.) In this case, there is no possible way that the island could form part of a grammatical string, so SPARSER can inform the control component that the theory is wrong.

When an active configuration is picked up to be extended and there is no word match at that point, the end of the island has been reached. That does not mean that no more transitions can be made, since arcs which do not test the input word can be taken as usual. Arcs which do use input cannot be taken, but they can be used to predict what sort of input would be acceptable at that position. For example, a CAT V arc which has a test requiring the verb to be untensed would allow SPARSER to predict an untensed verb beginning at the position of the current configuration. CAT and WRD arcs cause the prediction of syntactic categories and specific words, respectively, modified by the context-free test on the arc. TST arcs provide only the test which must be satisfied, and PUSH arcs cause a monitor to be set in the WFST as well as a TST monitor for the the look-ahead test (if any) on the arc.

Ending a Theory

When all the islands of a theory have been processed in the manner just described, it is time to deal with the gaps between the islands. As we have seen, arcs in the grammar which can

enter configurations at the left end of an island or which can leave configurations at the right end of an island can be used to make predictions about words that may be adjacent to the island. The prediction is a list of the arc, the configuration it would connect to, and an indication of whether the transition caused by the arc will enter the configuration from the left or leave it to the right.

If a gap between two islands is small enough that it may contain just one word, then it is likely that the arc which would process that word may have caused a prediction from both the left and right sides of the gap. If this is the case, and if the predictions intersect in a single possibility, it is highly probable that the word (or syntactic class) so predicted is correct. If the predictions do not intersect, parsing is continued from the active configurations which were not tried earlier because of their scores and from the configurations which could begin constituents at the right end of an island. This continued parsing is an attempt to find a path which results in a common prediction across the gap. If that too fails, then the configurations which were saved because they could lead up a chain of PUSHes or POPs to new configurations are tried. If no possibilities are left to try and there is still no prediction to fill the gap, this information is noted, but it does not definitely mean that the islands are incompatible, since in some cases the gap could actually be filled by two words instead of one.

SPARSER has two kind of predictions - those which seem highly likely and those which seem less likely. A highly likely prediction, such as one which is made from both sides of a small gap, is output in the form of a proposal, which is a request to the rest of the system to find a word meeting the requirements of the proposal. A proposal contains:

1) the item being proposed, which is either a particular word or list of words (from a WRD arc), or a syntactic class (from a CAT arc), or NIL, meaning any word (from a TST arc)

2) the left and/or right boundary point(s) of the item

3) a test which the item must satisfy (the context free test from the arc)

4) the context of the proposal, i.e. the word match(es) on the left and/or right side of the item being proposed. (This is to help the lexical retrieval component take into account phonological phenomena which may occur across word boundaries.)

All predictions whether or not they are confident enough to become proposals are output as monitors. A monitor is a notification to the control component that if a word meeting the requirements of the monitor is somehow found (perhaps by the action of a proposal) , it may be added to the theory. Thus a monitor acts like a demon which sits at a particular point in the word lattice and watches for the appearance of a word match which it can use. A monitor contains:

1) the item being monitored for (generally a syntactic categcry, but may be a word or a test)

2) the left or right boundary position of the item being monitored for

3) a test which the item must satisfy (same as for proposals)

4) the theory which generated the monitor

5) the arc in the grammar which will process the item if found

6) the configuration from which the prediction was made

7) a score, indicating roughly how important the monitor is, i.e. how much information is likely to be gained by processing an event for that monitor.

(Notice that monitors which are sent to the control component are very much like monitors which are set in the WFST by the occurrence of PUSH arcs.)

Once the proposals have been made and the monitors have been set, SPARSER bundles up the information it knows about the current theory, such as the configurations and transitions in the theory, any configurations which are still candidates for expansion, the constituents in the theory, the notices, proposals, and monitors which have been created, etc. and associates the bundle with the theory number. This insures that SPARSER will be able to pickup where it left off if it is later given the theory to process further.

Processing Multiple Theories

Thus far we have seen only the operations which SPARSER performs on a single theory, but we made the assumption that SPARSER would be given a number of theories to process in sequence. Let us now examine what will happen when the second (or nth) theory is processed.

SPARSER will no longer have a blank map and WFST; instead it will have all the configurations, transitions, and constituents which have been constructed by all previous theories. For concreteness, let us imagine that the theory (1 CHEMICAL 14 22) (2 ANALYSES 22 30) has been processed, resulting in the map shown in Figure 4.6. Now we are going to process a theory containing the island (4 NICKEL 16 22) (2 ANALYSES 22 30), which results in the map of Figure 4.7 where the configurations and transitions added by this theory are shown in dotted lines.

The process begins as usual with the creation of configuration NP/ADJ:16 and three possible lead-in transitions. The transitions for the two CAT N arcs, however terminate on configurations which already existed in the map, so the complete paths from configuration NP/:16 to configurations NP/N:30 and NP/N:22 will be discovered and processed, resulting in the construction of two new noun phrases. Those new constituents would then result in the creation of configuration PP/PREP:16 and two new transitions. Thus we have constructed only five new configurations and seven new transitions and have been able to take advantage of six old configurations and six old transitions.

In this fashion any information which has once been discovered about a possible parse path is made available to any other path which can use it. <u>No reparsing is ever done</u> SPARSER merely realizes the existence of relevant configurations and transitions and incorporates them into the current theory.

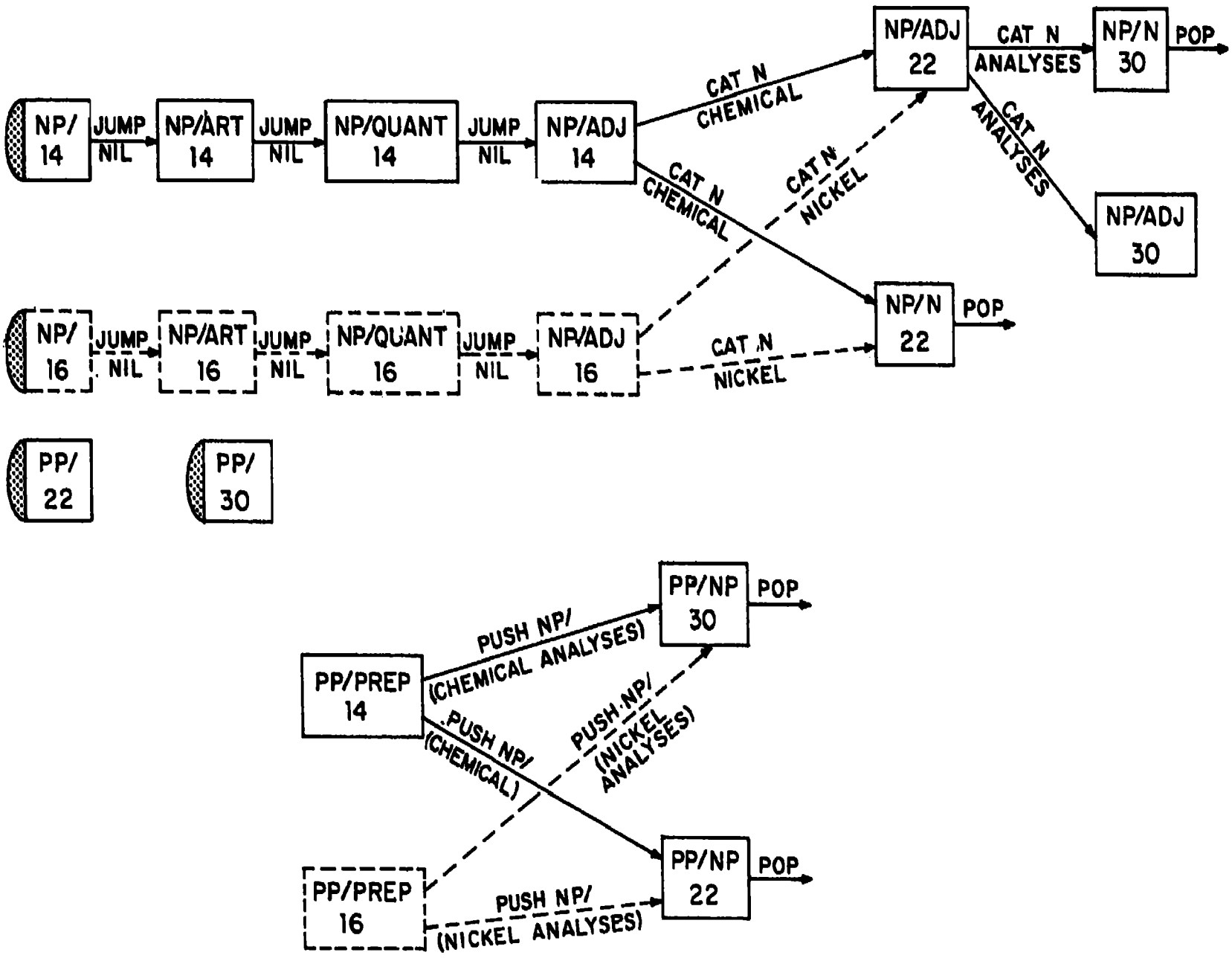

Figure 4.7

Map after processing island for "nickel analyses"

If the new word (or words) in a theory are at the end (or in the middle) of an island, when SPARSER begins to parse the island it will discover the existing configurations and transitions from the previous theory. Whenever a transition which can be used in the current theory is discovered in the map, it and its terminating configuration are added to the syntactic part of the current theory. This is called tracing the transition. In addition, all paths beginning with that transition which do not require the next word of input are also included in the syntactic part of the theory. This is accomplished by tracing from the terminating configuration all transitions which use either the

same word of input as the previous transition or no input word at all. (A similar process is used to trace backwards, i.e. right to left, when necessary.) When a configuration is reached which has no traceable transitions emanating from it, the tracing process stops. Since both transitions and configurations are stored in such a way as to facilitate tracing (for example, each transition has a code attached to indicate whether or not it consumes or tests input), this process is considerably faster than creating that portion of the map in the first place. (To illustrate this, a theory was processed twice, once with an empty map and once starting with the map previously created; the time required for processing the theory fell from 47.5 seconds to 16.5.)

Configurations which can end traced paths are put on the active configurations list. If, when one of them is picked up for extension, it is discovered that the next word of input was used on a transition already in the map, the tracing process is repeated. If the next word of input is new (or at least has not caused any transitions from the configuration being considered) then parsing continues in the normal manner.

Processing Events

As was mentioned earlier, SPARSER can be called upon to add some new word matches to a theory it has previously processed. In this case, SPARSER is said to process an event. An event may be thought of rather abstractly as the discovery of a piece of information that has been syntactically proposed, monitored for,

or noticed. Concretely, an event is a piece of data consisting of:

1) the old theory that proposed or set a monitor for the event

2) something to be added to the theory (a new word match or constituent)

4) the arc in the grammar which will process the new information

4) the configuration in the old theory which will be at one end of the transition created by the above arc

When SPARSER is given an event, it retrieves from its tables the bundle of configurations, transition, etc. in the old theory. Then using the arc and the new word or constituent in the event, it creates the appropriate transition(s). Then processing continues as usual, that is, any complete paths are noticed and processed, and any new active configurations are extended, if possible.

New predictions may be made as a result of this increased information. (A record is kept of previous predictions so none are remade unless with a more liberal score.) Finally SPARSER returns the new, larger theory. This new theory may be processed as part of another event at some later time, thus gradually reducing the number and size of the gaps in the theory.

If an event results in filling the final gap in a theory, and if the resulting complete sequence of words can be parsed, SPARSER notifies the control component of this fact, since the entire utterance may have been discovered. Of course, this may

not be the correct solution -- it is up to the control component to look at the acoustic goodness, semantic meaningfulness, pragmatic likelihood, etc. of the result as well as the syntactic structure before declaring the utterance to have been understood. If for reasons other than syntactic, the utterance appears to be bad, the control component of the system could go on to try to find another, more suitable, possibility.

Section 5

More Details of the Parsing Process

5.1 DEPTH vs BREADTH

The parsing strategy just outlined works bottom up when beginning to parse an island and when a constituent is created which was not monitored for by the current theory. It works top down after an island has been started and to make syntactic predictions at the ends of islands. Both top down and bottom up techniques can be either depth or breadth first. Depth first processing takes at every step the first piece of information available and pursues its consequences. Breadth first processing considers at every step every possible next step of every alternative and pursues all paths in parallel. Breadth first processing generally takes much more space than the depth first process since many paths would have to be remembered at once

instead of having just one stack which could be popped and reused when necessary.

The breadth first process might save some computation steps and might produce several ambiguous parsings simultaneously while the depth first process would find one before the others (the latter is a small difference, since both processes would have to be run to exhaustion to insure that all possible parsings had been found). In parsing speech, some mixture of breadth first and depth first processing can be extremely useful.

To illustrate an advantage of breadth first processing in the speech environment, consider what might happen if, during the processing of an island the parser picks up a configuration to extend which has several possible arcs emanating from it. If one arc is chosen and all the others are held as alternatives (i.e. depth first), but the chosen arc is wrong, all subsequent paths beginning with that arc would have to block before the alternatives would be tried. However, if the end of the island were reached before the success or failure of the first choice were confirmed, the only way that backup would ever take place would be to have one or more events add words to the theory so that the path could be extended until it failed. Since the gap would be likely to be filled by (incorrect) words predicted by the erroneous path, or by no words at all if the (incorrect) predictions were not satisfied, it is not at all clear how the process would ever know to back up.

This problem cannot be eliminated completely without pursuing all alternatives to their fullest extent (a combinatorially unacceptable solution) but it can be modified to a great extent by a judicious combination of depth and breadth first processing to find the best path, not just the first one, through the island. This "best path" is not guaranteed to be the correct one, so it is possible to continue processing by extending paths with were suspended earlier.

SPARSER handles the problem by assigning a score to every configuration which reflects the likelihood of the path which terminates on that configuration to be correct. The score can also be thought of as a measure of how good that configuration looks in relation to others as a candidate for extension. One question which was previously left unanswered, how a subset of the active configurations is chosen for extension, can now be answered: the subset of maximally scoring configurations is chosen at each step until the maximal score of active configurations begins to fall. (The score on a configuration and the score of a path terminating on that configuration are the same thing -- we will use which ever terminology seems most natural at the time.)

The result of this process is a sort of modified breadth first approach, where at one step all the alternatives are tried but at the next step only the best ones are chosen for further extension. This is similar to the best-first parser described by Paxton in [18] but it can be applied to the sort of partial paths which SPARSER generates rather than requiring the perfect

information resulting from a strictly left to right approach. The success of this method is directly dependent on the relative accuracy of the scores which are assigned to the paths.

