R. M. Kaplan

BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

BBN Report No. 2378

Job No. 11501

THE LUNAR SCIENCES NATURAL LANGUAGE INFORMATION SYSTEM: FINAL REPORT

W.A. Woods

R. M. Kaplan

B. Nash-Webber

Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts

June 15, 1972

Prepared for:
The Language Research Foundation
131 Mt. Auburn Street
Cambridge, Massachusetts

Supported by:
Contract No. NAS9-1115
NASA Manned Spacecraft Center
Houston, Texas

THE LUNAR SCIENCES NATURAL LANGUAGE INFORMATION SYSTEM: FINAL REPORT

W.A. Woods

R.M. Kaplan

B. Nash-Webber

Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

June 1972

CONTENTS

PREFACE

This report describes the current state of a two-year research project aimed toward the development of a prototype natural language question-answering system for lunar geologists. During these two years, the project has made considerable progress toward the ultimate goal of providing a general purpose natural language interface between men and machines. The project has built on the results of previous work in natural language understanding and is itself merely a stepping stone on the path of discovery that may someday make computers as generally available and conveniently accessible as one's next-door neighbor.

The Lunar Sciences Natural Language Information System (LSNLIS) is the result of these two years of joint effort by Bolt, Beranek, and Newman, and the Language Research Foundation, Cambridge, Mass. It is an experimental, research prototype of a question-answering system to enable a lunar geologist to conveniently access, compare, and evaluate the chemical analysis data on lunar rock and soil composition that is accumulating as a result of the Apollo moon missions. The objective of the research has been to develop a natural language understanding facility sufficiently natural and complete that the task of selecting the wording for a complex request becomes a negligible effort for the geologist user. Such a goal has not been achieved by any previous "natural-language" question-answering system.

Chapter 1 of the report gives a brief introduction to the system and an explanation of its goals and objectives. Chapter 5 contains a discussion of the capabilities of the current system and an evaluation of the prospects for further development. In between, Chapters 2, 3, and 4 give more detailed descriptions of the analysis system, the English grammar it contains, and the general techniques and strategies of semantic interpretation used to interpret the meanings of the user's requests.

Appendices to the report contain a brief users manual and a collection of examples of the system's performance (which may be useful to the casual reader in establishing a concrete understanding of what the system does) as well as complete listings of grammar and semantic rules and detailed system documentation.

In addition to the authors, the following staff have participated in the project:

Joe Becker
Dan Bobrow
Ben Wegbreit

We are also especially grateful to Gail Hedtler and Elsie Leavitt for assistance in preparing this report.

This report expands, updates, and supercedes BBN Report #2265 (Woods & Kaplan, 1971) which described the state of the project at the end of its first year.

W. Woods
June, 1972

Chapter 1

INTRODUCTION

1.1 Background and Objectives

The Lunar Sciences Natural Language Information System (LSNLIS) is a prototype computer system which allows English language access to a large data base of lunar sample information. The system was developed jointly by Bolt Beranek and Newman Inc. and Language Research Foundation, Cambridge, Massachusetts for the NASA Manned Spacecraft Center, Houston, Texas.

The motivation for the LSNLIS project arises from the difficulty of obtaining the basic information required by the working scientist to formulate and test his hypotheses. The data that bear on a hypothesis may be scattered through the literature in many different papers, and the task of finding the papers, collecting the information, standardizing the units, and making the necessary computations for a given application, is a formidable task. Moreover, when the results are in and the computation has been made, other questions are suggested to the scientist by the results, and the process begins again. Imagine instead that the published findings had already been collected into a computer system, which not only could give references to the literature, but which actually "understood" the numbers and measurements reported in the documents and was capable of performing calculations on these numbers. The evidence for or against a given hypothesis could then be obtained in a matter of minutes instead of days or weeks or months. In such a system, the remaining obstacle to the scientist would be the task of discovering whether the data base contained the necessary information for his need, finding out the formats of the tables, the notations used, units of measurement, etc., and learning how to use the system--specifically, learning

a programming language for expressing requests. Imagine, further, then, that this computer system could understand the scientist's natural language so that the scientist could merely state his request in English and the system would be capable of understanding what information was needed and either provide it (by retrieval or computation) or tell the user that the request was beyond the scope of its data base. The goals of the LSNLIS project are to develop a system which is as close to this idealized goal as the present state of the art allows, and do it in such a way that continual extensions of the system's capabilities can be made to converge on this goal sometime in the future.

There are two important reasons why one might want to use English as a mode of communication between a man and a machine. First, the man already knows his natural language and if he is to use a computer seldom or as a minor part of his work, then he may not have the time or inclination to learn a formal machine language. Second, the human thinks in his native language, and if the mode of communication involves the free and immediate communication of ideas to the machine which the user is conceiving in the course of the interaction, then the additional effort required for the human to translate his ideas into another language more suitable to the machine may slow down or otherwise interfere with the interaction. English is therefore an attractive medium because the human can express his ideas in the form in which they occur to him.

1.1.1 Can We Build Such a System?

Although the state of the art in "understanding" natural language by machine is still very limited, significant advances in this area have been made in recent years. Since Simmons' first survey of question answering systems, (Simmons, 1965), our understanding of the mysterious "semantic interpretation" component has been made more clear by work such as Woods (1967,1968), and

the techniques for mechanically parsing natural language sentences have been advanced by the advent of transition network grammars and their parsing algorithms (Woods, 1969,1970). The field is now at the point where prototype applications to real problems can make significant contributions to our understanding of the problems of natural language communication with machines. It must be realized, that such applications are still essentially research vehicles, since the problems of mechanical understanding of natural language remain far from solution. However, by using real problems (rather than imaginary toy problems) as the vehicles for such research, one can not only focus the effort on problems in need of solution, but may also reap the additional benefit of producing a system which will perform (in its limited way) a task which someone really wants done. The LSNLIS prototype is such an application.

1.1.2 Method of Approach

The method of approach which we have adopted in this study has been to look ahead to the potential capabilities for a future LSNLIS system, and to adopt general solutions to problems that will remain valid for applications of considerably greater scope than the current project. We have therefore chosen to implement the retrieval component within a general semantic framework (see Woods, 1968) and to provide a comprehensive and rigorous grammar of the subset of English involved. We have an existing parsing system for transition network grammars (Woods, 1969,1970) to provide a powerful general parsing capability within reasonable amounts of processing time, and have operated on the resulting parse trees with a general purpose, rule-driven semantic interpretation procedure (Woods, 1967,1968) for transforming them into representations of their meanings. Although the goal of accepting an input request in any phrasing which a user might ask is one which will require additional grammar development and semantic work, the system has already achieved considerable progress towards this goal, and the components and organization which we have used in building the system permit continual gradual evolution towards its achievement.

All of the components of the system have been implemented in BBN-LISP on the PDP-10 computer at BBN in Cambridge, Mass., running under the TENEX time sharing system with hardware paging and a virtual core memory for each user of up to 256K. Although there is considerable overhead in running time for programs written in LISP and executed in a paged environment, the flexibility of this system has been a critical factor in the development of the present level of capability within the time scale of the contract.

The design of the current system was carried out in a way that attempted to maximize the flexibility for such basic changes as: changing notations in dictionaries, changing parsing strategies, and modifying semantic interpretation rules and procedures; and indeed all of these have been changed extensively in the course of this project in order to achieve the current level of performance. Thus the current system represents the result of considerable evolution which would not have been possible within this time scale with a more rigid style of programming or a less flexible programming language.

1.2 Capabilities of the Current System

The current LSNLIS prototype allows a lunar scientist to ask questions, compute averages and ratios, and make selective listings based on the information in a chemical analysis data table. He can also retrieve references from a keyphrase index and make changes in the data base. The system permits the user to easily compare the measurements of different researchers, compare the concentrations of elements or isotopes in different types of samples or in different phases of a sample, compute averages over various classes of samples, compute ratios of two constituents of a sample, etc.--all in straightforward natural English.

The system removes from the user the burden of learning the detailed formats and codes of the data base tables, or learning a special programming language. For example, the system knows the various ways that a user may refer to a particular class of samples, it knows whether a given element is stored in the data base in terms of its elemental concentration or the concentration of its oxide, it knows what abbreviations of mineral names have been used in the tables, etc., and it converts the user's request into the appropriate form to agree with the data base tables, regardless of the form in which he actually makes his request. Thus, the present system has already made significant strides toward making the communication with the machine so natural and conveneint that it need not interfere with the researcher's train of thought.

In the following sections we will present a superficial description of the system and the kinds of operations it performs. More detailed descriptions of the organization of the system and the way it operates are given in chapters 2, 3, and 4, and in the appendices.

1.2.1 System Components

The LSNLIS system consists of three major components--a transition network parser with a large grammar of English and a large dictionary, a general purpose semantic interpretation component, and a retrieval component consisting of the data base and a collection of general purpose and specific retrieval functions. The parser performs a detailed syntactic analysis of the user's question and passes the resulting parsing to the semantic interpretation component for translation into the formal request language of the retrieval component. The first two components of the system thus function to translate the user's request into a program in the formal request language which will compute the answer to the question. This program is then executed in the retrieval component to produce the answer.

The system is operational on the TENEX time-sharing system in two 256K tasks (called "forks")--one containing the parser, interpreter, grammar and dictionary, and the other containing the data base and retrieval functions. Formal requests and answers to questions are passed between the two forks by means of file buffers. This division of the system between language processing component and retrieval component would make it easy to operate in a mode in which the language processing component resided on one computer and the retrieval component on another computer somewhere else.

The LSNLIS system presently contains a dictionary of approximately 3500 words consisting of a selection of general English vocabulary and a large technical vocabulary of geological and chemical terms. It's grammar is a transition network grammar of the type described in Woods (1969, 1970) and produces output in the form of Chomsky-type deep structures. This output is translated into formal requests for the retrieval component by a general purpose, rule-driven semantic interpreter.