5.2 SCORING PATHS

Several attempts have been made to develop rigorous systems for parsing errorful or speech-like input based on probabilities [1, 14, 27]. These attempts have all simplified the problem to such an extent that it is no longer realistic or extendible, e.g. by assuming the input is a sequence (rather than a lattice) of probability distributions, by assuming that all the necessary information is present in the search space to begin with so the only problem is to find an optimal path through the space, by requiring a small vocabulary, and/or by limiting the grammar to be context free.

The ideal scoring mechanism for SPARSER would be one which accurately reflected at every step the probability that the path was correct. Bayes rule could be used, but it would be necessary to know, at any point in the parsing process, what the probability is that the next arc under consideration is correct, given that the entire path up to the current step is correct. In order to use this application of Bayes rule it would be necessary to pre-calculate the probabilities for every possible path and partial path which could be generated -- a clearly impossible task since there are an infinite number of such paths.

Given that we cannot calculate the probabilities we need exactly, what is the next best option? If we ignore the effect of the path traversed up to the current point, but can say for any given state how likely each arc emanating from that state is to be correct, we would have a model which uses only local information rather than one which takes into account accurately all the left context which is available.

Since it was not practical to run large amounts of data through a parser in order to obtain accurate measurements even for the limited model, the author relied on considerable experience with ATN grammars to assign a weight to each arc of the grammar representing the intuitive likelihood that the arc (if it can be taken) is the correct one to choose from that state. These weights are small integers (0 through 5) -- the larger the weight the more likely the arc.

The question might arise as to why the score of the word match used by an arc should not be used to influence the score of the path using it. SPARSER tries to treat each theory as independently as possible and to assign scores based only on the syntactic information which is available. The one exception to this rule is the semantic information which is used to score constituents. If lexical word match scores were used, the control component would not be able to separate the lexical goodness from the syntactic goodness of the theory and make judgments as to their relative importance. In a syntax-driven speech understanding system, however, it would probably be useful to combine lexical scores with syntactic information.

As was described in the previous section, when SPARSER begins to parse an island each possible partial path is begun by creating a configuration at the head of a transition for an arc which can use the current word. Rather arbitrarily, it was decided to give this configuration a score of one. This starts all partial paths out equally, a technique which is not quite accurate, since some contexts are more likely than others. For example, the words "to" and "for" are more likely to occur in prepositional phrases than in sentential complements. If this simplification appears to harm the overall performance of SPARSER, it could be remedied by giving each state an a priori score similar to the weights on arcs. Configurations on lead-in paths are also given a score of one.

After the initial step, whenever a transition (other than a PUSH or POP) is made, the score of the subsequent configuration is influenced by the score of the configuration being extended and the weight on the arc being used. If the scores were actual probabilities; they would be multiplied; since they are not, it was arbitrarily decided to add them.

When attempting to create a configuration which already exists (a situation encountered whenever two or more parse paths for the same theory merge), the configuration is given the maximum of the existing score and the score which would have been assigned had the configuration been created anew.

When a PUSH arc is encountered and a configuration created to begin the search for the required constituent, the score of that configuration is set to be the sum of the score of the

configuration causing the PUSH, and the value (if any) of the look-ahead test on the PUSH arc. For example, upon encountering an arc such as (PUSH NP/ ((NPSTART) T T) ...) the look-ahead function NPSTART returns a high integer value if the next word is a noun and a lower value if it is a verb (e.g. "accounting costs"). Of course, if the look-ahead function fails altogether, the configuration is not set up, although the monitor in the WFST remains.

When a constituent is completed (or found in the WFST) and a PUSH transition is about to be made, the score of the configuration on which the transition terminates is a function of the score of the configuration being extended the weight on the arc, and the score of the constituent itself. The score of the constituent is currently very ad hoc, being a function of the number of words in the constituent (less a function of the number of sub-constituents subsumed by this constituent, boosted if the constituent is a major one) and the score which is determined by semantic verification. Thus semantically "good" constituents will boost the scores of the paths which use them more than semantically "bad" ones.

Due to the level of effort required to gather accurate statistics on the relative frequencies of arcs, the current scores are admittedly ad hoc. It is not clear whether different scoring mechanisms would be better, however it is clear that the current scoring strategy is better than no scoring at all, as preliminary measurements indcate that the number of transitions created (as well as the number of configurations and predicions)

is reduced about 25% by the current strategy.

(It is reasonable to ask why semantic scores are used to influence parse paths, since it was just argued that lexical scores should not be used in this way Semantic scores may be more reliable than lexical ones because we are assuming that the utterance is semantically meaningful. Under this assumption, a constituent like "range remainder" as a noun-noun modifier analogous to "surplus money" should be ruled out as early as possible. Since such constituents cannot be ruled out on syntactic grounds alone, since prosodic information (which might help to rule them out) is not available (see discussion in Section 7.2), and since they would seriously overrun the parser with a plethora of false paths if they were not rejected, it seems reasonable to permit semantics to influence the parser.)

5.3 SCORING PREDICTIONS

The previous section discussed three ways in which SPARSER can make predictions about what could fill in gaps between islands. Monitors wait for the occurrence of a word in the word lattice (or a constituent in the WFST), proposals request a search for a particular set of words, and notices indicate the presence of a usable word in the word lattice (or a constituent in the WFST). Since the processing of a typical theory is likely to result in a number of predictions it is necessary to be able to order them so that predictions most likely to be correct or most likely to yield important information will be acted upon

first. For example, it is more important to fill a gap between two islands than to extend a single island, since by filling the gap one can check the consistency of information which was locally good in each island individually but may not be consistent when they are joined. Since two words can occur together in (usually) many contexts but longer sequences are generally more restrictive, adding a word to a one word island is likely to be less profitable in terms of the number of possible paths which are eliminated by the addition than adding a word to a multi-word island.

It is up to the syntactic component to indicate to the control component the relative importance attached to each notice and monitor; the higher the score, the stronger the prediction.

Several factors influence the score attached to predictions. One is the length of the island to which the prediction is attached. One word islands, if they are processed at all, yield very little information anc many predictions, hence the predictions are not scored high. Proposals are less important if there is already a noticeable word in the word lattice (since that word is acoustically better than the word to be proposed, else it would have been found earlier. However, if a proposal fills a gap between two islands, it is given a higher score. Notices are boosted in importance if an entire constituent may be added and penalized if they will add onto a one word island. Scores range from 0 to 95 for proposals, 0 to 40 for notices, and 0 to 15 for monitors.

These scores appear to work fairly well with the rest of the BBN SPEECHLIS system, but have been developed by a process of interaction with the other components (in order to make the scores of syntactic predictions commensurate with those of semantic predictions) and may be changed considerably as the entire system evolves.

Small syntactic classes (e.g. determiners and prepositions) are proposed in their entirety (that is, their elements are to be enumerated and given to the lexical matching component for verification) if the island which monitored for them is more than one word long. If a gap between two islands is small enough for just one word and if a syntactic class has been monitored for from both sides of the gap, it is proposed ir its entirety also.

Section 6

Examples and Results

SPARSER is written in INTERLISP and runs on a PDP-10 under the TENEX operating system . The program and initial data structures occupy approximately 90000 words of virtual memory. (The other components of the BBN speech understanding system occupy separate forks from the syntactic component.)

At the time the examples in this section were run, the algorithm controlling the decision-making process in the control component was undergoing revision and was not solidified into a function which could operate automatically. Rather, there were a number of primitive operations such as scanning an utterance (or some specified portion of it), creating theories, calling SPARSER with a theory or event, calling for the processing of proposals, etc., which could be invoked by a human simulator The following examples were produced in this mode, with the user acting as the control component in a way which could be modelled by later implementation.

Several conventions have been used in tracing the operation of SPARSER. Configurations are represented as NUMBER : STATE : POSITION (SCORE). For example, the configuration written as 30:NP/HEAD:23(39) is the configuration for state NP/HEAD at position 23 which has been given the (unique) number 30 and which currently has a score of 39. The creation of a transition is indicated by naming the type of arc causing the transition, the (unique) number of the transition, and the configurations at each end of the transition. For example,

CAT N TRANS #9 FROM 14:NP/DET:6(1) TO 15:NP/DET:19(4).

Annotations have been inserted within brackets { }; typeout in upper case was produced by the program.

EXAMPLE 1

This example parallels that given in Section Four. A word lattice was artificially created which contained only the following three word matches:

(1 SUMMER 12 16 100)
(2 WINTER 12 16 100)
(3 TRIP 16 21 100 -S)

(In this version or the system, regular inflectional endings are included in word matches after the element representing the score, hence the somewhat peculiar word match for the word "trips".) Two theories were constructed, one for word matches 2 and 3, the other for 1 and 3. What follows is an annotated (but otherwise unedited except for considerations of spacing) transcript of SPARSER processing these two theories in sequence, using the MINIGRAMMAR of Figure 3.3 and Appendix I.

```
SPARSER PROCESSING THEORY 1:
0  12 WINTER 16 TRIP -S 21  30
        {This is a linear representation of the theory being
        processed.  The endpoints are 0 and 30, but the words
        occupy only the middle part of the utterance.}

STARTING AN ISLAND
"WINTER" TRYING CAT N ARC FROM NP/ADJ TO NP/ADJ
        {This is the first of two arcs retrieved from the index
        tables }
  CAT N TRANS #1 FROM 1:NP/ADJ:12(1) TO 2:NP/ADJ:16(3)
        {The first transition is created, and since there is a
        CAT N arc which enters state NP/ADJ, a monitor is set up
        to monitor for nouns which end at position 12.}
   ENDING AT 12:
    MONITORING [  N  ]
  JUMP TRANS #2 FROM 3:NP/QUANT:12(1) TO 1:NP/ADJ:12(1)
        {Now the lead-in transitions are being created, along
        with the monitors for syntactic categories which may
        precede the newly constructed configurations.
        Configurations along the lead-in path are all assigned a
        score of 1.}
   ENDING AT 12:
    MONITORING [  QUANT  ]
```

```
   MONITORING [  ADJ  ]
  JUMP TRANS #3 FROM 4:NP/ART:12(1) TO 3:NP/QUANT:12(1)
   ENDING AT 12:
    MONITORING [  ART  ]
  JUMP TRANS #4 FROM 5:NP/:12(1) TO 4:NP/ART:12(1)
        {The lead-in transitions are all made.  Now  the  second
        arc which can use the noun is about to be processed.}


"WINTER" TRYING CAT N ARC FROM NP/ADJ TO NP/N
  CAT N TRANS #5 FROM 1:NP/ADJ:12(1) TO 6:NP/N:16(6)
        {This is the second of the two arcs  obtained  from  the
        index  table  for  "winter".  The lead-in transitions to
        configuration 1 have already been constructed,  so  they
        are  not  remade.   Now  we  are  ready  to  choose
        configurations to extend.  The pool  of  candidates  for
        extension contains configurations 2 and 6.}

SELECTED CONFIGS (6) FOR EXTENSION
        {Only this one is chosen because it has a  higher  score
        than  configuration 2, since the use of a noun as a head
        noun of a noun phrase is more likely than its use  as  a
        modifier.}
PICKING UP CONFIG 6:NP/N:16(6) WITH WORD TRIP
 TRYING PUSH PP/ ARC
        {No action is taken about starting a  configuration  for
        state  PP/ because the look-ahead test which checks that
        the next word can begin a prepositional phrase fails  on
        the word trip.}
TRYING POP ARC
 POP TRANS #6 FROM 6:NP/N:16(6)
        {Creating the  POP  transition  completes  a  path  from
        configuration  5.   The  path  is expressed as a list of
        transition numbers.  We are about to execute  the  path,
        that  is,  check  the context-sensitive tests and do the
        register building actions along it.}
   EXECUTING PATH (4 3 2 5 6)
    BEGINNING AT TRANS 4, CONFIG 5
        {We  must  begin  executing  the  path  at  the  first
        transition,  because  no  part  of  it has been executed
        before.  Later we will see that it is possible to  begin
        execution  of  a  path in the middle, since the register
        contents are stored at each step.}
   DOING JUMP ARC FROM 5:NP/ TO 4:NP/ART
   DOING JUMP ARC FROM 4:NP/ART TO 3:NP/QUANT
   DOING JUMP ARC FROM 3:NP/QUANT TO 1:NP/ADJ
   DOING CAT ARC WITH WINTER FROM 1:NP/ADJ TO 6:NP/N
   DOING POP ARC FROM 6:NP/N
    TEST FAILED
        {The test failed because there  is  no  determiner,  and
        MINIGRAMMAR  requires  that singular, undetermined nouns
        can be complete noun  phrases  only  if  they  are  mass
        nouns.   "Winter"  is  not  marked as a mass noun in our
        dictionary, hence it will not parse as a  complete  noun
        phrase.}
```

```
SELECTED CONFIGS (2) FOR EXTENSION
          {Since extending configuration 6 did not do much for us,
          we go back to try the lower scoring configuration 2.}
PICKING UP CONFIG 2:NP/ADJ:16(3) WITH WORD TRIP
 TRYING CAT N ARC
  CAT N TRANS #7 FROM 2:NP/ADJ:16(3) TO 7:NP/N:21(8)
 TRYING CAT N ARC
  CAT N TRANS #8 FROM 2:NP/ADJ:16(3) TO 8:NP/ADJ:21(5)


SELECTED CONFIGS (7) FOR EXTENSION
          {Again, the higher scoring of the two active
          configurations, 7 and 8, is chosen.}
PICKING UP CONFIG 7:NP/N:21(8)
   STARTING AT 21:
    MONITORING [  PP/  ]
     SETTING UP CONFIG 9:PP/:21(8)
    MONITORING [ PREP ]
          {Since there is no next word to test, a configuration is
          set up to begin processing a prepositional phrase, and
          the syntactic categories which can begin such a phrase
          -- in this case, only one -- are monitored for.}
 TRYING POP ARC
  POP TRANS #9 FROM 7:NP/N:21(8)
    EXECUTING PATH (4 3 2 1 7 9)
     BEGINNING AT TRANS 1, CONFIG 1
          {Creation of the POP trans completed a path, the first
          part of which has already been executed. We can
          therefore pick up in the middle of the path and execute
          only the last three transitions.}
    DOING CAT ARC WITH WINTER FROM 1:NP/ADJ TO 2:NP/ADJ
    DOING CAT ARC WITH TRIP FROM 2:NP/ADJ TO 7:NP/N
    DOING POP ARC FROM 7:NP/N
****
    MADE #1 FROM 12 TO 21:
    NP ADJ NP N WINTER
               NU SG
       N TRIP
       NU PL
****
          {The path succeeds -- no determiner is needed since the
          head noun is plural -- and a constituent is constructed.
          The semantic component has been turned off for this
          example, so it adds nothing to the score which SPARSER
          assigns -- 5 points for each word in the constituent.}
SYN WEIGHT + SEM WT = 10 + 0 = 10


          {No monitor exists in the WFST for a NP/ at this place,
          so the arcs (in MINIGRAMMAR there is only one) which
          could push for a NP are processed bottom up in exactly
          the same manner as the two arcs which could use a noun
          at the beginning of the island.}
NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #10 FROM 10:PP/PREP:12(1)
   TO 11:PP/NP:21(7)
   ENDING AT 12:
```

```
    MONITORING [  PREP  ]
                          |
SELECTED CONFIGS (11) FOR EXTENSION
PICKING UP CONFIG 11:PP/NP:21(7)
 TRYING POP ARC
  POP TRANS #11 FROM 11:PP/NP:21(7)


SELECTED CONFIGS (8) FOR EXTENSION
PICKING UP CONFIG 8:NP/ADJ:21(5)
   STARTING AT 21:
    MONITORING [  N  ]
    MONITORING [  N  ]
 ALL ARCS TRIED AT THIS CONFIG
        {Now the theory has been processed.  There follows a
        summary  of  the  proposals, monitors, and notices
        constructed.  The syntactic score assigned to the theory
        is given -- here just the score of the constituent
        constructed.  Then there is a summary of statistics.}


PREDICTIONS:
MONITORING [  PREP  ] STARTING AT 21, SCORE 10
MONITORING [  N  ] STARTING AT 21, SCORE 10
MONITORING [  N  ] ENDING AT 12, SCORE 10
MONITORING [  QUANT  ] ENDING AT 12, SCORE 10
MONITORING [  ADJ  ] ENDING AT 12, SCORE 10
MONITORING [  ART  ] ENDING AT 12, SCORE 10
MONITORING [  PREP  ] ENDING AT 12, SCORE 10
PROPOSING (QUANT ART PREP) ENDING AT 12

FINISHED THEORY 1 WITH SYN SCORE 10

        {Exclusive  of  tracing  and  fork  interactions,  this
        processing took 5.5 seconds.}



        {Now  we  are  ready  to  process  the  second  theory
        syntactically.}

SPARSER PROCESSING THEORY 2:
0  12 SUMMER 16 TRIP -S 21  30

STARTING AN ISLAND
"SUMMER" TRYING CAT N ARC FROM NP/ADJ TO NP/ADJ
  CAT N TRANS #12 FROM 1:NP/ADJ:12(1) TO 2:NP/ADJ:16(3)
        {This  transition  completes  a  path  which  includes
        transitions  and  configurations  constructed  during the
        previous theory.}
    EXECUTING PATH (4 3 2 12 7 9)
     BEGINNING AT TRANS 12, CONFIG 1
    DOING CAT ARC WITH SUMMER FROM 1:NP/ADJ TO 2:NP/ADJ
    DOING CAT ARC WITH TRIP FROM 2:NP/ADJ TO 7:NP/N
****DOING POP ARC FROM 7:NP/N
```

```
    MADE #2 FROM 12 TO 21:
    NP ADJ NP N SUMMER
               NU SG
       N TRIP
       NU PL
****
SYN WEIGHT + SEM WT = 10 + 0 = 10


NP/ WAS PUSHED FOR AT CONFIG 10
          {This time there are monitors in the WFST, one which is
          looking for a NP starting at position 12 and one which
          is looking for a NP ending at position 21. One
          transition is sufficient to satisfy both of these, and
          the preposition needed to complete a PP/ is monitored
          for.}
 PUSH NP/ TRANS #13 FROM TO:PP/PREP:12(1)
   TO 11:PP/NP:21(8)
NP/ MAY LEAD TO CONFIG 11
          {This is caused by the fact that there was a monitor for
          a noun phrase ending at configuration 11 -- the one
          created when constituent 1 was made. The transition
          which would be set up is the transition just created, so
          it is not remade.
             All of the processing which resulted from the
          completion of a constituent is finished; however there
          are monitors still to be set for configurations along
          the path.}
    ENDING AT 12:
     MONITORING [  PREP  ]
    ENDING AT 12:
     MONITORING [  N  ]
    ENDING AT 12:
     MONITORING [  QUANT  ]
     MONITORING [  ADJ  ]
    ENDING AT 12:
     MONITORING [  ART  ]
          {Since each monitor consists of the item being monitored
          for, its associated test (if any) the theory which is
          to be notified when the monitor is satisfied, and the
          configuration and arc causing the monitor, monitors must
          be made anew each time one of the elements changes,
          although some of the list structure can be shared, hence
          the seeming proliferation of monitors.}


          {Now SPARSER processes the other arc which could use the
          word "summer".}
"SUMMER" TRYING CAT N ARC FROM NP/ADJ TO NP/N
  CAT N TRANS #14 FROM 1:NP/ADJ:12(1) TO 6:NP/N:16(6)
    EXECUTING PATH (4 3 2 14 6)
     BEGINNING AT TRANS 14, CONFIG 1
    DOING CAT ARC WITH SUMMER FROM 1:NP/ADJ TO 6:NP/N
    DOING POP ARC FROM 6:NP/N
     TEST FAILED
          {Because "summer" cannot be a complete noun phrase in
          this grammar.}
```

```
SELECTED CONFIGS (11) FOR EXTENSION
PICKING UP CONFIG 11:PP/NP:21(8)
  TRACING POP TRANS 11 FROM 11:PP/NP:21(8)
```

{This transition was created before, but is now made part of the current theory. It does not complete a path or cause any further action. If it had a terminating configuration, i.e. if a transition other than a POP transition had been traced, the terminating configuration would have been placed on the list of possible configurations to extend.}

```
SELECTED CONFIGS (6) FOR EXTENSION
PICKING UP CONFIG 6:NP/N:16(6) WITH WORD TRIP
  TRACING POP TRANS 6 FROM 6:NP/N:16(6)


SELECTED CONFIGS (2) FOR EXTENSION
PICKING UP CONFIG 2:NP/ADJ:16(3) WITH WORD TRIP
  TRACING CAT N TRANS 8 USING "TRIP" FROM
   2:NP/ADJ:16(3) TO 8:NP/ADJ:21(5)
   STARTING AT 21:
    MONITORING [  N  ]
    MONITORING [  N  ]
```

{There are two noun arcs leaving state NP/ADJ, hence two monitors.}

```
SELECTED CONFIGS (8) FOR EXTENSION
PICKING UP CONFIG 8:NP/ADJ:21(5)
 ALL ARCS TRIED AT THIS CONFIG


PREDICTIONS:
MONITORING [  N  ] STARTING AT 21, SCORE 10
MONITORING [  PREP  ] ENDING AT 12, SCORE 10
MONITORING [  N  ] ENDING AT 12, SCORE 10
MONITORING [  QUANT  ] ENDING AT 12, SCORE 10
MONITORING [  ADJ  ] ENDING AT 12, SCORE 10
MONITORING [  ART  ] ENDING AT 12, SCORE 10
PROPOSING (PREP QUANT ART) ENDING AT 12

FINISHED THEORY 2 WITH SYN SCORE 10
```

{The processing of this theory took approximately 4.5 seconds.}

This example has shown the trace produced by running SPARSER on input which is analogous to the example presented with illustrations of the map in Section Four. The interested reader is urged to draw his own maps while reading the following

examples in order to best understand the dynamic operation of SPARSER.

EXAMPLE 2

This example is more realistic than the previous one -- it shows the operation of SPARSER in the context of an utterance which has been automatically segmented and labeled, with the lexical retrieval and match component in operation. It demonstrates how SPARSER can help to select the best set of words from a more complex word lattice. This example uses the SPEECHGRAMMAR described in [4].

The utterance "What is the registration fee?" was spoken by an adult male speaker in a quite room and was record on tape. The utterance was automatically digitized and passed through the segmentation and labeling routines of the BBN speech understanding system. The initial scan of the utterance, using the lexical retrieval component, produced a word lattice of fifteen entries, including several for inflectional endings. (In this version of the system, they were not combined with the root form into a single word match, and hence could match even without a root word.) The format for a word match is:
(NUMBER WORD LEFT-END RIGHT-END LEXICAL-SCORE).

(2 WHAT 0 3 191)
(3 ONE 0 3 189)
(11 WHEN 0 3 102)
(9 THE 4 6)
(1 REGISTRATION 6 19 237)
(10 REGISTRATION 7 19 103)
(5 HAS 9 12 121)

```
(8 THIS 9 12 115)
(15 THE 9 11 90)
(6 -EST 10 13 118)
(12 -EST 10 14 101)
(13 IS 10 12 97)
(14 -ES 10 12 97)
(7 TRIP 12 17 116)
(4 FEE 19 23 155)
```

The two best matches, for "what" and "registration", appear to be good candidates for a theory, so we begin by building and processing that theory.

```
SPARSER PROCESSING THEORY 1:
0 WHAT 3  6 REGISTRATION 19  23

STARTING AN ISLAND
STARTING AT LEFT END OF SENTENCE
        {Knowing that it is not necessary to go through the
        usual startup procedure for islands when beginning an
        island at position 0, SPARSER starts with a
        configuration for state S/ at position 0.}
SELECTED CONFIGS (1) FOR EXTENSION
PICKING UP CONFIG 1:S/:0(1) WITH WORD WHAT
 TRYING JUMP S/Q ARC
  JUMP TRANS #1 FROM 1:S/:0(1) TO 2:S/Q:0(6)


SELECTED CONFIGS (2) FOR EXTENSION
PICKING UP CONFIG 2:S/Q:0(6) WITH WORD WHAT
 TRYING PUSH NP/ ARC
      MONITORING  [ NP/ ]
         SETTING UP CONFIG 3:NP/:0(11)
 TRYING CAT QWORD ARC
  CAT QWORD TRANS #2 FROM 2:S/Q:0(6) TO 4:S/NP:3(11)


SELECTED CONFIGS (3 4) FOR EXTENSION
        {This time two active configurations have the same
        maximal score, so they are both processed.}
PICKING UP CONFIG 3:NP/:0(11) WITH WORD WHAT
 TRYING CAT QDET ARC .
  CAT QDET TRANS #3 FROM 3:NP/:0(11) TO 5:NP/ORD:3(16)

PICKING UP CONFIG 4:S/NP:3(11)
   STARTING AT 3:
    MONITORING [  MODAL  ]
    MONITORING [  V  ]
```

```
 TRYING POP ARC
  POP TRANS #4 FROM 4:S/NP:3(11)
    EXECUTING PATH (1 2 4)
     BEGINNING AT TRANS 1, CONFIG 1
    DOING JUMP ARC WITH WHAT FROM 1:S/ TO 2:S/Q
    DOING CAT ARC WITH WHAT FROM 2:S/Q TO 4:S/NP
    DOING POP ARC FROM 4:S/NP
     TEST FAILED
          {This test failed because the grammar does not allow
          "what" to be a complete sentence.}


SELECTED CONFIGS (5) FOR EXTENSION
PICKING UP CONFIG 5:NP/ORD:3(16)
   STARTING AT 3:
    MONITORING [  QUANT/  ]
     SETTING UP CONFIG 6:QUANT/:3(16)
          {Here all the words which can start quantifiers, like "a
          hundred" or "point five", are proposed. The grammar
          does not preclude a quantifier following a
          question-determiner, e.g.  "What three men traveled to
          Spain?".}
    {MONITORING [ INTEGER ZERO NO POINT A]
          For considerations of space, long listings of monitors
          and proposals in this example will be compacted as shown
          here. Such alterations to the actual trace produced
          will be surrounded by brackets.}
 TRYING JUMP NP/QUANT ARC
  JUMP TRANS #5 FROM 5:NP/ORD:3(16) TO 7:NP/QUANT:3(21)


SELECTED CONFIGS (7) FOR EXTENSION
PICKING UP CONFIG 7:NP/QUANT:3(21)
 TRYING JUMP NP/DET ARC
  JUMP TRANS #6 FROM 7:NP/QUANT:3(21) TO 8:NP/DET:3(26)


SELECTED CONFIGS (8) FOR EXTENSION
PICKING UP CONFIG 8:NP/DET:3(26)
   STARTING AT 3:
    MONITORING [  NPR/ NPP/  ]
          {There are two PUSH NPR/ arcs from this state so two
          monitors are created, but only one configuration is set
          up.}
     SETTING UP CONFIG 9:NPR/:3(26)
    {MONITORING [NPR NPR N ADJ N V ADV]}
 TRYING JUMP NP/HEAD ARC
  JUMP TRANS #7 FROM 8:NP/DET:3(26) TO 10:NP/HEAD:3(29)


SELECTED CONFIGS (10) FOR EXTENSION
PICKING UP CONFIG 10:NP/HEAD:3(29)
          {This is an example of the fallibility of using only
          context free tests on partial paths. The parser thinks
          it has successfully reached state NP/HEAD, while in fact
          this cannot be the case because no head noun has been
          discovered for the noun phrase. Thus it is incorrect to
```

```
        predict relative clauses at this point.  This issue will
        be discussed in more detail below.}
  STARTING AT 3:
   MONITORING [  R/ PP/ R/NIL  ]
    SETTING UP CONFIG 11:R/:3(29)
   {MONITORING [PREP WHOSE WHO WHICH THAT WHOM]}
    {PROPOSING "WHOSE" "WHO" "WHICH" "THAT" "WHOM"}
    SETTING UP CONFIG 12:PP/:3(29)
   MONITORING [ PREP ]
    SETTING UP CONFIG 13:R/NIL:3(29)
   MONITORING [ THERE ]
    PROPOSING "THERE"
TRYING POP ARC
 POP TRANS #8 FROM 10:NP/HEAD:3(29)
   EXECUTING PATH (3 5 6 7 8)
    BEGINNING AT TRANS 3, CONFIG 3
   DOING CAT ARC WITH WHQ FROM 3:NP/ TO 5:NP/ORD
   DOING JUMP ARC FROM 5:NP/ORD TO 7:NP/QUANT
   DOING JUMP ARC FROM 7:NP/QUANT TO 8:NP/DET
   DOING JUMP ARC FROM 8:NP/DET TO 10:NP/HEAD
    TEST FAILED
        {A question-determiner alone cannot be a  complete  noun
        phrase;  although  this  is  permitted  by  considering
        "what" as a QWORD as in transition #2.}


STARTING AN ISLAND
"REGISTRATION" TRYING CAT N ARC FROM NP/DET TO NP/DET
  CAT N TRANS #9 FROM 14:NP/DET:6(1) TO 15:NP/DET:19(4)
        {This arc is using "registration" as a noun modifier for
        some future head noun.}
   ENDING AT 6:
    {MONITORING [  NPR/ ADJ N V  ]}
  JUMP TRANS #10 FROM 16:NP/QUANT:6(1) TO 14:NP/DET:6(1)
   ENDING AT 6:
    MONITORING [  QUANT/  ]
  JUMP TRANS #11 FROM 17:NP/ORD:6(1) TO 16:NP/QUANT:6(1)
   ENDING AT 6:
    {MONITORING [  ORD QDET ONLY  ]}
     PROPOSING "ONLY"
  JUMP TRANS #12 FROM 18:NP/ART:6(1) TO 17:NP/ORD:6(1)
   ENDING AT 6:
    {MONITORING [  ART QUANT POSS WHOSE  ]}
     NOTICING "THE"
     PROPOSING "WHOSE"
  JUMP TRANS #13 FROM 19:NP/ONLY:6(1) TO 18:NP/ART:6(1)
   ENDING AT 6:
    MONITORING [  ONLY  ]
     PROPOSING "ONLY"
  JUMP TRANS #14 FROM 20:NP/:6(1) TO 19:NP/ONLY:6(1)


 REGISTRATION" TRYING CAT N ARC FROM NP/DET TO NP/HEAD
  CAT N TRANS #15 FROM 14:NP/DET:6(1) TO 21:NP/HEAD:19(6)
        {This arc is using "registration" as the head noun of  a
        noun phrase.}
```

```
SELECTED CONFIGS (21) FOR EXTENSION
PICKING UP CONFIG 21:NP/HEAD:19(6)
   STARTING AT 19:
    MONITORING [  R/ PP/ R/NIL  ]
     SETTING UP CONFIG 22:R/:19(6)
    {MONITORING [PREP WHOSE WHO WHICH THAT WHOM]}
     SETTING UP CONFIG 23:PP/:19(6)
    MONITORING [ PREP ]
     SETTING UP CONFIG 24:R/NIL:19(6)
    MONITORING [ THERE ]
     NOTICING "FEE"
         {This notice is in response to the look-ahead test on
         the push arc to state R/NIL. Since "fee" can start a
         reduced relative clause, it is noticed, but there is not
         a specific monitor set up because the arc within the
         relative clause network which will actually process the
         word "fee" is not known.}
 TRYING POP ARC
  POP TRANS #16 FROM 21:NP/HEAD:19(6)
    EXECUTING PATH (14 13 12 11 10 15 16)
     BEGINNING AT TRANS 14, CONFIG 20
     TEST FAILED
         {The path failed because there is no determiner for
         "registration."}


PREDICTIONS:
NOTICING (4 FEE 19 23 155 0), SCORE -5
NOTICING (9 THE 4 6 103 0), SCORE 0
PROPOSING (ONLY WHOSE) ENDING AT 6
PROPOSING (ZERO NO POINT A WHOSE WHO WHICH THAT WHOM THERE)
 STARTING AT 3
{MONITORING [  PREP  ] STARTING AT 19, SCORE 0
MONITORING [  WHOSE WHO WHICH THAT WHOM THERE  ]
      STARTING AT 19, SCORE 5
MONITORING [  ADJ N V ORD QDET ART QUANT POSS  ]
      ENDING AT 6, SCORE 0
MONITORING [  WHOSE ONLY  ] ENDING AT 6, SCORE 5
MONITORING [  MODAL V INTEGER NPR N ADJ V ADV PREP  ]
      STARTING AT 3, SCORE 0
MONITORING [  WHOSE WHO WHICH THAT WHOM THERE ZERO NO POINT A  ]
      STARTING AT 3, SCORE 5}
PROPOSING (V N ADJ) FROM 3 TO 6
         {Proposals were made to fill the gap because there were
         monitors from both sides of a gap small enough to
         contain one word.}

FINISHED THEORY 1 WITH SYN SCORE 0

         {It took 11.9 seconds to process this theory.}


         {Processing the proposals just made results, notably, in
         the detection of the word "other" between "what" and
         "registration", but the word match score is very low.
         Word matches for "is" and "are" from position 3 (next to
         "what") to position 4 are also found, but since they do
```