1.2.2 The Data Base

The LSNLIS system is intended to eventually handle any number of data base files with different structures and characteristics. However, for the initial prototype, two data base files were provided by MSC. One is a 13,000 line table of chemical and age analyses of the Apollo 11 samples extracted from the reports of the First Annual Lunar Science Conference, and the second is a keyphrase index to those reports. Samples of these two data bases are shown in figures 2-9 and 2-10. The first contains entries specifying the concentration of some constituent in some phase of some sample, together with a reference to the article in which the measurement was reported. (There are generally several entries for each combination of sample, phase, and constituent--measured by different investigators.) The second is a list of keyphrases and documents which have been indexed by them.

The major thrust of this project has been the development of the parsing and semantic interpretation components to handle the natural language querying aspect of the problem. The retrieval component was implemented primarily to provide a complete on-line environment for carrying out the research. Thus, the retrieval component has been implemented in a relatively straightforward manner using the TENEX system's automatic paging facility to take care of the problems of memory allocation so that we could devote most of our effort to the natural language problems. Since the retrieval component resides in an entirely separate fork interfaced via the general purpose request language discussed above, there is no difficulty in substituting a more sophisticated retrieval component later.

When an input request has been processed in the English processor fork, the resulting formal request is communicated to the retrieval fork via a file buffer and control is passed to the retrieval fork until the request has been executed. The answer is then returned to the user via a file buffer and the English processor fork resumes control. This organization means that in principal, there would be little difference whether the retrieval component resided in another fork of the TENEX system or in another computer (e.g. at Houston) connected by telephone lines to the BBN computer.

1.2.3 Intended Types of Questions

Before beginning a discussion of the capabilities of the LSNLIS system, it is important to recognize a sharp distinction between the types of questions which the prototype system will now handle, the types of questions toward which it aspires, and the types of questions which in principle could be asked, but which we have no intentions of handling. The distinction between the first two types obviously changes with time, since new constructions are continually being added to the repertory of the system, even as this report is being written. This distinction

is primarily a measure of how far we have progressed toward our goals. A more important distinction is that between questions for which the system is intended and those for which it is not.

In designing the system, we assume that the questions will be asked by a scientist with an interest in obtaining information and that they will be stated in a direct and straightforward manner. Thus, we are especially concerned with handling constructions which might be used by such a user, and we do not want to devote extensive effort to handling "frivolous" questions. Thus, when choices must be made (as they must) as to which constructions are most important in the development of the grammar and understanding capability of the system, priority is given to constructions which we feel might be used by a serious user in need of information. We do not, for example, assign much priority to handling constructions such as "tag questions" (e.g. "Lunar rocks contain oxygen, don't they?", "Sample S10046 is a breccia, isn't it?", etc.), and many other constructions which occur in English but would not appreciably increase the usefulness of the system. Likewise, we are not interested in questions which require evaluation, judgment, or conclusions on the part of the system (e.g. "Does the moon have a hot core?", "What is the most probable source of the lunar dust?", etc.). It is the task of the scientist to interpret the data, and we are trying to aid him in this task--not replace him.

The questions for which the system is intended, are straightforward factual questions, which arise directly from measurements and observations of the samples. The following list gives a representative sample of the types of sentences for which the system is intended:

1. List the rocks which contain chromite and ulvospinel.
2. Give me all references on fayalitic olivine.
3. What minerals have been identified in the lunar samples?
4. What analyses of olivine are there?

5. What is the average analysis of Ir in rock S10055?
6. List the modes for all low Rb rocks.
7. Give me the K / Rb ratios for all lunar samples.
8. Has the mineral analcite been identified in any lunar sample?
9. What is the concentration of La in rock S10034?
10. Identify all samples in which glass was found.
11. Give me all modal analyses of lunar fines.
12. In what samples has apatite been identified?

1.2.4 Querying the Data Base

In this section we will give a sample of the types of querying interactions which the system permits. More examples are given in Appendix G.

Perhaps the most typical example of a request which a geologist might make to the LSNLIS system is illustrated by the following protocol:

```
38**(WHAT IS THE AVERAGE CONCENTRATION OF ALUMINUM IN
HIGH ALKALI ROCKS)
***
PARSING
1331 CONSES
4.987 SECONDS
INTERPRETING
2427 CONSES
11.025 SECONDS
INTERPRETATIONS:
(FOR THE X13 / (SEQL (AVERAGE X14 / (SSUNION X15 / (SEQ TYPEAS) :
T ; (DATALINE (WHQFILE X15) X15 (NPR* X16 / (QUOTE OVERALL)) (NPR*
X17 / (QUOTE AL203)))) : T)) : T ; (PRINTOUT X13))


BBN LISP-10  03-09-72 ...
EXECUTING
(8.134996 . PCT)

------
```

(Here, the system has typed the two asterisks, the user typed the question, beginning and ending with parentheses, and the system typed the rest. The comments 1331 CONSES and 4.987 SECONDS give a record of the memory resources and the time used during

the parsing phase. A similar record is generated for the interpretation phase. The expression following the comment INTERPRETATIONS: is the formal retrieval program which is executed in the data base to produce the answer.) This request illustrates a number of features of the system:

1. The user types the question exactly as he would say it in English (terminal punctuation is optional and was omitted in the example).

2. The system has translated the phrase "high alkali rocks" into the internal table form TYPEAS.

3. The system has filled in an assumed OVERALL phase for the concentration since the request does not mention any specific phase of the sample in which the concentration is to be measured.

4. The system is capable of computing answers from the data base as well as simply retrieving them (the average was not stored information).

Perhaps the simplest operation which the system will perform for the user is to collect and list selected portions (not necessarily contiguous) of the data base. For example, in response to a request "Give me all analyses of S10046," the system would respond as follows:

```
***
37**(GIVE ME ALL ANALYSES OF S10046)
***
PARSING
1456 CONSES
9.445 SECONDS
INTERPRETING
2112 CONSES
8.502 SECONDS
INTERPRETATIONS:
(DO (FOR EVERY X9 / (SSUNION X12 / (SEQ MAJORELTS) : T ; (DATALINE
(WHOFILE (NPR* X10 / (QUOTE S10046))) (NPR* X10 / (QUOTE S10046))
(NPR* X11 / (QUOTE OVERALL)) X12)) : T ; (PRINTOUT X9)))


BBN LISP-10  03-09-72 ...
EXECUTING
```

```
I HAVE 15 HITS
DO YOU WANT TO SEE THEM? YES
3956     S10046  OVERALL    SIO2        44.06752   PCT          D70-235    0
3967                        TIO2        8.3405
3968                                    6.50559                 D70-254
3865                        AL203       11.7149                 D70-235
3900                        FEO         16.9818
3901                                    15.438                  D70-254
3928                        MNO         .20659                  D70-235
3929                                    .22725                  D70-254
3927                        MGO         9.11845                 D70-235
3875                        CAO         13.71216
3917                        K20         .20478
3918                                    .19515                  D70-242
3919                                    .14455                  D70-254
3933                        NA20        .4718                   D70-235
3934                                    .50146                  D70-254
------
```

This example illustrates some additional features of the system. Again, since no phase was mentioned, the system assumed the OVERALL phase (i.e. the rock as a whole). If the user had wanted to see all the phases, he could have said explicitly "for all phases". Similarly, since no specific elements or isotopes were mentioned, the system assumed a standard list of major elements was intended (Our geologist informant assures us that this is what a geologist would mean by such a question). Again, if the user really wanted to see all chemical element analyses, he could say so explicitly. The comment I HAVE 15 HITS DO YOU WANT TO SEE THEM? illustrates another feature of the system. If the result of a request is more than 5 lines of output, the system types this comment and gives the user the option of listing them offline.

In addition to averaging and listing, the system can also compute ratios, count, and interpret some anaphoric references and comparatives as indicated in the following examples:

```
***
31**(HOW MANY BRECCIAS CONTAIN OLIVINE)
***
PARSING

GC: 8
12263, 12774 FREE WORDS
815 CONSES
4.633 SECONDS
INTERPRETING
1514 CONSES
7.29 SECONDS
INTERPRETATIONS:
(FOR THE X12 / (SEQL (NUMBER X12 / (SEQ TYPECS) : (CONTAIN X12 (NPR*
X14 / (QUOTE OLIV)) (QUOTE NIL)))) : T ; (PRINTOUT X12))


BBN LISP-10  03-09-72 ...
EXECUTING
(5)

------


***
32**(WHAT ARE THEY)
***
PARSING
487 CONSES
2.755 SECONDS
INTERPRETING
1158 CONSES
4.053 SECONDS
INTERPRETATIONS:
(FOR EVERY X12 / (SEQ TYPECS) : (CONTAIN X12 (NPR* X14 / (QUOTE OLIV))
(QUOTE NIL)) ; (PRINTOUT X12))


BBN LISP-10  03-09-72 ...
EXECUTING
S10019
S10059
S10065
S10067
S10073

------
```

```
34**(DO ANY SAMPLES HAVE GREATER THAN 13 PERCENT ALUMINUM)
***
PARSING
981 CONSES
4.614 SECONDS
INTERPRETING
902 CONSES
3.566 SECONDS
INTERPRETATIONS:
(TEST (FOR SOME X16 / (SEQ SAMPLES) : T ; (CONTAIN' X16 (NPR* X17
/ (QUOTE AL203)) (GREATERTHAN 13 PCT))))


BBN LISP-10  03-09-72 ...
EXECUTING
YES.
NO HITS
T

------


***
35**(WHAT ARE THOSE SAMPLES)
***
PARSING
607 CONSES
3.34 SECONDS
INTERPRETING
625 CONSES
2.38 SECONDS
INTERPRETATIONS:
(FOR EVERY X16 / (SEQ SAMPLES) : (AND T (CONTAIN' X16 (NPR* X17 /
(QUOTE AL203)) (GREATERTHAN 13 PCT))) ; (PRINTOUT X16))


BBN LISP-10  03-09-72 ...
EXECUTING
GC: 8
6414, 12546 FREE WORDS
I HAVE 10 HITS
DO YOU WANT TO SEE THEM? YES
S10005
S10063
S10066
S10067
S10070
S10073
S10074
S10075
S30084
S10085

------
```

```
30**(LIST K / RB RATIOS FOR BRECCIAS)
***
PARSING
662 CONSES
3.366 SECONDS
INTERPRETING
1642 CONSES
6.537 SECONDS
INTERPRETATIONS:
(DO (FOR GEN X9 / (SSUNION X10 / (SEQ TYPECS) : T ; (RATIO (QUOTE
K20) (QUOTE RB) X10 (NPR* X11 / (QUOTE OVERALL))) : T ; (PRINTOUT
X9)))


BBN LISP-10  03-09-72 ...
EXECUTING
I HAVE 17 HITS
DO YOU WANT TO SEE THEM? YES
(472.2222 S10018 D70-205)
(473.5884 S10018 D70-242)
(518.2477 S10019 D70-218)
(345.4411 S10019 D70-256)
(463.3003 S10021 D70-242)
(568.8333 S30046 D70-235)
(462.4408 S10046 D70-242)
(408.2933 S10048 D70-220)
(566.1499 S10056 D70-235)
(480.1913 S10059 D70-253)
(481.85 S10060 D70-235)
(457.9177 S10060 D70-242)
(487.5714 S10060 D70-248)
(489.1304 S10061 D70-205)
(458.9973 S10065 D70-236)
(473.1551 S10065 D70-258)
(500.173 S10073 D70-215)

------
```

The system also understands restrictive relative clauses and certain adjective modifiers (some of which cause restrictions on the range of quantification of the noun phrase and some of which change the interpretation of the head they modify). Some other modifiers (such as "lunar" modifying samples) are known to be redundant and are deliberately ignored. The following example contains all three:

```
46**(LIST MODAL PLAG ANALYSES FOR LUNAR SAMPLES
THAT CONTAIN OLIV)
***
PARSING
1099 CONSES
4.346 SECONDS
INTERPRETKNG
2774 CONSES
12.33 SECONDS
INTERPRETATIONS:
(DO (FOR GEN X20 / (SSUNION X1 / (SEQ SAMPLES) : (CONTAIN X1 (NPR*
X3 / (QUOTE OLIV)) (QUOTE NIL)) ; (DATALINE (WHOFILE X1) X1 OVERALL
(NPR* X4 / (QUOTE PLAG)))) : T ; (PRINTOUT X20)))

BBN LISP-10  03-09-72 ...
EXECUTING

GC: 30
510, 1022 FREE WORDS
I HAVE 13 HITS
DO YOU WANT TO SEE THEM? YES
1679     S10020   OVERALL    PLAG          30.7        ***          D70-159    0
1680                                        21.4                     D70-173    31
1681                                        28.5                                40
1682                                        24.6                     D70-305    0
2141     S10022                             15.6                     D70-179
3109     S10044                             33.1                     D70-154    41
3110                                        34.1                                42
4440     S10047                             37.8                     D70-159    0
5796     S10058                             37.1                     D70-155
8582     S10072                             20.4                     D70-173
8583                                        18.5                     D70-179
9311     S10084                             22.0                     D70-186
9312                                        15.0                     D70-304

------
```

The structure of the formal query language for accessing the data base and the techniques for semantic interpretation enable the user to make very explicit requests with a wide range of diversity within a natural framework. As a natural consequence of the arrangement, it is possible for the user to combine the basic predicates and functions of the retrieval component in ways that were not specifically anticipated, to ask questions about the system itself. For example, one can make requests such as "List the phases.", "What are the major elements?", "How many minerals are there?", etc. Although

these questions are not likely to be sufficiently useful to merit special effort to handle them, they fall out of the mechanism for semantic interpretation in a natural way with no additional effort required. If the system knows how to enumerate the possible phases for one purpose, it can do so for other purposes as well. Furthermore, anything that the system can enumerate, it can count. Thus, the fragmentation of the retrieval operations into basic units of quantifications, predicates, and functions provides a very flexible and powerful facility for expressing requests.

In addition to the above operations, the system provides facilities for requesting keyphrase document retrieval and for updating and adding to the data base - both in natural English.

1.2.5 User Aids

In any natural language understanding system (even in people) it will occasionally (or frequently) happen that the user will be misunderstood. This is especially true for current computer systems due to their limited linguistic capabilities. It is therefore important that the system be able to give the user some useful feedback when it fails to understand him so that the user can adjust his requests to meet the limitations of the system. LSNLIS has extremely limited capabilities in these directions but we have implemented a small collection of user aids to help the user when his request fails to be understood.

A sentence may fail to be understood by LSNLIS for any of several reasons. First, it may fail to parse at all in the syntactic component. This may be due to its being genuinely ungrammatical, its use of a word in a sense not included in the dictionary, its use of a totally unknown word, or due to bugs in the system grammar or dictionary. The system notifies the user when it encounters an unknown word and won't proceed until the user either specifies a synonym (or a dictionary

entry for the new word if he knows how, which is unlikely) or quits. Bugs in the grammar and dictionary would gradually be detected and corrected in a working system and are thus not a conceptual problem. However, we have no effective solution in the current system for sentences that fail to parse when all the words are known. This is detected by the system when it has tried all of the alternative choices at its disposal and has failed to find a parse. At this point it has no idea which of its analysis paths was "closest" to being right. The best it can do is tell you how far into the sentence it got in a left-to-right parse, but this is not really very satisfactory.

If a sentence parses successfully, it may still fail to be understood due to a failure in the semantic interpretation component. When a request fails to interpret, it may be for one of two reasons: There may be no semantic rules available in the system for interpreting some node of the syntactic structure, or none of the semantic rules given for interpreting some word in context actually match. This latter situation may be the result of incorrect parsing, for example, incorrect modifier placement. The user, who may see the syntactic structure by setting PPRINT to T, has the recourse of seeking an alternative parsing by saying to TALKER, GO(PARSE).

When RULES fails to find any semantic rules to use in interpreting a sentence node, it calls the function SYNONYMS?. SYNONYMS? asks the user whether the head of the current node is a synonym of one of its known words, of which it was previously unaware. If the user can specify a synonym, the head is marked appropriately, and RULES returns the semantic rules to use in matching the node. The user is given the alternative of quitting the interpretation if he is unable to provide any usable synonyms. The nouns that the system understands (i.e. the nouns with which a set of NRULES is associated) are recorded on the list SEM-NOUNS, while the verbs which it

can understand as the head of an S node are kept on the list SEM-VERBS. These lists should be updated when new words become interpretable in the system via new NRULES and SRULES.

For example, one of the concepts the semantic interpreter understands is that of an analysis. Consider a request for

**(W DETERMINATIONS IN TYPE B ROCKS)

The request will fail because RULES can find no NRULES to use in matching the node headed by "determinations". "Determination" neither has NRULES of its own, nor is it marked as being synonymous with any other word in the system. With USERFLAG on, the function SYNONYMS? will solicit the user for usable synonyms and should find out that "determination" is synonymous with "analysis". From here, the interpretation will proceed without further problem. (It is not necessary that the two words be synonymous in all contexts, just so that when they are in the same context, they are indeed synonymous.)

Even if RULES is able to provide a list of semantic rules to use in interpreting a node, there is the possibility that none of them will match. The templates in the rules specify one or several possible structural descriptions for the node and also semantic or lexical requirements that its components must meet for the rule to apply. A rule may fail to match because some structural component is missing, or because the semantic or lexical requirements on some subnode fail. When MATCHER fails to match any of the rules given it by RULES, it calls the function NO-MATCHES. NO-MATCHES informs the user that the system cannot understand his use of the word which heads the node, within the context given by the node. The user is given the option of quitting (at which point he may call for his request to be reparsed) or breaking and investigating the problem for himself. (The latter would only be useful for a programmer familiar with the operation of the semantic interpreter.)

It may also happen that the semantic interpreter does not understand a user's request completely. This happens because a semantic rule can match a node without using all of its constituents. In this case, the semantic interpreter can produce an interpretation, but it will be missing some of the information specified in the request. Sometimes, this information is stylistic: For example, the relative clause in the request "Are there any analyses of Whitlockite in the samples that you have?" Since the system takes "analyses of Whitlockite in the samples" to refer only to the ones it knows about, the relative clause "that you have" does not state any new information. If the system were to tell the user that it could not understand "that you have" as a modifier of the rest of the noun phrase, which it does understand, the user should be able to tell the system to ignore it.

However, sometimes the constitutents the Semantic Interpreter has ignored are really necessary for the node's complete interpretation. When this happens, the user should be informed of the constituents being ignored and given the opportunity to scrap the request and try again. For example, the system does not have any referent for "microcrystalline inclusion" beyond that of papers which have been written on the subject. In a request for "Analyses of microcrystaline inclusions in olivines". the Semantic Interpreter would otherwise ignore this unrecognizable phrase and go on to interpret "Analysis in olivine", which would certainly be incorrect. In the current system, the user is informed whenever the Semantic Interpreter is about to ignore any of three types of noun phrase modifier - adjective, prepositional phrase, or relative clause, and he is asked whether it is safe to ignore it. If he says no, then the request is aborted.