not fill the gap, the event scores are low. The best event is that for the word "fee". Processing it is fairly uninteresting, since it completes no constituent, so we will omit the trace of that event. After it has been processed, however, the best event is that for the word "the" and the theory just created.}

```
SYNTAX PROCESSING EVENT FOR THEORY#2
  WITH NEW WORD (4 THE 6)
  TO GET NEW THEORY#3.
0 WHAT 3   4 THE 6 REGISTRATION 19 FEE 23


"THE" TRYING (CAT ART --) FROM STATE NP/ONLY TO CONFIG 18
  CAT ART TRANS #26 FROM 32:NP/ONLY:4(3) TO 18:NP/ART:6(6)
   ENDING AT 4:
    MONITORING [  ONLY  ]
     PROPOSING "ONLY"
  JUMP TRANS #27 FROM 33:NP/:4(1) TO 32:NP/ONLY:4(3)
    EXECUTING PATH (27 26 12 11 10 9 22 25)
     BEGINNING AT TRANS 22, CONFIG 15
****
    MADE #1 FROM 4 TO 23:
    NP DET ART THE
       ADJ NP N REGISTRATION
              NU SC
       N FEE
       FEATS NU.SG
****
```

{The format of this noun phrase is slightly different from that in the previous example because the structure building action for noun phrases in SPEECHGRAMMAR is different from that in MINIGRAMMAR.

There are many places in the SPEECHGRAMMAR which push for noun phrases, and since there were no monitors in the WFST which can use this constituent, all of them must be tried, resulting in a number of predictions and notices.}

```
SYN WEIGHT + SEM WT = 15 + 0 = 15


NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #28 FROM 34:FOR/FOR:4(1) TO 35:TO/:23(10)
   ENDING AT 4:
    MONITORING [  FOR  ]
     PROPOSING "FOR"

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #29 FROM 36:PP/PREP:4(1) TO 37:PP/NP:23(10)
   ENDING AT 4:
    MONITORING [  PREP  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #30 FROM 38:R/NIL:4(1) TO 39:S/NP:23(9)

NP/ WAS NEVER PUSHED FOR
```

```
  PUSH NP/ TRANS #31 FROM 40:R/WH:4(1) TO 39:S/NP:23(9)
   ENDING AT 4:
    MONITORING [  R/WHOSE  ]
    MONITORING [  WHICH THAT WHO WHOM WHICH WHOM  ]
     {PROPOSING "WHICH" "THAT" "WHO" "WHOM" "WHICH" "WHOM"}
         {There are two arcs entering state R/WH which use the
         words "which" and "whom". There is a check made to see
         that duplicate proposals are not actually communicated
         to the control component, although they appear to be
         duplicated in the trace.}

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #32 FROM 41:S/DCL:4(1) TO 39:S/NP:23(9)
  JUMP TRANS #33 FROM 42:S/:4(1) TO 41:S/DCL:4(1)
   ENDING AT 4:
    MONITORING [  PP/  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #34 FROM 43:S/NO-SUBJ:4(1) TO
   44:VP/V:23(9)
  JUMP TRANS #35 FROM 45:S/AUX:4(1) TO 43:S/NO-SUBJ:4(1)
   ENDING AT 4:
    {MONITORING [  MODAL NEG V  ]}
     {NOTICING "IS" "ARE"}

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #36 FROM 46:S/Q:4(1) TO 39:S/NP:23(9)
   ENDING AT 4:
    MONITORING [  QADV  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #37 FROM 47:VP/HEAD:4(1) TO 48:VP/NP:23(9)
   ENDING AT 4:
    MONITORING [  PARTICLE  ]
    MONITORING [  V  ]
  JUMP TRANS #38 FROM 49:VP/V:4(1) TO 47:VP/HEAD:4(1)
   ENDING AT 4:
    {MONITORING [  NP/ NP/ V V ADV V  ]}
     {NOTICING "IS" "ARE" "IS" "ARE"}
         {The words "is" and "are" failed the context free test
         on the arc causing the last monitor, hence they are not
         noticed.}
  JUMP TRANS #39 FROM 45:S/AUX:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #40 FROM 43:S/NO-SUBJ:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #41 FROM 43:S/NO-SUBJ:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #42 FROM 50:S/THERE:4(1) TO 49:VP/V:4(1)
   ENDING AT 4:
    MONITORING [  THERE  ]
     PROPOSING "THERE"

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #43 FROM 51:VP/NP:4(1) TO 52:VP/VP:23(9)
   ENDING AT 4:
    MONITORING [  NP/  ]
  JUMP TRANS #44 FROM 47:VP/HEAD:4(1) TO 51:VP/NP:4(1)
NP/ WAS NEVER PUSHED FOR
```

```
  PUSH NP/ TRANS #45 FROM 49:VP/V:4(1) TO 44:VP/V:23(9)

          {The creation of transition #27 completed 3 paths.  The
          first  two have been executed, resulting respectively in
          failure and the completion of a constituent with all the
          processing  that  entails.  Now the third path is still
          pending and is about to be executed.}
     EXECUTING PATH (27 26 12 11 10 15 16)
      BEGINNING AT TRANS 15, CONFIG 14
     DOING CAT ARC WITH REGISTRATION FROM 14:NP/DET TO
    21:NP/HEAD
     DOING POP ARC FROM 21:NP/HEAD
****
     MADE #2 FROM 4 TO 19:
     NP DET ART THE
        N REGISTRATION
        FEATS NU SG
****
          {This constituent can now satisfy the  monitors  set  by
          the  discovery  of  the  larger  one,  resulting  in the
          creation  of   many   new   transitions   but   no   new
          predictions.}
SYN WEIGHT + SEM WT = 10 + 0 = 10

NP/ WAS PUSHED FOR AT CONFIG 34
  PUSH NP/ TRANS #46 FROM 34:FOR/FOR:4(1) TO 53:TO/:19(8)
          {Similar NP/ transitions are set  up  at  configurations
          36,38,40,41,43,46,47,49,  and 51 because of the monitors
          set when the first constituent was found.}

SELECTED CONFIGS (55 54 39 37) FOR EXTENSION
          {Because these are the maximally scoring  configurations
          from the large pool of possibilities.}
PICKING UP CONFIG 55:S/NP:19(8) WITH WORD FEE
 TRYING POP ARC
  POP TRANS #56 FROM 55:S/NP:19(8)
    EXECUTING PATH (48 56)
     BEGINNING AT TRANS 48, CONFIG 38
     TEST FAILED
    EXECUTING PATH (33 50 56)
     BEGINNING AT TRANS 33, CONFIG 42
****
     MADE #3 FROM 4 TO 19:
     S NPU
       NP DET ART THE
          ADJ NP N REGISTRATION
                   NU SG
          N FEE
          FEATS NU SG
     WITH FEATURES (NPU)
****
          {Here is an example of a constituent which has  features
          attached  to  it.  The feature NPU can be tested by the
          semantic component to determine that the constituent  is
          a noun phrase utterance.  If necessary, it could also be
          tested on a PUSH S/ arc in the grammar, since there  are
          some   times,   e.g.   during  the  construction  of  a
```

```
          sentential complement, when an embedded sentence must
          contain a verb.}
SYN WEIGHT + SEM WT = 10 + 0 = 10
          {No arcs in this grammar push for noun phrase
          utterances, so this constituent is not used further.}

PICKING UP CONFIG 54:PP/NP:19(8) WITH WORD FEE
 TRYING POP ARC
  POP TRANS #57 FROM 54:PP/NP:19(8)

PICKING UP CONFIG 39:S/NP:23(9)
 TRYING POP ARC
  POP TRANS #58 FROM 39:S/NP:23(9)
    EXECUTING PATH (30 58)
     BEGINNING AT TRANS 30, CONFIG 38
     TEST FAILED
    EXECUTING PATH (33 32 58)
     BEGINNING AT TRANS 32, CONFIG 41
****
    MADE #4 FROM 4 TO 23:
    S NPU
      NP ADJ NP N REGISTRATION
                 NU SG
         DET ART THE
         N FEE
         FEATS NU SG
    WITH FEATURES (NPU)
****
SYN WEIGHT + SEM WT = 15 + 0 = 15

PICKING UP CONFIG 37:PP/NP:23(10)
 TRYING POP ARC
  POP TRANS #59 FROM 37:PP/NP:23(10)


PREDICTIONS:
NOTICING (19 IS 3 4 -79 0), SCORE 10
NOTICING (20 ARE 3 4 -128 0), SCORE 10
PROPOSING (ONLY FOR WHICH THAT WHO WHOM THERE) ENDING AT 4
{MONITORING [  ONLY FOR WHICH THAT WHO WHOM THERE  ]
      ENDING AT 4, SCORE 15
MONITORING [  MODAL NEG V QADV PARTICLE V V V ADV PREP V  ]
      ENDING AT 4, SCORE 10
MONITORING [  MODAL V INTEGER NPR N ADJ V ADV PREP  ]
      STARTING AT 3, SCORE 0
MONITORING [  ZERO NO POINT A WHOSE WHO WHICH THAT WHOM THERE]
     STARTING At 3, SCORE 5}
PROPOSING (MODAL) FROM 3 TO 4
PROPOSING (MODAL PREP) STARTING AT 3
PROPOSING (PREP MODAL NEG QADV) ENDING AT 4

CREATING THEORY 3:
0 WHAT 3   4 THE 6 REGISTRATION 19 FEE 23
 WITH SYN SCORE 15

          {This event took 34.5 seconds, largely because of the
          extensive bottom up processing necessitated by the
```

discovery of the noun phrases which were not monitored for.}

{Processing the proposals from this theory results in the best event being the one for "is" in the last gap. The word "are" also fills the gap, but the lower lexical score prevents the event for it from surfacing. If it were syntactically processed, however, no new theory would be created since the completed string would be ungrammatical.}

```
SYNTAX PROCESSING EVENT FOR THEORY#3
  WITH NEW WORD (3 IS 4)
  TO GET NEW THEORY#4:
0 WHAT 3 IS 4 THE 6 REGISTRATION 19 FEE 23

"IS" TRYING (CAT V --) FROM CONFIG 4
  CAT V TRANS #60 FROM 4:S/NP:3(11) TO 45:S/AUX:4(16)
```