1.2.6 Sample Sentences Handled

The following is a list of types of sentences handled as of March 1972:

1. (Samples with silicon)
   (Which rocks do not contain chromite and ulvospinel)
   (Give me all lunar samples with magnetite)
   (In which samples has apatite been identified)
   (Which elements has someone found in breccias and in basalts)

2. (What is the specific activity of Al26 in soil)

3. (Analyses of strontium in plagioclase)
   (What are the plag analyses for breccias)
   (Rare earth analyses for S10005)
   (I need all chemical analyses of lunar soil)
   (Chemical compositions of glassy materials)
   (What is the composition of ilmenite in rock 10017)
   (Has anyone analysed rock 10046 for major elements)
   (What are the analyses of aluminum in vugs)
   (Nickel content of opaques)

4. (Which samples are breccias)
   (What are the igneous rocks)
   (Are any of the samples volcanics)
   (What type of rock is S10003)
   (What types of sample are there)

5. (What is the average concentration of olivine in breccias)
   (What is the average analysis of olivine in breccias)
   (What is the average olivine concentration)
   (What is the average age of the basalts)
   (What is the average potassium / rubidium ratio in basalts)
   (In which breccias is the average concentration of titanium greater than 6 percent)
   (What is the average concentration of titanium in each breccia)
   (What is the average concentration of tin in breccias)
   (What is the mean analysis of iridium in type b rocks)
   (I want the average composition for glasses in dust)
   (What is the average plagioclase content in crystalline rocks)

6. (Modal plag analyses for S10058)
   (Modal olivine analyses)
   (Give me the modal olivine analyses for S10022)
   (Give me all modal analyses of plag in lunar fines)

7. (Which samples have greater than 20% modal plagioclase)
(Which samples are more than 20 percent plag)
(How many rocks have greater than 50 ppm nickel)
(Which samples contain more than 15 ppm barium in plag)

8. (How much titanium does S10017 contain)
(How much nickel is in rock 10046)

9. (What is the number of phases in each sample)
(How many samples contain titanium)
(How many papers have been published on lunar material)
(How many different moon rocks do we have)

10. (Bulk chemistry of soil samples)
(Give me all references on fayalitic olivine)

11. (Of the type A rocks which is the oldest)
(Which rock is the oldest)
(Which is the oldest rock)
(The highest titanium concentration)

Chapter 2

THE ANALYSIS SYSTEM

2.1 Overview

In order to "understand" and respond correctly to an English query, it is necessary not only to determine the syntactic structure of the input sentence but also to determine the "meaning" of the sentence to the system. This is determined by both the syntactic structure of the sentence and semantic information about the particular words which occur in it as they are related to the data base. In the MSC application, the meaning of a request is a procedure for computing its answer. The LSNLIS system represents this meaning in a powerful formal request language (Woods, 1968) which is then executed in the retrieval component to produce the answer to the request. The system processes English queries in three successive phases:

(i) syntactic analysis using heuristic information to select the most "likely" parsings,

(ii) semantic interpretation to produce a formal representation of the "meaning" of the query to the system,

(iii) execution of this formal expression in the retrieval component to produce the answer to the request.

The English Language preprocessor makes use of a general parsing algorithm for transition network grammars and a general rule-driven semantic interpretation procedure which

were developed at Harvard University and BBN over a period of years from 1967 to 1970, and which have been reported on in the literature (Woods, 1967, 1968, 1969, 1970). For this contract, we have adapted these programs to the MSC application, developed a grammar for a large subset of English, developed a set of semantic interpretation rules for interpreting requests for references, chemical analyses, ratios, etc. and constructed a large dictionary of approximately 3500 words. In addition, we have provided functions for setting up and interrogating a data base of chemical analysis data, computing averages and ratios, and retrieving references from inverted files in response to Boolean combinations of key words. The overall organization of the English Language Preprocessor is shown in Figure 2-1. In this chapter we will be given a basic description of the operation of the major components of the system; a complete and detailed description of the individual functions which make up the system is given in Appendix D.

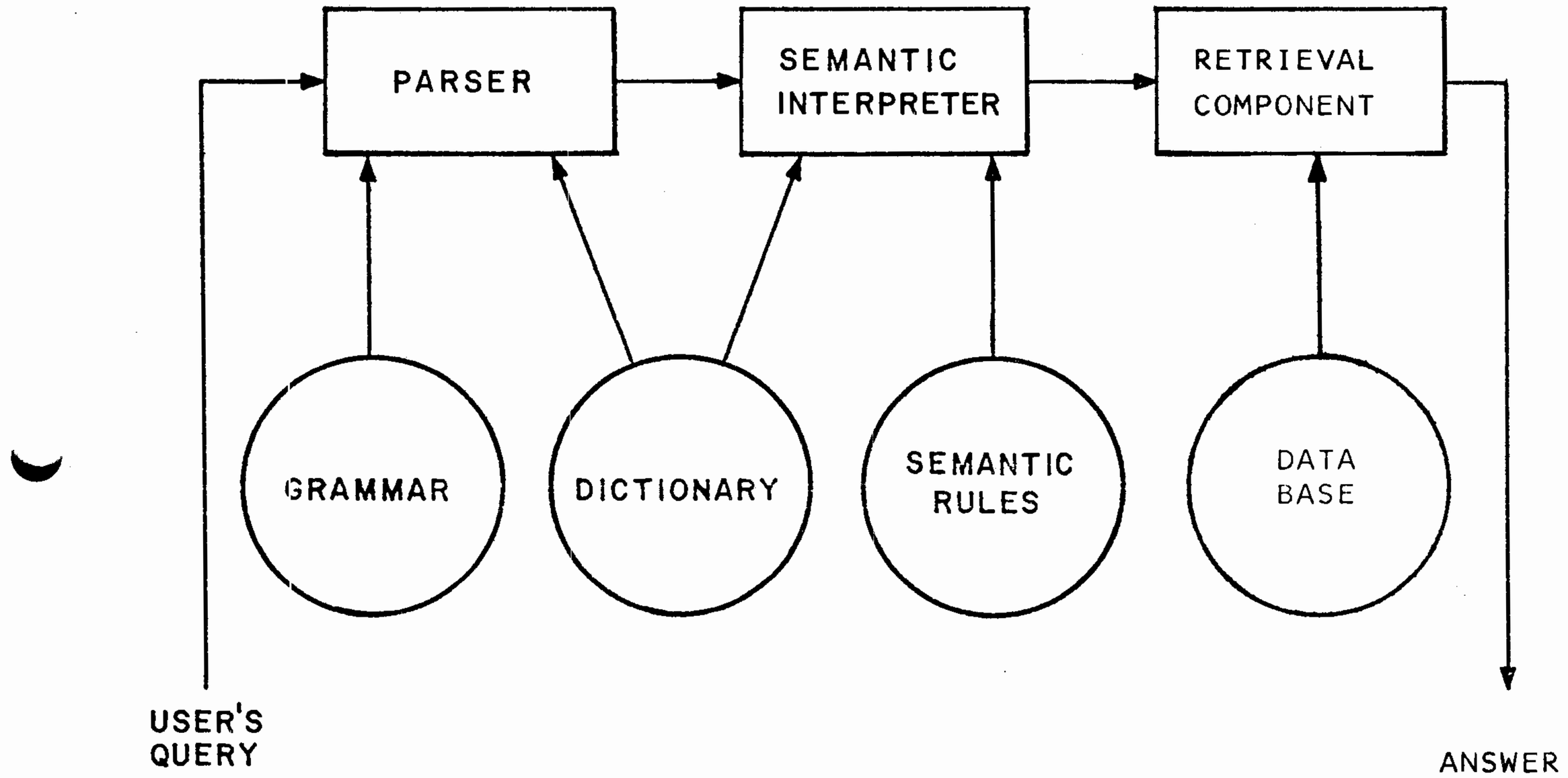


Figure 2-1. Organization of the LSNLIS system

2.2 The Parsing System

The MSC English preprocessor makes use of a general parsing procedure for transition network grammars developed at Harvard University and extended at Bolt, Beranek, and Newman (Woods, 1969,'70,'72). This section gives a basic description of the transition network grammar model and the operation of the parsing system. (For more detail see Appendix D.) For more detail on the philosophy and motivation of the transition network grammar model, see the above cited references.

The transition network grammar model is an extension of the notion of state transition diagram well-known to automata theory. A transition network grammar consists of a network of nodes with arcs connecting them. The nodes represent states of a hypothetical parsing machine and the arcs connecting them represent possible transitions and are labelled with the types of events in the environment of the machine which permit the transitions. In the case of a transition network grammar, the types of events are the occurrences of words and phrases in the input string upon which the grammar is operating.

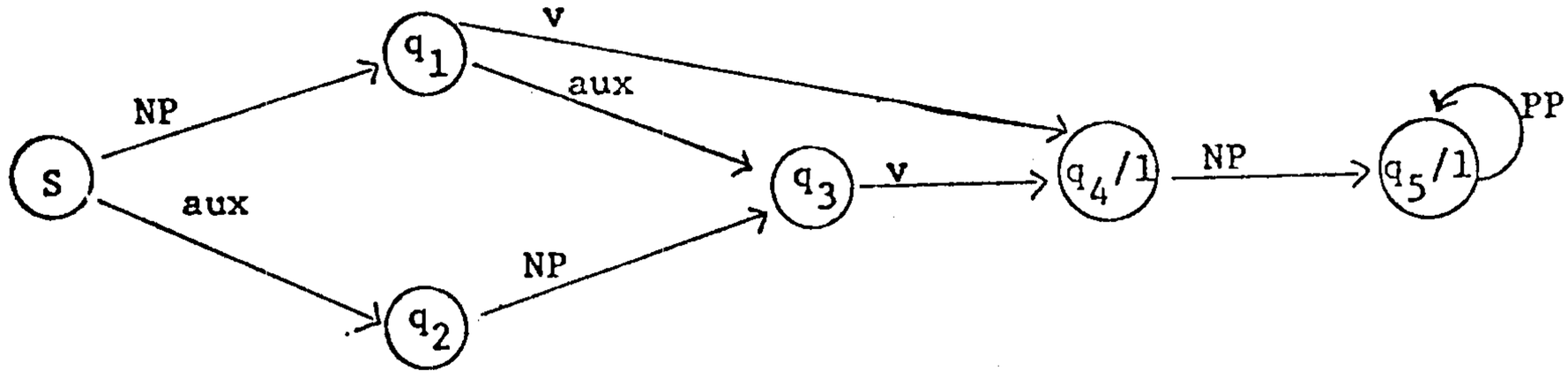
The type of transition network grammar which we are using for the MSC grammar is an augmented recursive transition network grammar in which the arcs of the network include arbitrary conditions for determining when their transitions are permitted, and arbitrary structure-building actions which build up the syntactic representation of the sentences recognized. This model has only recently been applied in the field of natural language processing, and it provides a practically feasible means of obtaining the types of analyses formerly obtainable only from laborious inversions of transformations specified by a transformational grammar of the

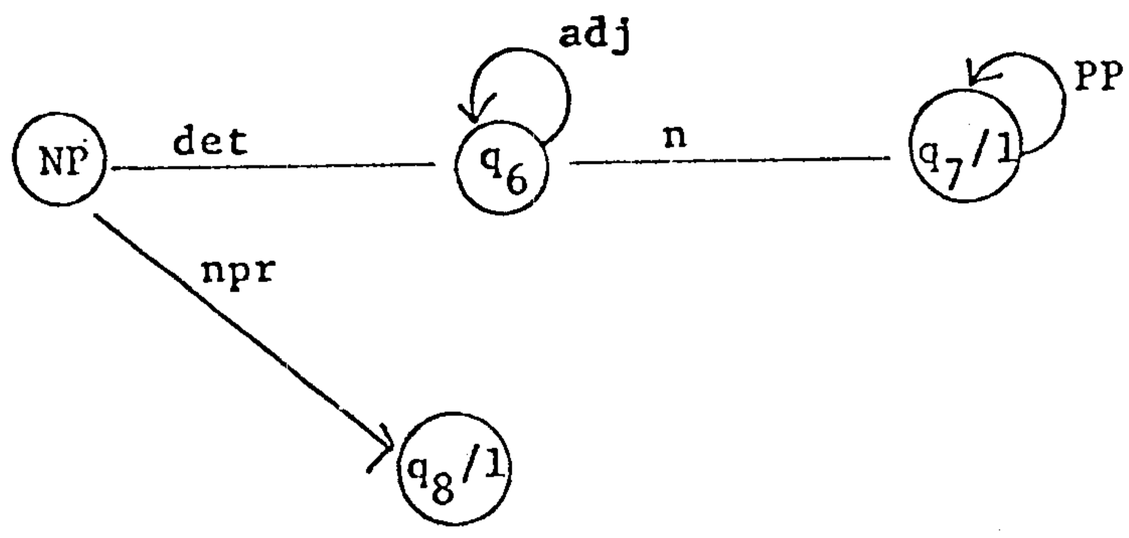
Chomsky variety (Chomsky, 1965). The transition network model permits analyses effectively equivalent to those of the transformational grammar, and it is the first parsing procedure to enable such sophisticated linguistic principles to be embodied in a practically feasible manner.

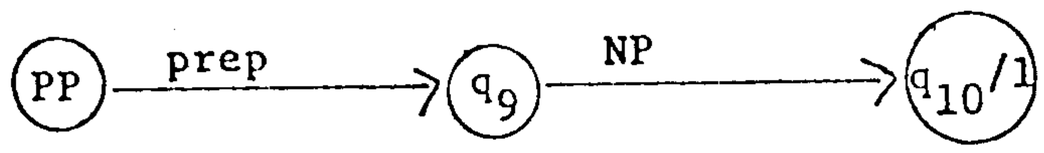
We say a state accepts a given string if that string permits a sequence of transitions which lead from that state to some state which is distinguished as a "final" state (in our system, this is indicated by the presence of a POP arc which not only marks the state as being a final state, but orders the alternative of accepting the string at that point with respect to the other arcs which leave that state).

A recursive transition network grammar contains two types of arcs--lexical arcs which correspond to transitions permitted by single words, and recursion arcs (or PUSH arcs) which invok re-cursive applications of the network to recognize a phrase or word gouping of some kind. The most common type of the former is the CAT arc which recognizes members of a specified syntactic category. For example a CAT N arc permits a transition if the current word in the input string is a word in the syntactic category N (for noun). A PUSH NP/ arc permits a transition if the state NP/ can recognize a noun phrase at the current spot in the input string. In addition, there are JUMP arcs which perform actions without advancing the input string (normally the input string is advanced past the word or words which permit a transition) and a variety of other special arc types. These are covered more fully in the appendices.

Figure 2-2 gives a simple example of a transition network grammar. It recognizes simple declarative and interrogative sentences with noun phrases containing adjective modifiers and prepositional phrases. Lexical arcs are indicated with lower case labels, and PUSH arcs are indicated with upper case labels that name the state to which control is to "push". It is easy to visualize the range of acceptable sentences from inspection of the transition network. To recognize the sentence, "Did the red barn collapse," the network is started in state S. The first transition is the aux transition to state q_2 permitted by the auxiliary "did". From state q_2 we see that we can get to state q_3 if the next "thing" in the input string is a NP. To ascertain if this is the case, we call the state NP. From state NP we can follow the arc labeled DET to state q_6 because of the determiner "the". From here, the adjective "red" causes a loop which returns to state q_6, and the subsequent noun "barn" causes a transition to state q_7. Since state q_7 is a final state, it is possible to "pop up" from the NP computation and continue the computation of the top level S beginning in state q_3 which is at the end of the NP arc. From q_3 the verb "collapse" permits a transition to the state q_4, and since this state is final and "collapse" is the last word the string is accepted as a sentence.







S is the start state

q_4, q_5, q_7, q_8, and q_{10} are the final states

Figure 2-2. A Sample Transition Network

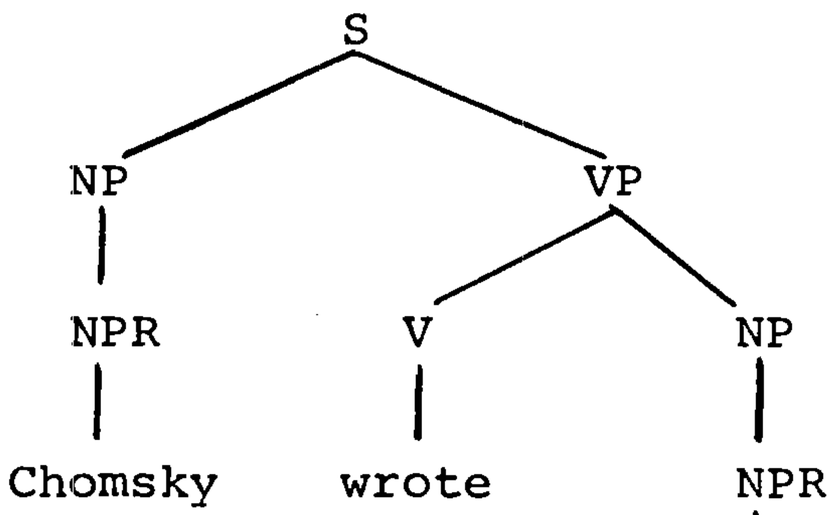
2.2.1 Structure building on the arcs

A parsing system must do more than just say whether or not a given string is a sentence; it must also build up a representation of the syntactic structure of the sentence. Such a representation must exhibit the syntactic relationships among the words and phrases of the sentence. In the augmented transition network model, this is accomplished by the use of structure-building actions on the arcs of the grammar, and by the association of a form with each final state of the grammar which specifies how to build the structural representation to be returned by that state. This form is given by the label on the POP arc associated with that state.

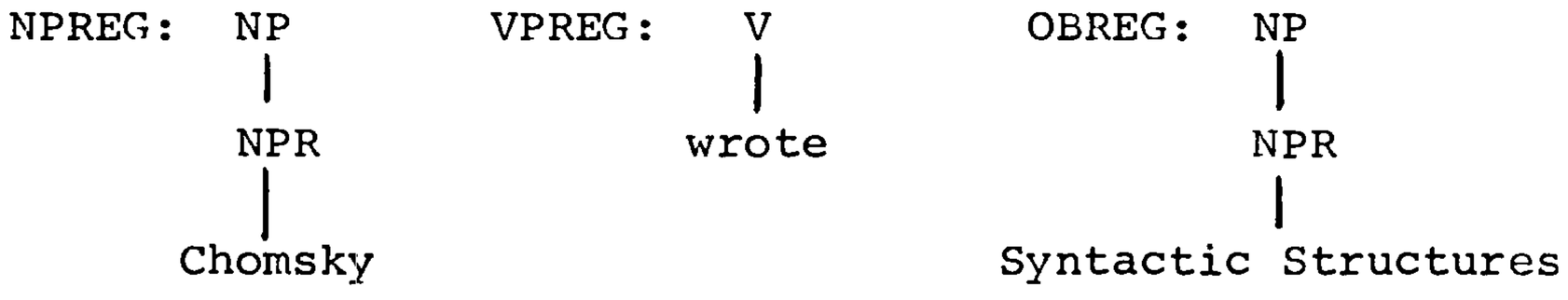
The structure-building actions as well as the arbitrary conditions on the arcs operate on the contents of a set of registers which are maintained at each level of recursive application of the network and are set and reset by the actions on the arcs. A special "current constituent pointer" * is also available for reference in the conditions and actions. The structure-building form associated with the POP arcs uses the contents of these registers to assemble its structural representation. Each register may contain an arbitrary piece of tree structure, and may also be used to hold flags for testing by the conditions on the arcs.

The basic structure building actions is that which attaches the contents of specified existing registers at specially marked points in a prototype tree fragment and puts the result into

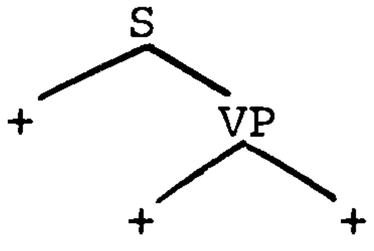
a register. For example, the sentence structure:



can be built by attaching the contents of registers NPREG, VPREG and OBREG:



as leaves of the fragment:



where the + signs indicate leaves that are to be replaced by register contents.

The use of registers to hold pieces of sentence structure allows considerable flexibility in the way that structures are built up. The final structure of a construction does not need to be fixed until the parser is ready to pop up with the total structure of the construction. That is, the decision as to the final structure can be postponed until all of the pieces of the structure have been found in whatever order they occur. The relative order among the

pieces of structure contained in different registers is not decided until the pieces are put together at the end, and this order need not have anything to do with the order in which the pieces were found. Moreover, even when one has made a tentative decision as to the funciton of a particular part of the structure and assigned it to a register accordingly, it is always possible to change one's mind in the light of subsequent input and move that piece of structure to a different register. Nothing about the structure is frozen until the moment that it is popped up to the higher level computation which wanted it.