{This transition does not immediately complete any paths, so the best scoring configurations of the theory are tried.}

```
SELECTED CONFIGS (31) FOR EXTENSION
PICKING UP CONFIG 31:NP/DET:23(36)
 TRYING JUMP NP/HEAD ARC
  JUMP TRANS #61 FROM 31:NP/DET:23(36) TO
   30:NP/HEAD:23(39)


SELECTED CONFIGS (52 48 44) FOR EXTENSION
PICKING UP CONFIG 52:VP/VP:23(9)
 TRYING JUMP S/VP ARC
  JUMP TRANS #62 FROM 52:VP/VP:23(9) TO 59:S/VP:23(12)
PICKING UP CONFIG 48:VP/NP:23(9)
 TRYING JUMP VP/VP ARC
  JUMP TRANS #63 FROM 48:VP/NP:23(9) TO 52:VP/VP:23(11)
PICKING UP CONFIG 44:VP/V:23(9)
 TRYING JUMP VP/HEAD ARC
  JUMP TRANS #64 FROM 44:VP/V:23(9) TO 60:VP/HEAD:23(13)


SELECTED CONFIGS (60) FOR EXTENSION
PICKING UP CONFIG 60:VP/HEAD:23(13)
 TRYING JUMP VP/NP ARC
  JUMP TRANS #65 FROM 60:VP/HEAD:23(13) TO 48:VP/NP:23(16)


SELECTED CONFIGS (59) FOR EXTENSION
PICKING UP CONFIG 59:S/VP:23(12)
 TRYING JUMP S/S ARC
  JUMP TRANS #66 FROM 59:S/VP:23(12) TO 61:S/S:23(14)

SELECTED CONFIGS (61) FOR EXTENSION
PICKING UP CONFIG 61:S/S:23(14)
```

```
 TRYING POP ARC
  POP TRANS #67 FROM 61:S/S:23(14)
    EXECUTING PATH (1 2 60 35 34 64 65 63 62 66 67)
     BEGINNING AT TRANS 34, CONFIG 43
****
    MADE #5 FROM 0 TO 23:.
    S Q
      SUBJ NP DET ART THE
              ADJ NP N REGISTRATION
                      NU SG
              N FEE
              FEATS NU SG
      AUX TNS PRESENT
          VOICE ACTIVE
      VP V BE
         OBJ NP N WHAT
                 FEATS NU SG/PL
****
          {This is the complete parse of the utterance, but
          SPARSER continues the operations it has pending before
          returning to Control.}
   NO SEMANTICS FOR HEAD
          {This is a comment from the semantic component
          indicating that it cannot currently interpret the
          construction.}
SYN WEIGHT + SEM WT = 25 + 0 = 25

S/ WAS NEVER PUSHED FOR
  PUSH S/ TRANS #68 FROM 62:COMPL/NTYPE:0(1) TO
   63:COMPL/S:23(15)

S/ WAS NEVER PUSHED FOR
  PUSH S/ TRANS #69 FROM 64:S/THEN:0(1) TO
   65:S/IFTHEN:23(15)

S/ WAS NEVER PUSHED FOR
  PUSH S/ TRANS #70 FROM 66:VP/HEAD:0(1) TO
   52:VP/VP:23(13)
  JUMP TRANS #71 FROM 67:VP/V:0(1) TO 66:VP/HEAD:0(1)
  JUMP TRANS #72 FROM 68:S/AUX:0(1) TO 67:VP/V:0(1)
  JUMP TRANS #73 FROM 69:S/NO-SUBJ:0(1) TO 67:VP/V:0(1)
  JUMP TRANS #74 FROM 68:S/AUX:0(1) TO 69:S/NO-SUBJ:0(1)
  JUMP TRANS #75 FROM 69:S/NO-SUBJ:0(1) TO 67:VP/V:0(1)
  JUMP TRANS #76 FROM 70:S/THERE:0(1) TO 67:VP/V:0(1)


          {One of the pending operations is to check the other
          arcs which caused monitors for the verb "is".}
"IS" TRYING (CAT V --) FROM STATE S/NP TO CONFIG 45


"IS" TRYING (CAT V --) FROM STATE FOR/TO TO CONFIG 49
  CAT V TRANS #77 FROM 71:FOR/TO:3(5) TO 49:VP/V:4(20)

"IS" TRYING (CAT V --) FROM STATE VP/V TO CONFIG 49
  CAT V TRANS #78 FROM 72:VP/V:3(5) TO 49:VP/V:4(20)
  JUMP TRANS #79 FROM 73:S/AUX:3(1) TO 72:VP/V:3(5)
```

```
JUMP TRANS #80 FROM 74:S/NO-SUBJ:3(1) TO 72:VP/V:3(5)
JUMP TRANS #81 FROM 73:S/AUX:3(1) TO 74:S/NO-SUBJ:3(1)
JUMP TRANS #82 FROM 74:S/NO-SUBJ:3(1) TO 72:VP/V:3(5)
JUMP TRANS #83 FROM 75:S/THERE:3(1) TO 72:VP/V:3(5)

CREATING THEORY 4:
0 WHAT 3 IS 4 THE 6 REGISTRATION 19 FEE 23
 WITH SYN SCORE 15
```

{This processing took 34.45 seconds.}

This example was run with a very simple, mechanical control structure. After the processing of the initial theory, the proposals which had been made by SPARSER were processed by the lexical retrieval component and the results added to the word lattice -- a process which can set off monitors and result in the creation of event notices. The events are scored by a combination of the monitor score assigned by SPARSER and the lexical score assigned by the word match component. In this sentence, syntax and lexical score alone were sufficient to make the best scoring event at each step be one which resulted in a correct extension of the theory.

EXAMPLE 3

We now show how the same utterance used in the previous example can be recognized when different theories are created and when events and theories are processed in a different order from that in Example 2. Suppose that after the initial scan of the utterance the semantic component created two theories, one for the words "what" and "fee" and the other for the words "what" and "registration" Let us see what happens in SPARSER when we begin by processing these two theories in sequence.

```
SPARSER PROCESSING THEORY 1:
O WHAT 3   19 FEE 23
          {The processing of this theory is very similar to that
          of the first theory in the previous example, and will
          not be commented upon here. The purpose in showing it
          is to provide a map, part of which the next call to
          SPARSER will trace.}

STARTING AN ISLAND
STARTING AT LEFT END OF SENTENCE

SELECTED CONFIGS (1) FOR EXTENSION
PICKING UP CONFIG 1:S/:O(1) WITH WORD WHAT
 TRYING PUSH PP/ ARC
 TRYING JUMP S/Q ARC
  JUMP TRANS #1 FROM 1:S/:O(1) TO 2:S/Q:O(6)
 TRYING WRD IF ARC
 TRYING JUMP S/IMP ARC
 TRYING JUMP S/DCL ARC

SELECTED CONFIGS (2) FOR EXTENSION
PICKING UP CONFIG 2:S/Q:O(6) WITH WORD WHAT
 TRYING PUSH NP/ ARC
      MONITORING  [ NP/ ]
         SETTING UP CONFIG 3:NP/:O(11)
 TRYING WRD HOW ARC
 TRYING CAT QWORD ARC
  CAT QWORD TRANS #2 FROM 2:S/Q:O(6) TO 4:S/NP:3(11)
 TRYING CAT QADV ARC
 TRYING JUMP S/NP ARC

SELECTED CONFIGS (3 4) FOR EXTENSION

PICKING UP CONFIG 3:NP/:O(11) WITH WORD WHAT
 TRYING WRD ONLY ARC
 TRYING CAT QDET ARC
  CAT QDET TRANS #3 FROM 3:NP/:O(11) TO 5:NP/ORD:3(16)
 TRYING PUSH DATE/ ARC
 TRYING TST ARC
 TRYING JUMP NP/ONLY ARC

PICKING UP CONFIG 4:S/NP:3(11)
   STARTING AT 3:
    MONITORING [  MODAL  ]
    MONITORING [  V  ]
 TRYING POP ARC
  POP TRANS #4 FROM 4:S/NP:3(11)
    EXECUTING PATH (1 2 4)
     BEGINNING AT TRANS 1, CONFIG 1
    DOING JUMP ARC WITH WHAT FROM 1:S/ TO 2:S/Q
    DOING CAT ARC WITH WHAT FROM 2:S/Q TO 4:S/NP
    DOING POP ARC FROM 4:S/NP
     TEST FAILED

SELECTED CONFIGS (5) FOR EXTENSION
PICKING UP CONFIG 5:NP/ORD:3(16)
   STARTING AT 3:
```

```
    MONITORING [  QUANT/  ]
     SETTING UP CONFIG 6:QUANT/:3(16)
    MONITORING [ INTEGER ]
 TRYING JUMP NP/QUANT ARC
  JUMP TRANS #5 FROM 5:NP/ORD:3(16) TO 7:NP/QUANT:3(21)

SELECTED CONFIGS (7) FOR EXTENSION
PICKING UP CONFIG 7:NP/QUANT:3(21)
 TRYING JUMP NP/DET ARC
  JUMP TRANS #6 FROM 7:NP/QUANT:3(21) TO 8:NP/DET:3(26)

SELECTED CONFIGS (8) FOR EXTENSION
PICKING UP CONFIG 8:NP/DET:3(26)
   STARTING AT 3:
    {MONITORING [  NPR/ NPR/ NPR NPR N ADJ N V ADV  ]}
     SETTING UP CONFIG 9:NPR/:3(26)
 TRYING JUMP NP/HEAD ARC
  JUMP TRANS #7 FROM 8:NP/DET:3(26) TO 10:NP/HEAD:3(29)

SELECTED CONFIGS (10) FOR EXTENSION
PICKING UP CONFIG 10:NP/HEAD:3(29)
   STARTING AT 3:
    {MONITORING [ R/ PP/ R/NIL PREP WHOSE WHO WHICH THAT WHOM ]}
     SETTING UP CONFIG 11:R/:3(29)
     {PROPOSING "WHOSE" "WHO" "WHICH" "THAT" "WHOM"}
     SETTING UP CONFIG 12:PP/:3(29)
    MONITORING [ PREP ]
     SETTING UP CONFIG 13:R/NIL:3(29)
    MONITORING [ THERE ]
     PROPOSING "THERE"
 TRYING POP ARC
  POP TRANS #8 FROM 10:NP/HEAD:3(29)
    EXECUTING PATH (3 5 6 7 8)
     BEGINNING AT TRANS 3, CONFIG 3
    DOING CAT ARC WITH WHQ FROM 3:NP/ TO 5:NP/ORD
    DOING JUMP ARC FROM 5:NP/ORD TO 7:NP/QUANT
    DOING JUMP ARC FROM 7:NP/QUANT TO 8:NP/DET
    DOING JUMP ARC FROM 8:NP/DET TO 10:NP/HEAD
     TEST FAILED


STARTING AN ISLAND
"FEE" TRYING CAT N ARC FROM NP/DET TO NP/DET
  CAT N TRANS #9 FROM 14:NP/DET:19(1) TO 15:NP/DET:23(4)
   ENDING AT 19:
    MONITORING [  NPR/  ]
    MONITORING [  ADJ  ]
    MONITORING [  N  ]
     NOTICING "REGISTRATION"
     NOTICING "REGISTRATION"
         {There are two instances of the word  "registration"  in
         the word lattice, hence two notices are created.}
    MONITORING [  V  ]
  JUMP TRANS #10 FROM 16:NP/QUANT:19(1) TO 14:NP/DET:19(1)
   ENDING AT 19:
    MONITORING [  QUANT/  ]
  JUMP TRANS #11 FROM 17:NP/ORD:19(1) TO 16:NP/QUANT:19(1)
```

```
   ENDING AT 19:
    {MONITORING [  ORD QDET ONLY  ]}
     PROPOSING "ONLY"
  JUMP TRANS #12 FROM 18:NP/ART:19(1) TO 17:NP/ORD:19(1)
   ENDING AT 19:
    {MONITORING [  ART QUANT POSS WHOSE  ]}
     PROPOSING "WHOSE"
  JUMP TRANS #13 FROM 19:NP/ONLY:19(1) TO 18:NP/ART:19(1)
   ENDING AT 19:
    MONITORING [  ONLY  ]
     PROPOSING "ONLY"
  JUMP TRANS #14 FROM 20:NP/:19(1) TO 19:NP/ONLY:19(1)

"FEE" TRYING CAT N ARC FROM NP/DET TO NP/HEAD
  CAT N TRANS #15 FROM 14:NP/DET:19(1) TO 21:NP/HEAD:23(6)

SELECTED CONFIGS (21) FOR EXTENSION
PICKING UP CONFIG 21:NP/HEAD:23(6)
 TRYING POP ARC
  POP TRANS #16 FROM 21:NP/HEAD:23(6)
    EXECUTING PATH (14 13 12 11 10 15 16)
     BEGINNING AT TRANS 14, CONFIG 20
     TEST FAILED


PREDICTIONS:
NOTICING (1 REGISTRATION 6 19 277 0), SCORE -5
NOTICING (10 REGISTRATION 7 19 103 0), SCORE -5
PROPOSING (ONLY WHOSE) ENDING AT 19
PROPOSING (WHOSE WHO WHICH THAT WHOM THERE) STARTING AT 3
{MONITORING [  ADJ N V ORD QDET ART QUANT POSS  ]
      ENDING AT 19, SCORE 0
MONITORING [  WHOSE ONLY  ] ENDING AT 19, SCORE 5
MONITORING [  MODAL V INTEGER NPR N ADJ V. ADV PREP  ]
      STARTING AT 3, SCORE 0
MONITORING [  WHOSE WHO WHICH THAT WHOM THERE  ]
      STARTING AT 3, SCORE 5}

FINISHED THEORY 1 WITH SYN SCORE 0

         {This processing took 12.5 seconds.}


         {Now we will process the second theory.}
SPARSER PROCESSING THEORY 2:
0 WHAT 3   6 REGISTRATION 19  23

STARTING AN ISLAND
STARTING AT LEFT END OF SENTENCE

SELECTED CONFIGS (1) FOR EXTENSION
         {Upon picking up this configuration to extend it,
         SPARSER finds the transitions which were created during
         the processing of the word "what" by the previous
         theory.  It "traces" them all, that is, it does not
         recreate them but simply puts the transition numbers on
         a list which will form part of the syntactic information
```

associated with the current theory. The tracing process also involves the creation of monitors (and notices, where applicable) for constituents along the path. These monitors and notices must be remade, since the previous monitors will activate only the previous theory.

Due to the recursive nature of the tracing process, the transitions are not necessarily followed in the same order that they were originally created, nor are the monitors made in exactly the same order.