2.2.2 Parsing with a transition network grammar

A transition network grammar is essentially a non-deterministic machine. That is, the transitions which are permissable from a given state are not uniquely determined by the input string. It is this characteristic of the model which mirrors the notion of ambiguity in English sentences. A sentence is ambiguous if there is more than one possible accepting path for that sentence. There are a number of complexities forced on a natural language parser by the fundamental ambiguity of English, and one of them is the need to provide an algorithm which is capable of pursuing various possible alternatives in the course of parsing. The enumeration of these alternatives is the major source of effort in most natural language parsing systems. While it is not possible to avoid completely this fundamental fact of life for natural language processing, the techniques of the transition network grammar go a long way toward minimizing the problem. The factoring and merging of paths in the network and the postponing of decisions until there is information to make them tend to reduce the total number of alternatives which in principle must be considered. Furthermore, the ordering of the arcs leaving the states permits a selection of the "more likely"

alternatives first so that in many cases, the most likely parsing is found while many of the other alternatives have not yet been pursued. This permits a parsing system in which the parser uses the ordering of the arcs and the complex conditions on the arcs to try to determine the most likely parsing first and thereby avoid a large part of the enumeration required by other parsing algorithms. The basic necessity for dealing with a non-deterministic or enumerative algorithm, however, remains.

2.2.3 Configurations

In simulating the operation of a nondeterministic machine by a deterministic machine such as a real computer, it is necessary to keep track of alternative configurations of the nondeterministic machine. In the case of a transition network grammar a configuration is determined by the current state, the current register contents, and a stack of the states and register contents at all higher levels in the analysis (since in general, the current state may be several levels down in recursive calls to the network). Each recursive call to the network adds another entry to the stack to contain the state and registers associated with that level and then clears the registers for the new level and sets the state to the state which was named on the PUSH arc. In addition, the stack entries remember the actions which remain to be performed on the PUSH arc after a successful return from the PUSH.

In the parsing system which we have implemented, the configuration is represented by a list consisting of the state, the stack, a list of register contents, the contents of a special HOLD list, and a PATH entry which records the history of how the current state was reached from the initial state at the current level. The stack is represented by a list whose elements (STACKELT's) record

the state, the register contents, the actions on the PUSH arc, and the partial path entry for the computations at higher levels. Register contents are kept on a list of alternating register names and register values.

2.2.4 Organization of the parser

Parsing of a sentence begins by calling the function PARSER with a string to be parsed. PARSER constructs an initial configuration consisting of the start state (with empty registers and stack) and then calls a function STEP to simulate the transitions in the network. It calls a function LEXIC to perform the lexical analysis of the input string--determining the next word, accessing its dictionary entry, expanding contractions, compressing compound expressions, making substitutions, etc. Thus PARSER provides the basic overall control, while LEXIC interfaces the input string and STEP performs the basic simulation of the transition network. Flow charts of these basic functions are given in figures 2-3, 2-4, and 2-5.

2.2.5 Simulation of Nondeterminism

Although the parsing system we are using provides for following alternative paths either in series or in parallel or in combinations of the two, the MSC system as we have implemented it makes use only of the sequential mechanism. In this mode, the arcs leaving a state are considered in the order in which they occur, and the first arc which can be followed is chosen. At this point, any arcs remaining in the list, together with the current configuration and the place in the input string, are combined into a list called an ALTARC alternative and saved on an ALTS list of the parser to be persued later if the current choice turns out not to be successful.

PARSER:

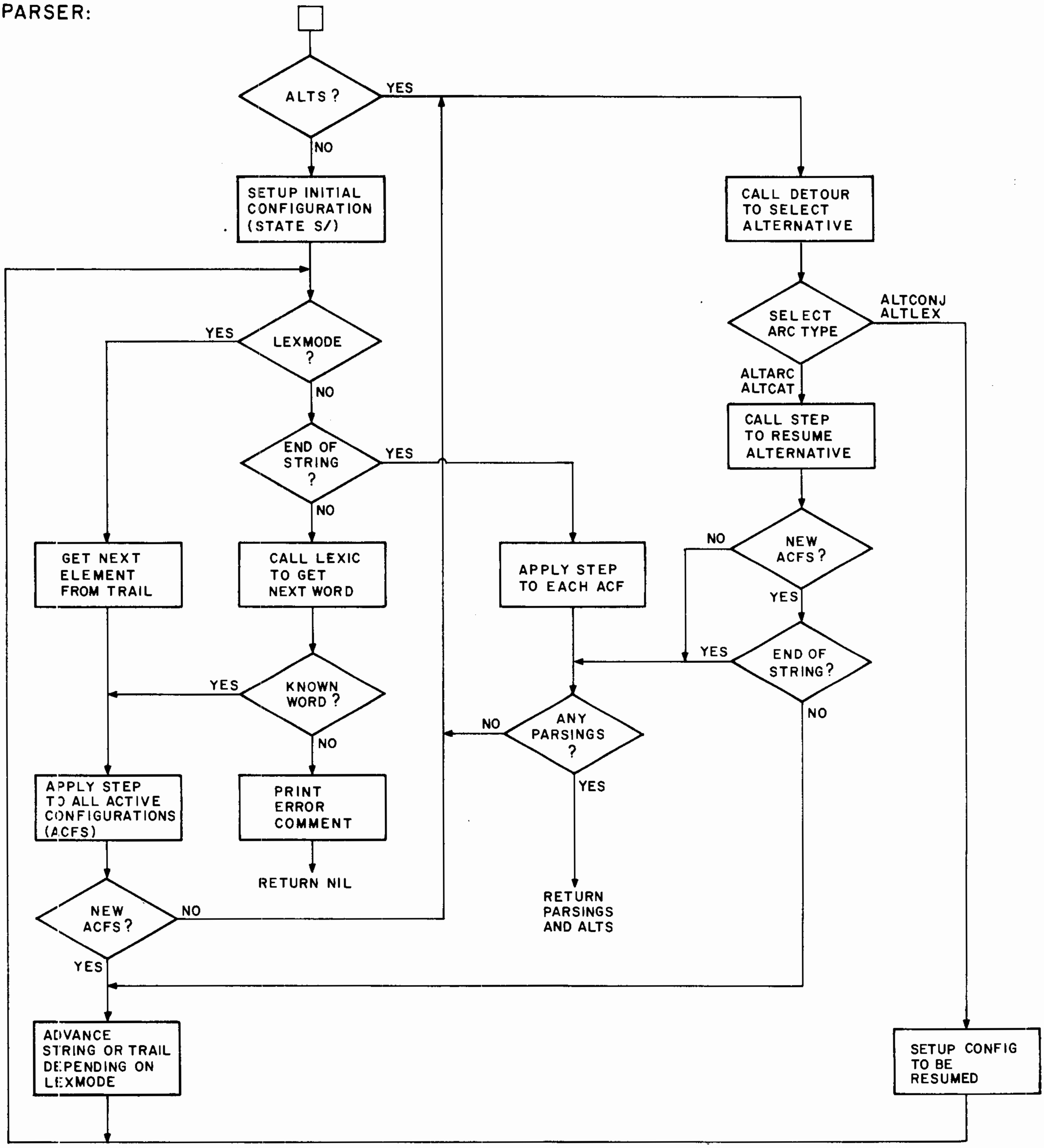


Figure 2-3. The Function PARSER

LEXIC:

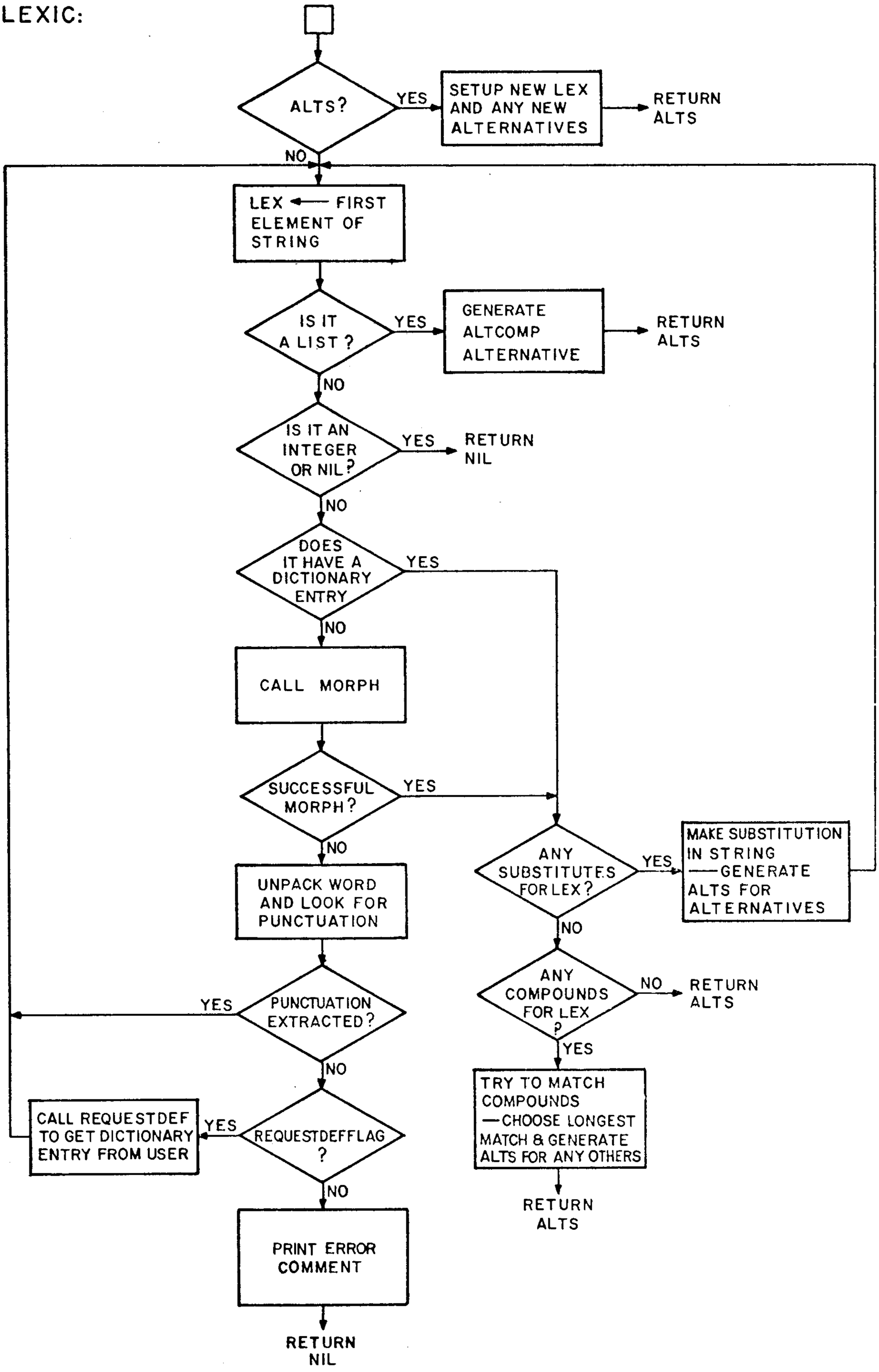


Figure 2-4. The Function LEXIC

STEP:

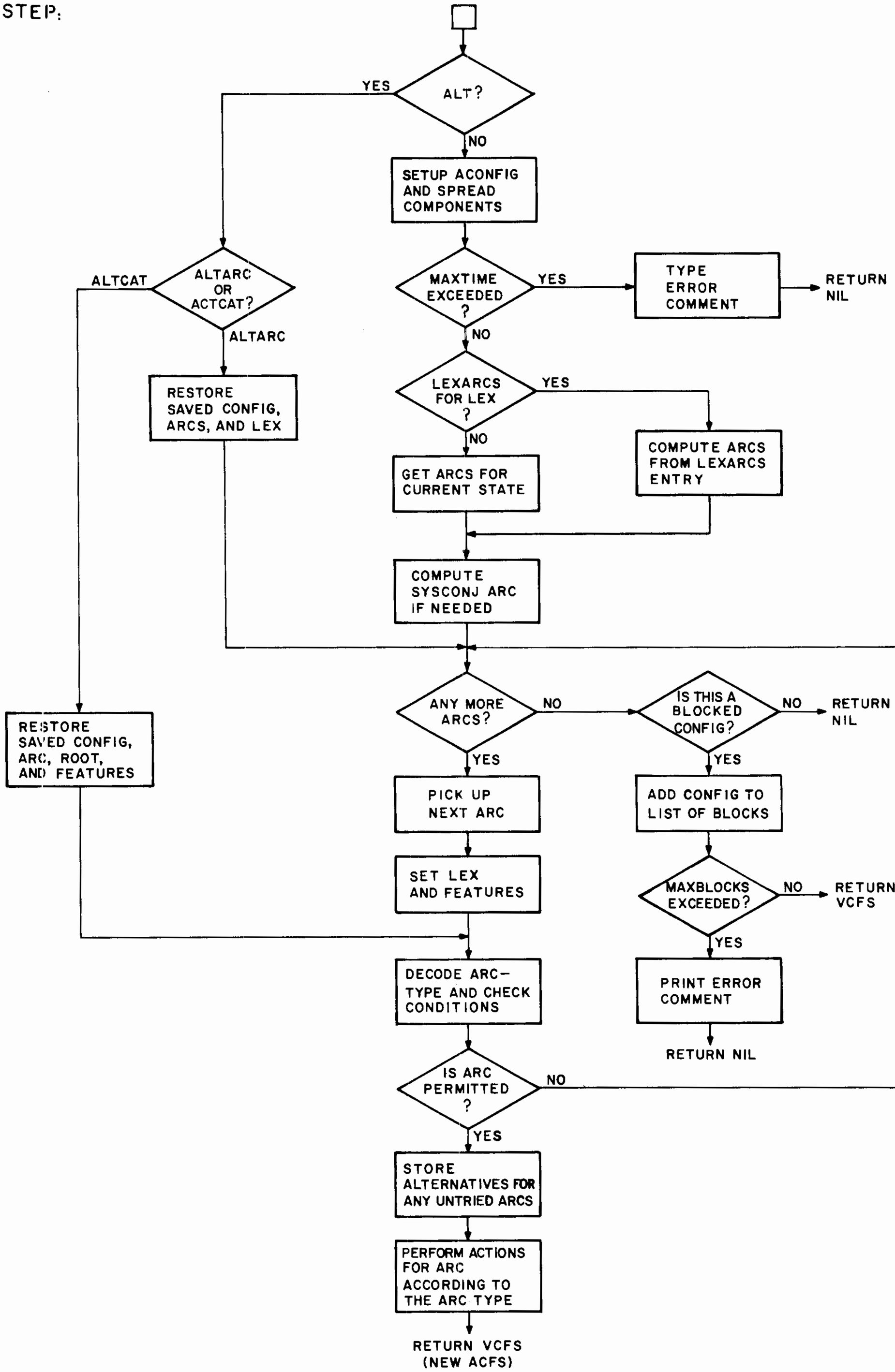


Figure 2-5. The Function STEP

We feel that this means of dealing with nondeterminism preserves the potential for backtracking and trying other alternatives which is essential for dealing with natural language ambiguity while retaining most of the advantages of a deterministic algorithm. It depends for its success, however, on the ability for selecting the right parsing first or soon thereafter, (since otherwise all alternatives have to be enumerated and the advantages are lost). The present grammar does a very good job of selecting a reasonable parsing (if not the best one) in most cases, but there remains one major area in which syntactic information alone (without semantic information) has so far proven insufficient for making good choices for the "most likely" parsing. This area is that of choosing the scope for conjunctions.

2.2.6 Morphological analysis

One of the features of the current English processor is a facility for morphological analysis of regularly inflected nouns and verbs. This facility permits a single dictionary entry for the root form of the word with a code which indicates the type of regular inflection which the word undergoes. The system will then automatically recognize all of the regularly inflected forms of that root. This facility is performed by a function MORPH called by LEXIC. In addition to inflectional analysis, MORPH is able to recognize some items which appear to be contract numbers, hyphenated adjective modifiers, integers, and times of day without their having to be entered in the dictionary. Other types of morphological analysis are possible for "guessing" the parts of speech for words that are unknown to the system, but this type of analysis has not been incorporated into the current system.

2.3 The Semantic Interpreter

The semantic interpretation component of the LSNLIS system is an adaptation of the semantic interpretation procedure presented in Woods (1967, 1968). It operates on a syntactic structure or fragment of syntactic structure which has been constructed by the parser and it assigns semantic interpretations to the nodes of this structure to indicate the "meanings" of those constructions to the system. The procedure is such that the interpretation of nodes can be initiated in any order, but if the interpretation of a node requires the interpretation of a constituent node, then the interpretation of that constituent node is performed before the interpretation of the higher node is completed. Thus, it is possible to perform the entire semantic interpretation by calling for the interpretation of the top node (the sentence as a whole), and this is the normal mode in which the interpreter is operated in the LSNLIS system.

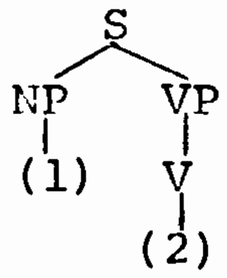
2.3.1 Semantic Rules

In determining the meaning of a construction, two types of information are used--syntactic information about sentence construction and semantic information about constituents. For example, in interpreting the meaning of the sentence, "Chomsky wrote Syntactic Structures," it is both the syntactic structure of the sentence (subject = Chomsky; verb = "write"; object = Syntactic Structures) plus the semantic facts that Chomsky is a person and Syntactic

Structures is a book that determine the interpretation (AUTHOR: SYNTACTIC STRUCTURES CHOMSKY). In the Woods interpretation procedure, this information is embodied in semantic rules consisting of patterns that determine whether a rule can apply, and actions that specify how the semantic interpretation is to be constructed.

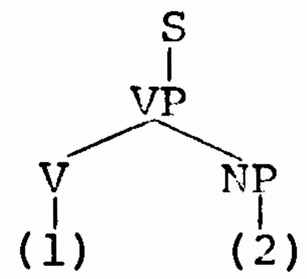
Syntactic information about a construction being interpreted is tested by tree fragments such as those indicated below:

S:NP-V



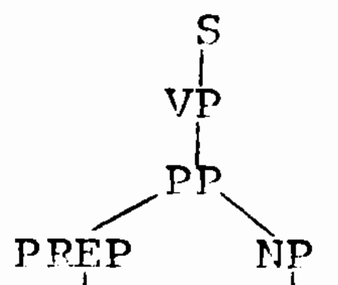
SUBJECT-VERB

S:V-OBJ



VERB-OBJECT

S:PP



PREPOSITION-OBJECT MODIFYING A VP

Fragment S:NP-V matches a sentence if it has a subject and a verb and also associates the numbers 1 and 2 with the subject noun phrase and the verb respectively. The numbered nodes can be referred to for checking semantic conditions and for specifying the interpretation of the construction. Fragments in the system are named mnemonically for readability.

The basic element of the pattern part of a semantic rule is a template consisting of a tree fragment plus additional semantic conditions on the numbered nodes of the fragment. For example, the template (S:NP-V (AND (MEM 1 PERSON) (EQU 2 WRITE))) matches a sentence if its subject is semantically marked as a person and its verb is "write". The pattern part of a rule consists of a limited

Boolean combination of such templates and the action of the rule specifies how the interpretation of the sentence is to be constructed from the interpretations of the numbered nodes of the templates.

The left-hand side of a semantic rule consists of a list of components, each of which may be either a single template, a negated template (embedded in a NOT), or a disjunction (OR) of templates. A component consisting of a simple template matches a node of the syntax tree it its template does, and a NOT component matches a node if its embedded template fails. An OR component matches if any of its constituent templates match (including a possible DEFAULT template at the end which matches if nothing else does). A semantic rule matches a node if all of its components match. In addition, in the process of matching a rule, a record is maintained of the nodes of the syntax tree which match the numbered fragments in each of the components.