Notice that the many arcs which were tried but which did not result in the creation of transitions in the previous theory are not retried here.}

```
PICKING UP CONFIG 1:S/:0(1) WITH WORD WHAT
  TRACING JUMP S/Q TRANS 1 FROM 1:S/:0(1) TO 2:S/Q:0(6)
      MONITORING  [ NP/ ]
  TRACING CAT QWORD TRANS 2 USING "WHAT" FROM 2:S/Q:0(6)
   TO 4:S/NP:3(11)
   STARTING AT 3:
    MONITORING [  MODAL  ]
    MONITORING [  V  ]
  TRACING POP TRANS 4 FROM 4:S/NP:3(11)
  TRACING CAT QDET TRANS 3 USING "WHQ" FROM 3:NP/:0(11)
   TO 5:NP/ORD:3(16)
   STARTING AT 3:
    MONITORING [  QUANT/  ]
     SETTING UP CONFIG 6:QUANT/:3(16)
```

{This does not mean that configuration 6 was just created. Since it already existed in the map, having been created during the processing of the previous theory, the configuration number is merely put on the list of configurations in the current theory.}

```
  MONITORING [ INTEGER ]
TRACING JUMP NP/QUANT TRANS 5 FROM 5:NP/ORD:3(16) TO
 7:NP/QUANT:3(21)
TRACING JUMP NP/DET TRANS 6 FROM 7:NP/QUANT:3(21) TO
 8:NP/DET:3(26)
 STARTING AT 3:
  {MONITORING [  NPR/ NPR/ NPR NPR N ADJ N V ADV  ]}
   SETTING UP CONFIG 9:NPR/:3(26)
TRACING JUMP NP/HEAD TRANS 7 FROM 8:NP/DET:3(26) TO
 10:NP/HEAD:3(29)
 STARTING AT 3:
  {MONITORING [  R/ PP/ R/NIL. PREP WHOSE WHO WHICH
                 THAT WHOM PREP THERE  ]}
   SETTING UP CONFIG 11:R/:3(29)
   {NOTICING "WHOSE" "WHO" }
   SETTING UP CONFIG 12:PP/:3(29)
   SETTING UP CONFIG 13:R/NIL:3(29)
```

{No proposals were made here because proposals are not theory dependent; that is, the word proposals which were made during the processing of the previous theory resulted in some words being placed in the word lattice which were noticed here. Remaking the proposals would not lead to the discovery of any new information.}

```
TRACING POP TRANS 8 FROM 10:NP/HEAD:3(29)
```

```
        {The processing of the island for "registration" is
        identical to that in the last example, so the remainder
        of the trace will be omitted. The total processing took
        12.2 seconds.}


        {Let us now process the event which adds the word "the"
        to the theory just processed. This will result in the
        creation of a constituent event.}
SYNTAX PROCESSING EVENT FOR THEORY#2
  WITH NEW WORD (4 THE 6)
  TO GET NEW THEORY#3:
0 WHAT 3   4 THE 6 REGISTRATION 19  23

"THE" TRYING (CAT ART --) FROM STATE NP/ONLY TO CONFIG 25
  CAT ART TRANS #25 FROM 32:NP/ONLY:4(3) TO 25:NP/ART:6(6)
   ENDING AT 4:
    MONITORING [  ONLY  ]
     PROPOSING "ONLY"
  JUMP TRANS #26 FROM 33:NP/:4(1) TO 32:NP/ONLY:4(3)
    EXECUTING PATH (26 25 20 19 18 17 15 16)
     BEGINNING AT TRANS 26, CONFIG 33
****
    MADE #1 FROM 4 TO 23:
    NP DET ART THE
       ADJ NP N REGISTRATION
             NU SG
      N FEE
      FEATS NU SG
****
         NOTIFYING THEORY 3 ABOUT CONSTITUENT #1
        {This constituent cannot be used immediately by this
        theory because it contains a word ("fee") which is not
        in the theory. Therefore a notice is sent to Control
        which may be turned into an event at some later time.
        Nothing further is done with this constituent at this
        time, i.e., no transitions using it are created. It is,
        however, placed in the WFST for later use.}

    EXECUTING PATH (26 25 20 19 18 23 24)
     BEGINNING AT TRANS 23, CONFIG 22
        {The creation of transition #26 completes another path.}
****
    MADE #2 FROM 4 TO 19:
    NP DET ART THE
       N REGISTRATION
       FEATS NU SG
****
SYN WEIGHT + SEM WT = 10 + 0 = 10

        {This constituent is completely consistent with the
        current theory, that is, it is composed only of word
        matches already in the theory, and there are no monitors
        in the WFST for it, so it is processed bottom up as we
        have seen before.}
NP/ WAS NEVER PUSHED FOR
```

```
  PUSH NP/ TRANS #27 FROM 34:FOR/FOR:4(1) TO 35:TO/:19(7)
   ENDING AT 4:
    MONITORING [  FOR  ]
     PROPOSING "FOR"

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #28 FROM 36:PP/PREP:4(1) TO
   37:PP/NP:19(7)
   ENDING AT 4:
    MONITORING [  PREP  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #29 FROM 38:R/NIL:4(1) TO 39:S/NP:19(6)

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #30 FROM 40:R/WH:4(1) TO 39:S/NP:19(6)
   ENDING AT 4:
    MONITORING [  R/WHOSE  ]
    MONITORING [  WHICH THAT WHO WHOM WHICH WHOM  ]
     {NOTICING "WHO" "WHOM" "WHICH" "WHOM"}

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #31 FROM 41:S/DCL:4(1) TO 39:S/NP:19(6)
  JUMP TRANS #32 FROM 42:S/:4(1) TO 41:S/DCL:4(1)
   ENDING AT 4:
    MONITORING [  PP/  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #33 FROM 43:S/NO-SUBJ:4(1) TO
   44:VP/V:19(6)
  JUMP TRANS #34 FROM 45:S/AUX:4(1) TO 43:S/NO-SUBJ:4(1)
   ENDING AT 4:
    {MONITORING [  MODAL NEG V  ]}
     {NOTICING "IS" "ARE" "PAY"}

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #35 FROM 46:S/Q:4(1) TO 39:S/NP:19(7)
   ENDING AT 4:
    MONITORING [  QADV  ]

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #36 FROM 47:VP/HEAD:4(1) TO
   48:VP/NP:19(7)
   ENDING AT 4:
    MONITORING [  PARTICLE  ]
    MONITORING [  V  ]
     NOTICING "PAY"
  JUMP TRANS #37 FROM 49:VP/V:4(1) TO 47:VP/HEAD:4(1)
   ENDING AT 4:
    {MONITORING [  NP/ NP/ V ADV V  ]}
     {NOTICING "IS" "ARE" "PAY" "IS" "ARE" "PAY"}
  JUMP TRANS #38 FROM 45:S/AUX:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #39 FROM 43:S/NO-SUBJ:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #40 FROM 43:S/NO-SUBJ:4(1) TO 49:VP/V:4(1)
  JUMP TRANS #41 FROM 50:S/THERE:4(1) TO 49:VP/V:4(1)
   ENDING AT 4:
    MONITORING [  THERE .]
```

```
     PROPOSING "THERE"

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #42 FROM 51:VP/NP:4(1) TO 52:VP/VP:19(6)
   ENDING AT 4:
    MONITORING [  NP/  ]
  JUMP TRANS #43 FROM 47:VP/HEAD:4(1) TO 51:VP/NP:4(1)

NP/ WAS NEVER PUSHED FOR
  PUSH NP/ TRANS #44 FROM 49:VP/V:4(1) TO 44:VP/V:19(7)

SELECTED CONFIGS (35 37 39 44 48) FOR EXTENSION
PICKING UP CONFIG 35:TO/:19(7)
   STARTING AT 19:
    MONITORING [  NEG  ]
    MONITORING [  TO  ]
     PROPOSING "TO"
 ALL ARCS TRIED AT THIS CONFIG

PICKING UP CONFIG 37:PP/NP:19(7)
 TRYING POP ARC
  POP TRANS #45 FROM 37:PP/NP:19(7)

PICKING UP CONFIG 39:S/NP:19(7)
   STARTING AT 19:
    MONITORING [  MODAL  ]
    MONITORING [  V  ]
 TRYING POP ARC
  POP TRANS #46 FROM 39:S/NP:19(7)
    EXECUTING PATH (32 31 46)
     BEGINNING AT TRANS 32, CONFIG 42
****
    MADE #3 FROM 4 TO 19:
    S NPU
      NP DET ART THE
         N REGISTRATION
         FEATS NU SG
    WITH FEATURES (NPU)
****
SYN WEIGHT + SEM WT = 10 + 0 = 10

PICKING UP CONFIG 44:VP/V:19(7)
   STARTING AT 19:
    {MONITORING [  NP/ N QDET ADJ INTEGER ARE QUANT
                   PRO NPR POSS V V ADV TEST(NOT (CAT V))  ]}
     SETTING UP CONFIG 20:NP/:19(7)
     NOTICING "FEE"
 ALL ARCS TRIED AT THIS CONFIG

PICKING UP CONFIG 48:VP/NP:19(7)
   STARTING AT 19:
    {MONITORING [  COMPL/ TO/ COMPL/ NP/ FOR THAT TO FOR THAT
                   N QDET ADJ INTEGER ARE QUANT PRO NPR POSS
                   V PARTICLE  ]}
     SETTING UP CONFIG 53:COMPL/:19(7)
     SETTING UP CONFIG 35:TO/:19(7)
    SETTING UP CONFIG 53:COMPL/:19(7)
```

```
     SETTING UP CONFIG 20:NP/:19(7)
     NOTICING "FEE"
 TRYING JUMP VP/VP ARC
  JUMP TRANS #47 FROM 48:VP/NP:19(7) TO 52:VP/VP:19(9)

SELECTED CONFIGS (52) FOR EXTENSION
PICKING UP CONFIG 52:VP/VP:19(9)
   STARTING AT 19:
    MONITORING [  PP/  ]
     SETTING UP CONFIG 30:PP/:19(9)
    MONITORING [ PREP ]
    MONITORING [  PREP  ]
 TRYING JUMP S/VP ARC
  JUMP TRANS #48 FROM 52:VP/VP:19(9) TO 54:S/VP:19(12)

SELECTED CONFIGS (54) FOR EXTENSION
PICKING UP CONFIG 54:S/VP:19(12)
 TRYING JUMP S/S ARC
  JUMP TRANS #49 FROM 54:S/VP:19(12) TO 55:S/S:19(14)

SELECTED CONFIGS (55) FOR EXTENSION
PICKING UP CONFIG 55:S/S:19(14)
 TRYING POP ARC
  POP TRANS #50 FROM 55:S/S:19(14)

PREDICTIONS:
NOTICING (4 FEE 19 23 155 0), SCORE 5
NOTICING (19 WHO 3 4 -180 0), SCORE 10
NOTICING (21 IS 3 4 -39 0), SCORE 10
NOTICING (23 ARE 3 4 -128 0), SCORE 10
NOTICING (25 PAY 3 4 -146 0), SCORE 5
PROPOSING (TO) STARTING AT 19
PROPOSING (ONLY FOR WHOM WHICH THERE) ENDING AT 4
PROPOSING (V MODAL) FROM 3 TO 4
PROPOSING (MODAL PREP) STARTING AT 3
PROPOSING (PREP MODAL NEG QADV) ENDING AT 4
        {The lengthy summary of monitors set by  this  event  is
        omitted.}

CREATING THEORY 3:
0 WHAT 3   4 THE 6 REGISTRATION 19  23
 WITH SYN SCORE 15

        {This took 30.8 seconds.}


        {Now we will  process  the  constituent  event  for  the
        theory  just  created.  Because  of the constituent for
        "the registration" there are now monitors  in  the  WFST
        for  a  noun  phrase  beginning  at  position  4, so the
        appropriate transitions are made.}

SYNTAX PROCESSING EVENT FOR THEORY#3 WITH CONSTITUENT #1
  TO GET NEW THEORY#4
0 WHAT 3   4 THE 6 REGISTRATION 19 FEE 23

        {Processing begins exactly where it left  off  when  the
```

```
          constituent  was made -- the constituent is semantically
          evaluated with  respect  to  this  theory  so  that  the
          constituent  weight  may  be  altered.   In  this  case,
          however, Semantics has been turned off, so there  is  no
          increment in the score.}
SYN WEIGHT + SEM WT = 15 + 0 = 15

NP/ WAS PUSHED FOR AT CONFIG 34
  PUSH NP/ TRANS #51 FROM 34:FOR/FOR:4(1) TO 56:TO/:23(11)
          {Similar  transitions  are  set  up  for  all  9   other
          configurations  where  an  NP/  was used in the previous
          theory.  The monitors set by these paths are copied from
          the previous theory, so there is no indication here of a
          new monitor being created.}

SELECTED CONFIGS (58 57) FOR EXTENSION
PICKING UP CONFIG 58:S/NP:23(10)
 TRYING POP ARC
  POP TRANS #61 FROM 58:S/NP:23(10)
    EXECUTING PATH (32 55 61)
     BEGINNING AT TRANS 55, CONFIG 41
    DOING PUSH ARC WITH #1 FROM 41:S/DCL TO 58:S/NP
    DOING POP ARC FROM 58:S/NP
****
    MADE #4 FROM 4 TO 23:
    S NPU
      NP DET ART THE
         ADJ NP N REGISTRATION
                 NU SG
         N FEE
         FEATS NU SG
    WITH FEATURES (NPU)
****
SYN WEIGHT + SEM WT = 15 + 0 = 15

PICKING UP CONFIG 57:PP/NP:23(11)
 TRYING POP ARC
  POP TRANS #62 FROM 57:PP/NP:23(11)

PREDICTIONS:
NOTICING (19 WHO 3 4 -180 0), SCORE 10
NOTICING (21 IS 3 4 -39 0), SCORE 10
NOTICING (23 ARE 3 4 -128 0), SCORE 10
NOTICING (25 PAY 3 4 -146 0), SCORE 5
PROPOSING (V MODAL) FROM 3 TO 4
PROPOSING (MODAL PREP) STARTING AT 3
PROPOSING (PREP MODAL NEG QADV) ENDING AT 4
          {Again, the monitor list is omitted  for  considerations
          of space.}

CREATING THEORY 4:
0 WHAT 3   4 THE 6 REGISTRATION 19 FEE 23
 WITH SYN SCORE 15

          {This event took only 9.7 seconds.}
```

The processing of the final event, that which adds the word "is" to the theory just created, will not be shown.

These examples have shown that SPARSER is a useful tool in the automatic recognition of speech. The timing measurements indicate that considerable processing is done when the parser is forced to work in bottom up mode, especially with a large grammar Of course there is some implementaion overhead involved in doing the timings themselves. If the parsing algorithm were to be carefully recoded in assembly language a speed up of at least a factor of 20 (and perhaps much more) could be achieved. Another way to cut down the time-consuming processing might be to attempt to obtain more semantic guidance. For example, if the semantic hypothesis associated with a theory indicates that a particular noun is likely to be used in a noun phrase modifier (e.g. "tomorrow"), then SPARSER should be able to take advantage of this information by scoring the PUSH NP/ transition from a configuration for state PP/ (i.e. to get something like "by tomorrow") higher than those PUSH NP/ transitions for other syntactic slots. In fact, the others may not need to be constructed at all. The grammar could also be further tuned to eliminate some spurious predictions and reduce the time spent following erroneous paths.

Section 7

Conclusions and Further Research

7.1 STRENGTHS AND WEAKNESSES OF SPARSER

One of the weak points of the current system is the fact that some context information is not used until a path is complete, resulting in the creation of false paths and predictions which should not have been made. This is partly mitigated by the fact that this avoids a too great dependence on left context and allows the creation of partial paths which may be followed if an earlier word is changed.

It is important, however, to minimize the number of predictions which are made and to make the predictions as accurate as possible. In this regard, it is unfortunate that the current system makes predictions on the left of an island solely on the basis of the first word in the island and makes predictions on the right end from configurations which, if context sensitive tests had been done along the path, would never have been created.

One way to help tighten the predictions would be to take each context free path through an island and walk it in a special mode after the island has been processed but before predictions are communicated to the control component. This mode would set and check registers, assuming that any tests which require unknown left context are true. Only if the path did not fail

under this mode of operation would the predictions at either end of it be made. If a really efficient way of handling unknown left context and of storing this information were developed, it could be used in place of the context free pass in the first place, thus eliminating all inconsistent paths.

The problem with storing all possible contexts is that they must be recomputed each time a new step is added to the path. This is relatively easy if the next step is taken to the right of an existing path, since ATN s are more suited to left to right processing, but it becomes extremely complex when a transition is added to the left end of a path (or set of paths) or when a transition joins two sets of paths together. To be absolutely sure that no contexts have been missed, all the paths would have to be walked and their contexts reprocessed and copied in whole or in part (since the new step may be wrong, the old context must be preserved.) Of course this is not the only approach which could be used -- a merging technique like that of Earley's algorithm might be feasible, if the structure of the grammar were also changed to make it less left to right oriented.

One great strength of the system is its ability to store and merge information in such a way that it does not have to be redone when the context is changed. For example, once an arc has been tried with a particular word match, a transition will be created if the arc may be taken and the arc will be removed from further consideration if it may not be taken. Then, if the configuration should ever be reached with the same word match again (perhaps in a later theory) not only will any relevant

transitions be recognized without having to go through the work of re-creating them, but also no arcs which had previously failed will ever be retried.

Another feature of SPARSER is the fact that it was designed and implemented with many unsolved problems and unavailable data in mind, and therefore many "holes" have been left on which to "hook" further developments. For example, although prosodic verification of constituents is not yet available, the scoring mechanism for constituents is structured in such a way that it would be easy to include the results of verification by prosodics (or any other component). The original implementation of SPARSER used a depth first search but was implemented in such a way that the change to modified breadth first was quite simple. This foresight has paid off in a flexible system which has shown that it can be readily experimented with in order to explore many still unsolved problems concerning the nature and use of syntactic information in understanding

7.2 PROSODICS

A tremendous amount of information in speech is conveyed by prosodic features: stress, intonation duration, loudness, pauses, pitch. For example, if John mumbles to Bill, "The mailman left something for you," Bill may reply either "What?" with much energy and a sharply rising intonation or "What?" with a flat or falling intonation. In the first case John is very likely to shout "I said, The mailman left something for you "

interpreting ."What?" to mean "What did you say?" whereas in the second case he is likely to say something like "A package from your mother," interpreting "What?" to mean "What is it?" To ignore prosodics is to ignore a source of information which has been shown repeatedly to be an extremely important factor in human understanding.

Consider the following examples of sentences and sentence fragments which illustrate some of the ways prosodies are used:

1. I stepped on the man with black shoes. (Who was wearing the shoes?)

2a. The new gnu knew news.

2b. The gnu knew new gnus.

3. I m going to move on Thursday. (stress on "move" implies moving to a new house; stress on "on" implies traveling to a new place.)

4a. Can you swim to Daddy?

4b. Can you swim too, Daddy?

5a. ... two-fifty for ...

5b. ... two-fifty-four ...

Prosodic verification could help a lot in rejecting semantically correct, syntactically consistent phrases which are nonetheless wrong. If the constituent "speech understanding" were identified and relied upon, it might be very difficult to produce a correct analysis of the utterance: "Because of peculiarities in his speech, understanding Joe is not easy."

Besides indicating syntactic boundaries and/or providing intonation contours for certain constituents, prosodic features can be used to mark emphasis, introduce new topics, convey information about the speaker s internal mental and emotional state (e.g. whether he is teasing or serious), and probably more. It is particularly interesting to note that some well known phenomena such as "pronouns are almost never stressed" and "in discourse when a new topic word is mentioned it is almost always stressed" have very natural explainations in light of what we know about acoustic processing. Stressed words are generally easier to identify because there is less acoustic ambiguity, but unstressed words may differ greatly from their ideal pronunciation and hence are harder to reliably identify Pronouns refer to antecedents which are presumably known to the listener, so he can anticipate them or at least verify them easily, hence they need not have good acoustic characteristics. A new topic may not have been anticipated, so the listener will have to depend heavily on identifying the word from acoustic information alone and the speaker can provide this extra reliable information by stressing the word.

Unfortunately, not a great deal is known about either the acoustic correlates of prosodic features or the ways in which they are used. Many of the rules which have been developed thus far are speaker dependent and are sufficient for conveying information but are not necessary. This makes them difficult to use in the analysis mode. Although a good start has been made in exploring prosodies (see, for example, Lea [52, 13] and Bates and Wolf [8]), much more work remains to be done before prosodic

information can be reliably used by speech understanding systems.

SPARSER could use prosodic information in several ways. Verification of constituents would be a great help, but local prosicid information could be used even earlier in the parsing process. For example, if major constituent boundaries could be accurately determined, then instead of both POPing a constituent and continuing it in parallel, as is done now, one alternative could be chosen instead of the other on the badis of prosodic information. If, as is more likely, some major boundaries could be reliably detected, then it would be easy to revise SPARSER to begin processing at such places even within an island at states which can begin constituents. This would again reduce the number of partial paths created when parsing an island.

7.3 EXTENSIONS AND FURTHER RESEARCH

Ungrammatical input

One of the obvious extensions to a basic speech understanding system is to relax the restrictions on the input to the system. Syntactically, this can mean removing the requirement that the initial utterance be grammatical. Since people frequently speak ungrammatically in informal discourse, this is a natural step to want to take.

In order to extend SPARSER to handle such input, several approaches are possible. Certain types of errors may be called errors of style (and may not be called errors at all by some

people) such as the use of "ain't" and the occurrence of a preposition at the end of a sentence. These regularities may simply be declared grammatical by modifying the grammar to accept them. Many speech errors have been shown to follow regular patterns and hence may be amenable to this approach.

Other common errors violate specific tests which appear on the arcs of the grammar, for example, to prohibit double negatives and to check for number agreement between subject and verb or between determiner and noun (e.g. *"There is some very severe restrictions on this rule."). In this case, rather than removing the tests from the grammar it would be more suitable to modify them so that if they failed the arc could still be taken, though with a much reduced weight or with an indication in some register that an error has occurred. One way to implement this would be to have all tests return a number as their value indicating how well they succeeded on some scale from "perfectly" to "not at all".

Not all arc tests are of this relaxable nature, however, since certain types of errors are so rare, if they occur at all, that they may be judged unacceptable. Examples of such tests are the case checks for pronouns (e.g. *"I gave it to he") and the requirement that a verb modifying a noun must be in either the present or past participle form (e.g. "the singing brook" vs. *"the sing brook").

These methods would not allow all types of grammatical errors to be handled (in particular it ignores the problem of constituent ordering errors such as "Throw Mama from the train a

kiss"), but would handle many of the most common syntactic errors.

An experiment

Keeping in mind that SPARSER is not intended to be a model of human syntactic analysis, it is nonetheless reasonable to ask whether there are any similarities which may be seen. The following experiment is suggested with the hypothesis that it will indicate that people do considerable processing at the end of syntactic constituents in a way similar to some register setting and testing actions and semantic (or other) verification

The experiment is this: a subject is seated in front of a switch which he is asked to press whenever he is sure that he is hearing an anomalous sentence, He is then presented with a number of recorded utterances, some of which are incorrect, e.g.

"The cat and dog which lives next door are friendly.

"I saw a red big barn on the farm.

I hypothesize that the subject will indicate the presence of an error at a point shortly after the end of the constituent in which the error occurred more often thand shortly after the earliest possible place where the error coul be detected.

7 4 CONCLUSION

In conclusion, it is obvious that there is much work yet to be done in the problem of speech understanding, but it is hoped that the system presented here has not only advanced our current

understanding of the role of syntactic knowledge in comprehension but will continue to be a useful tool for further exploration of the field.

APPENDIX I

MINIGRAMMAR

This appendix contains a listing (slightly edited for clarity) of the grammar called MINIGRAMMAR which was discussed in Section Three (illustrated in Figure 3.3) and which was used in Section Six.

```
(NP/
   (CAT ART (T T)
        5
        (SETR ART (BUILDQ ((ART *))))
        (TO NP/ART))
   (JUMP NP/ART (T T)
        4))

(NP/ADJ
   (CAT N (T T)
        5
        (SETR N *)
        (SETR NU (GETFEATURE NUMBER))
        (TO NP/N))
   (CAT N (T T)
        2
        (ADDL ADJS (BUILDQ (ADJ (NP (N *)
                                    (NU  )))
                           (GETFEATURE NUMBER)))
        (TO NP/ADJ)))

(NP/ART
   (CAT QUANT (T T)
        4
        (SETR QUANT (BUILDQ ((QUANT *))))
        (TO NP/QUANT))
   (JUMP NP/QUANT (T T)
        5))
```

```
(NP/QUANT
   (CAT ADJ (T T)
        4
        (ADDR ADJS (BUILDQ (@ (ADJ)
                              (*))
                           FEATURES))
        (TQ NP/QUANT))
   (JUMP NP/ADJ (T T)
        4))

(NP/N
   (PUSH PP/ ((PPSTART)
          T T)
        4
        (ADDL NMODS *)
        (TO NP/N))
   (POP (BUILDQ (@ (NP)
                   + + + ((N +))
                   ((NU +))
                   +)
                ART QUANT ADJS N NU NMODS)
        (T (DETAGREE))
        5))

(PP/
   (CAT PREP (T T)
        5
        (SETR PREP *)
        (TO PP/PREP)))

(PP/PREP
   (PUSH NP/ ((NPSTART)
          T T)  -
        5
        (SETR NP *)
        (TO PP/NP)))
                  -

(PP/NP
   (POP (BUILDQ (PP (PREP +)
                    +)
                PREP NP)
        (T T)
        5))
```

APPENDIX II

Vocabulary and Syntax Classes

This appendix lists the 351 words which were in the dictionary of the BBN speech understanding system when the examples in Chapter Six were run (July 1975). (A 569 word dictionary and one with 1000 entries are now available. After the listing of the words in the dictionary, they are broken into syntactic classes, with the number of words in each class indicated beside the class name. Finally, the syntactic features are given together with a list of the words which carry each feature. Features may be of the form FEATURE, (FEATURE), or (FEATURE VALUE).

This is not a listing of the dictionary as it appears to the system, but rather a derived cross reference which indicates the various parts of speech and syntactic features for each word

The words:

(A ABOUT ABOVE ACL ACOUSTICAL ACOUSTICS ADDITIONAL AFFORD AFTER AI AIR AIRPLANE ALL ALREADY ALSO AM AMHERST AMOUNT AN AND ANTICIPATE ANY ANYONE ANYWHERE APRIL ARE ARPA ARRANGE ASA ASK ASSOCIATION ASSUME ASSUMPTION AT ATTEND AUGUST AVAILABLE BATES BE BECAUSE BEEN BEFORE BEGINNING BEING BIG BILL BONNIE BOSTON BOTH BREAKDOWN BUDGET BUS BY CALIFORNIA CAN CANCEL CAR CARNEGIE CENT CHANGE CITY COLARUSSO COMPUTATIONAL CONFERENCE CONTINUE COSELL COST COSTING COSTS COUNTRY CRAIG CURRENT DATE DAVE DAY DECEMBER DENNIS DID DIVISION DO DOES DOLLAR DONE DUE@TO DURING EACH EIGHT EIGHTEEN EIGHTEENTH EIGHTH EIGHTY EITHER ELEVEN ELEVENTH END ENGLAND ENOUGH ESTIMATE-N ESTIMATE-V EVERY EVERYONE EXPECT EXPENSE EXPENSIVE FALL FARE FEBRUARY FEE FIFTEEN FIFTEENTH FIFTH FIFTY FIGURE FINAL FIRST FISCAL FIVE FOR FORTY FOUR FOURTEEN

FOURTEENTH FOURTH GET GETS GETTING GIVE GIVEN GIVES GIVING GO GOES GOING GONE GOT GOTTEN GROUP HAD HALF HALVES HAS HAVE HAVING HE HER HIM HIS HOW HOWMANY HOWMUCH HUNDRED I IF IFIP IJCAI IN INTERNATIONAL IS IT JANUARY JERRY JOHN JULY JUNE K KNOW L.A. LAST LATE LEFT LINDA LINGUISTICS LIST LONDON LONG LOS@ANGELES LYN LYNN MADE MAKE MAKES MAKHOUL MAKING MANY MARCH MASSACHUSETTS MAY ME MEETING MEMBER MISCELLANEOUS MONEY MONTH MORE MOST MUCH MY NEED NEW@YORK NEXT NINE NINETEEN NINETEENTH NINETY NINTH NO NOT NOTE NOVEMBER NOW OCTOBER OF ON ONE ONLY OR OTHER OUT OVERHEAD PAJARRO@DUNES PARTICIPANT PAUL PAY PENNSYLVANIA PEOPLE PER PERSON PHONOLOGY PITTSBURGH PLACE PLEASE PLUS PRINT PROJECT-N PROJECT-V PURPOSE QUARTER REGISTRATION REMAIN REST REVISE RICH RICHARD ROUND@TRIP SANTA@BARBARA SCHEDULE SECOND SEND SENDING SENDS SENT SEPTEMBER SEVEN SEVENTEEN SEVENTEENTH SEVENTH SEVENTY SHE SINCE SITE SIX SIXTEEN SIXTEENTH SIXTH SIXTY SO SOCIETY SOME SOMEONE SPEECH SPEND SPENDING SPENDS SPENT SPRING ST.LOUIS START STATUS STOCKHOLM SUMMER SUPPOSE SUPPOSED SUPPOSITION SUR SWEDEN TAKE TAKEN TAKES TAKING TEN TENTH THAN THANK@YOU THAT THE THEIR THEM THERE THESE THEY THIRD THIRTEEN THIRTEENTH THIRTIETH THIRTY THIS THOSE THOUSAND THREE TIME TO TOO TOOK TOTAL TRAVEL TRIP TWELFTH TWELVE TWENTIETH TWENTY TWO UNANTICIPATED UNBUDGETED UNSPENT UNTAKEN US VARIOUS VISIT WANT WAS WASHINGTON WE WEEK WENT WERE WHAT WHEN WHERE WHICH WHO WHOM WHOSE WILL WINTER WISCONSIN WITH WITHIN WORKSHOP YEAR YES YOU)

The syntactic categories:

(ADJ 23 (ACOUSTICAL ADDITIONAL AVAILABLE BIG COMPUTATIONAL CURRENT EACH ENOUGH EXPENSIVE FINAL FISCAL INTERNATIONAL LATE LEFT LONG MANY MISCELLANEOUS OTHER UNANTICIPATED UNBUDGETED UNSPENT UNTAKEN VARIOUS))

(ADV 18 (ALREADY ALSO ANYWHERE EITHER ENOUGH HOW LATE LONG MORE MOST MUCH NOW ONLY PLEASE SO THERE TOO YES))

(ART 8 (A AN NO THAT THE THESE THIS THOSE))

(CONJ 8 (AND BECAUSE BOTH IF OR PLUS SINCE SO))

(INTEGER 27 (EIGHT EIGHTEEN EIGHTY ELEVEN FIFTEEN FIFTY FIVE FORTY FOUR FOURTEEN NINE NINETEEN NINETY ONE SEVEN SEVENTEEN SEVENTY SIX SIXTEEN SIXTY TEN THIRTEEN THIRTY THREE TWELVE TWENTY TWO))

(MODAL 5 (CAN DID DO DOES WILL))

(N 70 (ACOUSTICS AIR AIRPLANE AMOUNT ASSOCIATION ASSUMPTION BEGINNING BREAKDOWN BUDGET BUS CAR CENT CHANGE CITY CONFERENCE COST COUNTRY DATE DAY DIVISION END ESTIMATE-N EXPENSE FALL FARE FEE FIGURE GROUP HALF HALVES LINGUISTICS LIST MEETING MEMBER MONEY MONTH MUCH NEED NOTE OVERHEAD PARTICIPANT PEOPLE PERSON PHONOLOGY PLACE PROJECT-N PURPOSE QUARTER REGISTRATION REST ROUND@TRIP SCHEDULE SITE SOCIETY SOME SPEECH SPRING STATUS SUMMER SUPPOSITION

```
        THANK@YOU TIME TOTAL TRAVEL TRIP VISIT WEEK WINTER
        WORKSHOP YEAR))

(NEG 1 (NOT))

(NPR 53 (ACL AI AMHERST APRIL ARPA SA AUGUST BATES BILL BONNIE
        BOSTON CALIFORNIA CARNEGIE C:)LARUSSO COSELL CRAIG DECEMBER
        DENNIS ENGLAND FEBRUARY IFIP IJCAI JANUARY JERRY JOHN JULY
        JUNE L.A. LINDA LONDON LOS@ NGELES LYN LYNN MAKHOUL MARCH
        MASSACHUSETTS MAY NEW@YORK NOVEMBER OCTOBER PAJARRO@DUNES
        PENNSYLVANIA PITTSBURGH RICH RICHARD SANTA@BARBARA
        SEPTEMBER ST.LOUIS STOCKHOLM SUR SWEDEN WASHINGTON
        WISCONSIN))

(ORD 23 (EIGHTEENTH EIGHTH ELEVENTH FIFTEENTH FIFTH FIRST
        FOURTEENTH FOURTH LAST NEXT NINETEENTH NINTH SECOND
        SEVENTEENTH SEVENTH SIXTEENTH SIXTH TENTH THIRD THIRTEENTH
        THIRTIETH TWELFTH TWENTIETH))

(PARTICLE 3 (IN ON OUT))

(POSS 5 (HER HIS MY THEIR WHOSE))

(PRECONJ 2 (BOTH EITHER))

(PREP 18 (ABOUT ABOVE AFTER AT BEFORE BY DUE@TO DURING FOR IN OF
        ON OUT PER SINCE TO WITH WITHIN))

(PRO 23 (ANYONE EVERYONE HE HER HIM I IT ME ONE SHE SOMEONE THAT
        THEM THESE THEY THIS THOSE US WE WHAT WHO WHOM YOU))

(QADV 2 (WHEN WHERE))

(QDET 5 (HOWMANY HOWMUCH WHAT WHICH WHOSE))

(QUANT 14 (ALL ANY BOTH EACH EITHER ENOUGH EVERY HOWMANY HOWMUCH
        MANY MORE MUCH OTHER SOME))

(QWORD 7 (HOW HOWMANY HOWMUCH WHAT WHICH WHO WHOM))

(SPECIAL 8 (DOLLAR HUNDRED K NO THAN THANK@YOU THOUSAND YES) )

(V 85 (AFFORD AM ANTICIPATE ARE ARRANGE ASK ASSUME ATTEND BE BEEN
        BEGINNING BEING BUDGET CAN CANCEL CHANGE CONTINUE COST
        COSTING COSTS DID DO DOES DONE END ESTIMATE-V EXPECT
        FIGURE GET GETS GETTING GIVE GIVEN GIVES GIVING GO GOES
        GOING GONE GOT GOTTEN HAD HAS HAVE HAVING IS KNOW LAST
        LEFT LIST MADE MAKE MAKES MAKING NEED NOTE PAY PRINT
        PROJECT-V REMAIN REVISE SCHEDULE SEND SENDING SENDS SENT
        SPEND SPENDING SPENDS SPENT START SUPPOSE TAKE TAKEN TAKES
        TAKING TOOK TOTAL TRAVEL VISIT WANT WAS WENT WERE WILL))
```

The syntactic features:

```
(INGCOMP (CANCEL CONTINUE GIVE START START))
(INTRANS (CONTINUE GO GO GO START))
(PASSIVE (CANCEL CONTINUE FIGURE GET GIVE  MAKE  MAKE  SEND  SEND
       START START TAKE) )
(QCOMP (CANCEL CONTINUE FIGURE GIVE SEND SEND TAKE))
(THATCOMP (END FIGURE))
(TRANS (CANCEL CONTINUE END FIGURE GET GIVE MAKE MAKE  SEND  SEND
       START START TAKE))
((ANAPHORIC) (WHICH))
((DETERMINED) (ANYONE I IT ME THAT THESE THIS  THOSE  US  WE  WHO
       WHOM YOU))
((INDOBJ FOR) (MAKE MAKE))
((NUMBER PL) (HALVES PEOPLE THEM THESE THEY THOSE US WE))
((NUMBER SG) (ANYONE HE HER HIM I IT ME ONE SHE SOMEONE THAT THIS
       WHO WHOM WHOM))
((NUMBER SG/PL) (WHAT WHAT WHICH WHO YOU))
((PARTICLEOF (LEAVE PUT ADD)) (IN))
((PARTICLEOF (ADD CONTINUE)) (ON))
((PARTICLEOF (LEAVE PRINT  SEND  MAKE  CANCEL  FIGURE  FIND  GO))
       (OUT))
((PASTPART) (BEEN COST DONE GIVEN GONE GOTTEN HAD LEFT MADE  SENT
       SPENT TAKEN))
((PNCODE 13SG) (WAS))
((PNCODE 1SG) (AM))
((PNCODE 3SG) (COST COST COST COST COSTS  DOES  DOES  GETS  GIVES
       GOES HAS IS MAKES SENDS SPENDS TAKES))
((PNCODE ANY) (CAN CAN DID))
((PNCODE X13SG) (ARE WERE))
((PNCODE X3SG) (COST DO WILL))
((PRESPART) (BEGINNING BEING COSTING GETTING GIVING GOING  HAVING
       MAKING SENDING SPENDING TAKING))
((ROLE OBJ) (HER HIM ME THEM US WHOM WHOM))
((ROLE SUBJ) (HE I SHE WE))
((ROLE SUBJ/OBJ) (WHAT WHICH WHO WHO))
((TNS FUTURE) (WILL WILL))
((TNS PAST) (COST DID DID GOT HAD MADE SENT SPENT TOOK  WAS  WENT
       WERE))
((TNS PRESENT) (AM ARE CAN CAN COST COST COST COST COST COSTS  DO
       DOES  DOES GETS GIVES GOES HAS IS MAKES SENDS SPENDS TAKES
       WILL))
((UNTENSED) (BE WILL)))) )
```

BIBLIOGRAPHY

[1] Baker, James K. (1975b)
Stochastic Modeling as a Means of Automatic Speech Recognition
PhD thesis, Speech and Computer Science,
Carnegie-Mellon Univ., April 1975

[2] Barnett, Jeffrey (1973)
A Vocal Data Management System
IEEE Trans. on Audio and Electroacoustics,
AU-21:3 (June 1973) p. 185-188

[3] Bates, M. (1974)
The Use of Syntax in a Speech Understanding System
in Erman, IEEE Symposium on Speech Recognition,
p. 226-233

[4] Bates, M. (1975)
Syntactic Analysis in a Speech Understanding System
Ph.D. thesis, Harvard University, 1975
also BBN Report No. 3116, Bolt Beranek and
Newman Inc., Cambridge, Mass. August 1975

[5] Bates, M. and Wolf, J. (1975)
Prosodics
technical report in BBN Speech Understanding
System QPR no. 2, BBN Report no 3080 (AI
Report no. 30), Bolt Beránek and Newman Inc,
Cambridge, Mass. (May, 1975)

[6] Bruce, Bertram (1976)
Pragmatics in Speech Understanding
Proceedings of the IJCAI, 1976

[7] Erman, Lee (ed.) (1974)
IEEE Symposium on Speech Recognition
IEEE, Inc. (1974)

[8] Forgie, James W. (1974b)
Speech Understanding Systems; Semiannual Technical Summary
Lincoln Laboratory, Lexington, Mass., 31 May 1974

[9] Fu, K.C. (1976)
SYNTACTIC PATTERN RECOGNITION
Springer-Verlag, N.Y.

[10] Hyde, S.R. (1968)
Automatic Speech Recognition: Literature Survey and Discussion
Report No. 45, Post Office Research Dept.
Dollis Hill, London, 1968

[11] Klovstad, John W. (1975)
personal communication

[12] Lea, W., Medress M., and Skinner, T. (1972)
Prosodic Aids to Speech Recognition:
I. Basic Algorithms and Stress Studies
Univac Report no. PX7940, Oct. 1972

[13] Lea, W., Medress, M., and Skinner, T. (1973)
Prosodic Aids to Speech Recognition:
II. Syntactic Segmentation and Stressed
Syllable Location
Univac Report no. PX10232, April 15, 1973

[14] Lipton, Richard J. and Snyder, Lawrence (1974)
On the Optimal Parsing of Speech
Research Report 37, (Oct. 1974) Dept. of Computer
Science, Yale University, New Haven, Conn.

[15] Nash-Webber, Bonnie (1974)
Semantic Support for a Speech Understanding System
IEEE Trans. on ASSP, 23:1 (Feb. 1975), p. 124-129

[16] Neely, R.B. (1973)
On the use of Syntax and Semantics in a Speech
Understanding System
PhD. thesis, Stanford Univ, 1973
also Tech Rept. Comp. Sci. Dept. CMU, May, 1973

[17] Newell, A. et al (1973)
SPEECH UNDERSTANDING SYSTEMS:
Final Report of a Study Group
North-Holland, Amsterdam

[18] Paxton, William H. (1974)
A Best-First Parser
in Erman, IEEE Symposium on Speech Recognition,
p. 218-225

[19] Paxton, William H. and Robinson, Ann E. (1975)
System Integration and Control in a Speech Understanding
System
Technical Note 111, Artificial Intelligence Center,
Stanford Research Institute, September 1975

[20] Reddy, D.R., Erman, L., and Neely, R.B. (1970)
The CMU Speech Recognition Project
IEEE System Sciences and Cybrenetics Conf.,
Pgh. Pa., 1970

[21] Reddy, D.R.,et.al. (1973b)
The HEARSAY Speech Understanding System:
An Example of the Recognition Process
Proc. 3rd IJCAI, Stanford, August, 1973, p. 185-193

[22] Reddy, D. R. (ed.) (1975)
Speech Recognition (Invited Papers Presented at the 1974
IEEE Symposium)
Academic Press, New York

[23] Rovner, P., Nash-Webber, B., and Woods, W. (1974)
Control Concepts in a Speech Understanding System
IEEE Trans. on ASSP,, 32:1 (Feb. 1975), p. 136-140

[24] Rovner, P., et.al. (1974)
Where the Words are:
Lexical Retrieval in a Speech Understanding System
in Erman, IEEE Symposium on Speech Recognition,
p. 160-164

[25] Rustin, Randall (ed.) (1973)
NATURAL LANGUAGE PROCESSING
Algorithmics Press, N.Y. 1973

[26] Schwartz, R. and Makhoul, J. (1974)
Where the Phonemes are:
Dealing with Ambiguity in Acoustic-Phonetic Recognition
IEEE Trans. on ASSP, 23:1 (Feb. 1975), p. 50-53

[27] Shirai, K. and Fujisawa, H. (1974)
An Algorithm for Spoken Sentence Recognition and its
Application to the Speech Input-Output System
IEEE Trans. on Systems, Man, and Cybrenetics,
Sept 1974, p. 475-479

[28] Walker, Donald E. (1972)
Speech Understanding Research
SRI Annual Technical Report, Oct 1971 - Oct 1972

[29] Walker, Donald E. (1973a)
Speech Understanding Through Syntactic
and Semantic Analysis
Proc. 3rd IJCAI, Stanford, Aug. 1973, p. 208-215

[30] Walker, Donald E. (1973b)
Automated Language Processing
in Cuadra (ed.), Annual Review of Information Science and
Technology, vol 8

[31] Wolf, Jared (1976)
Speech Recognition and Understanding
in Fu, Syntactic Pattern Recognition

[32] Woods, W.A. (1970)
Transition Network Grammars for Natural Language Analysis
CACM, 13:10 (Oct 1970) p. 591-606

[33] Woods, W.A. (1974)
Motivation and Overview of BBN SPEECHLIS:
An Experimental Prototype for Speech Understanding Research
IEEE Trans. on ASSP, 23:1 (Feb. 1975), p. 2-10

[34] Woods, W.A. (1975)
Syntax, Semantics, and Speech
in Reddy, Speech Recognition: Invited Papers
Presented at the IEEE Symposium, Academic Press

[35] Woods, W.A. et. al. (1974)
Natural Language Communication with Computers,
Final Report, Volume I, Speech Understanding Research
at BBN
BBN Report 2976, Bolt Beranek and Newman Inc.,
Cambridge, Mass., Dec., 1974

[36] Woods, W.A., et.al. (1972)
The Lunar Sciences Natural Language Information
System: Final Report
Report No. 2378, Bolt Beranek and Newman Inc.,
Cambridge, Mass.

END