2.3.2 Right-hand Sides

The right hand sides (or actions) of semantic rules are forms (or schemata) into which the interpretations of embedded constituents are inserted before the form is evaluated to give the semantic interpretation (or a part of it) which is to be attached to a node. The expressions in the right-hand sides which indicate the places where interpretations of embedded constituents are to be inserted are indicated by lists (called a REF's) which begin with the atom # and contain one or two numbers and an optional "TYPEFLAG". The numbers indicate the node in the tree whose interpretation is to be inserted by naming first the number of a component of the rule and then the number of a node in a tree fragment of that component. Thus the reference (#2 1) represents the interpretation of the node that matches node 1 of 2nd component of the

rule. In addition, the single number ∅ can also be used to reference the current node.

The TYPEFLAG element, if present, indicates how the node is to be interpreted. (For example, in the MSC system there is a distinction between interpreting a node as a topic description and interpreting it for what it says.) Thus (# ∅ TOPIC) represents the interpretation of the current node as a topic description. There are a variety of types of interpretation used for various purposes in the semantic interpretation rules of the system. The absence of a specific TYPEFLAG in a REF indicates that the interpretation is to be done in the normal mode for the type of node which it matches. In this case, there is an alternative form of the REF consisting of a dotted pair of the two numbers. Thus (2 . 1) is equivalent to (# 2 1).

As an example, consider the semantic rule:

```
(S:WRITE
     (S:NP (MEM 1 PERSON))
     (S:V-OBJ (AND (MEM 2 DOCUMENT) (EQU 1 WRITE)))
          -->(PRED (AUTHOR: (# 2 2) (# 1 1))))
```

This rule says that if the sentence has a subject which is a person, a verb "write", and an object which is a document, then the meaning of the sentence is computed by substituting the interpretations of the node numbered 1 in the first component (# 1 1) and the node numbered 2 in the second component (# 2 2) into the indicated places in the schema (AUTHOR (# 2 2) (# 1 1)) and treating it as a predicate (PRED). (S:WRITE is the name of the rule.)

2.3.3 Organization of Rules

The semantic rules for interpreting sentences are usually governed by the verb of the sentence. That is to say that out of the entire set of semantic rules, only a relatively small number of them can possibly apply to a given sentence because of the verb mentioned in the rule. For this reason, the semantic rules can be indexed according to the verb (or verbs) of sentences to which they could apply and recorded in the dictionary entry for the verb. Each rule then characterizes a syntactic/semantic environment in which the verb can occur and specifies its interpretation in that environment. The templates of the rule thus describe the necessary and sufficient constituents and semantic restrictions in order for the verb to be meaningful. There are also situations, however, in which the type of construction and the mode in which it is being interpreted determine a set of rules which does not depend on the head of the construction.

2.3.4 Multiple Matches

Since the templates of a rule may match a node in several ways, and since several rules may simultaneously match a single node, it is necessary to indicate how the interpretation of a node is to be constructed in such a case. To provide this information, the lists of rules which the interpreter uses--whether taken from global lists or from the property lists of heads of constructions--are not necessarily simple lists of rules, but may be organized into rule groups with each group indicating how (or whether) simultaneous matches by different rules are to be combined. In addition, at the top level of such lists, the atom NIL may be used as a "barrier" to indicate that by the time the matching process has reached that point in the list it will proceed further only if there have been no successful matches so far.

The mode for combining simultaneous matches at the top level of this list is a default mode determined by TYPEFLAG and the type of node. Possible modes are SPLIT (which keeps multiple matches separate as semantic ambiguities), FAIL (which prohibits multiple matches), AND (which combines multiple matches with an AND), and OR (which combines multiple matches with an OR). For example, a rule list of the form (A B NIL C (OR D E)) with default mode AND indicates that if either of the rules A or B is successful, then no further matches are tried (NIL is a barrier); otherwise, rules C, D, and E are tried, and if both D and E match then the results are OR'ed together, and if C matches together with D or E or both, it is AND'ed to the results of the OR group.

The modes (SPLIT, FAIL, AND, and OR) also apply to multiple matches of a single rule. A rule may either specify the mode for multiple matches as its first element prior to the list of components, or else it will be governed by the rule group mode setting at the time it is matched.

2.3.5 Organization of the Semantic Interpreter

The overall operation of the semantic interpreter is as follows: A top level routine calls the recursive function INTERP with TYPEFLAG NIL looking at the top level of the parse tree. Thereafter, INTERP attempts to match semantic rules against the specified node of the tree, and the right-hand sides of matching rules specify the interpretation to be given to the nodes. The possibility of semantic ambiguity is recognized, and therefore the routine INTERP produces a list of possible interpretations (usually a singleton, however). Each interpretation consists of two parts--a node interpretation (called the SEM of the node) and a quantifier "collar" (called the QUANT of the node) which is to be returned to

the routine which called for the semantic interpretation of the current node. Thus the result of a call to INTERP for a given node P is a list of SEM-QUANT (or S-Q) pairs--one for each possible interpretation of the node.

INTERP then calls a function HEAD to determine the head of the construction which it is interpreting and a function RULES to determine the list of semantic rules (depending on the type of node and the value of TYPEFLAG) which it is to use to interpret the construction. It then dispatches control to a routine MATCHER. If no interpretations are found, then, depending on the TYPEFLAG and various mode settings, INTERP either returns a default interpretation T, goes into a break with a comment that the node is uninterpretable (permitting a systems programmar to debug rules), or returns NIL indicating that the node has no interpretations for the indicated TYPEFLAG.

The function MATCHER calls a function MATCHGROUP to match groups of semantic rules, and MATCHGROUP, in turn, calls the function RMATCH to match single rules. RMATCH calls the function TEMPMATCH to match templates in the left-hand side of the rule and SEMSUB to insert the interpretations of constituents into the right-hand side of the rule and compute the resulting interpretation. The relationships among these functions is indicated by the diagram in Figure 2-6, and flowcharts for the routines INTERP and RMATCH are given in figures 2-7 and 2-8.

Figure 2-6. Subroutine control map for the routine INTERP

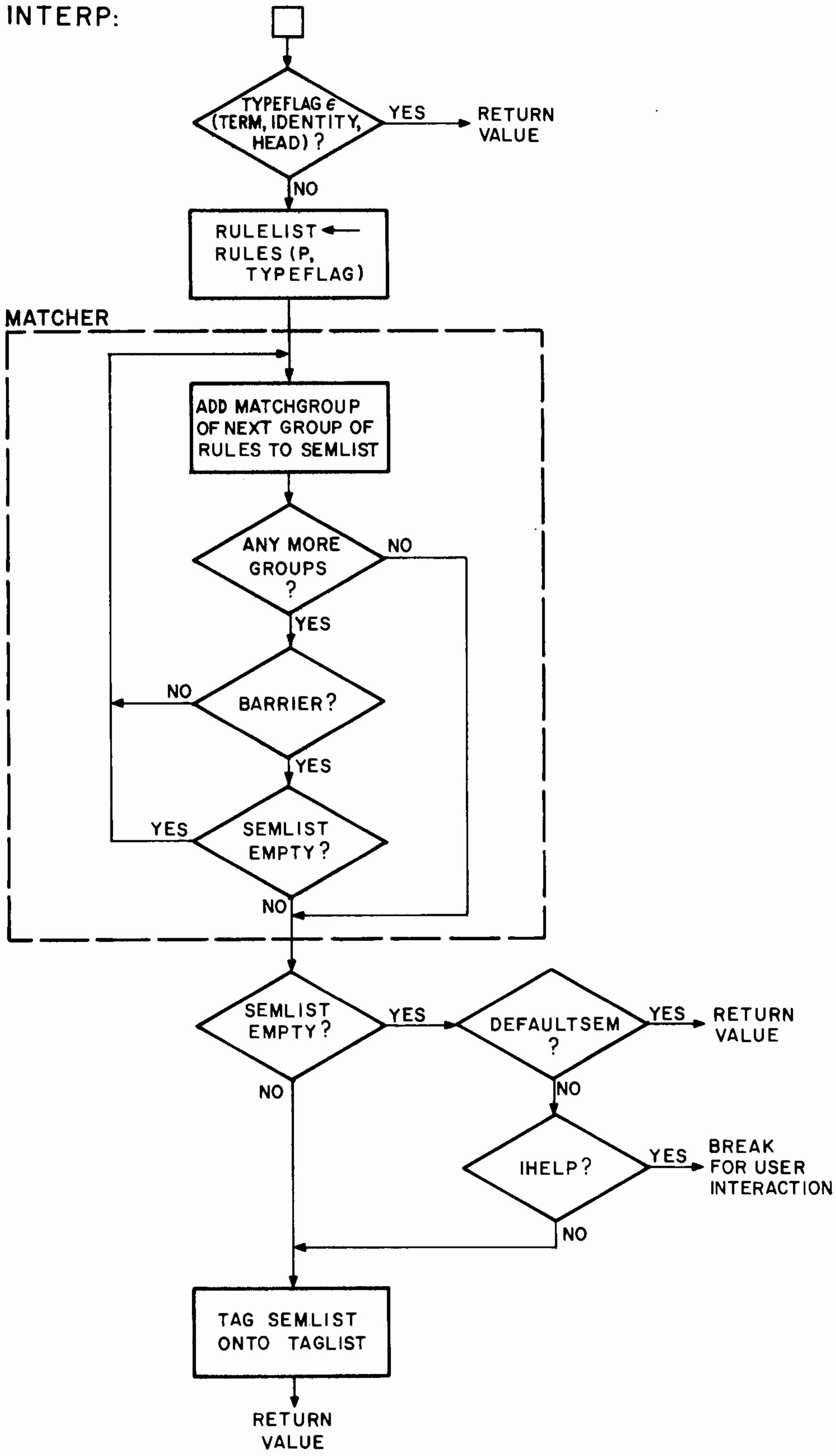


Figure 2-7. The Function INTERP

RMATCH:

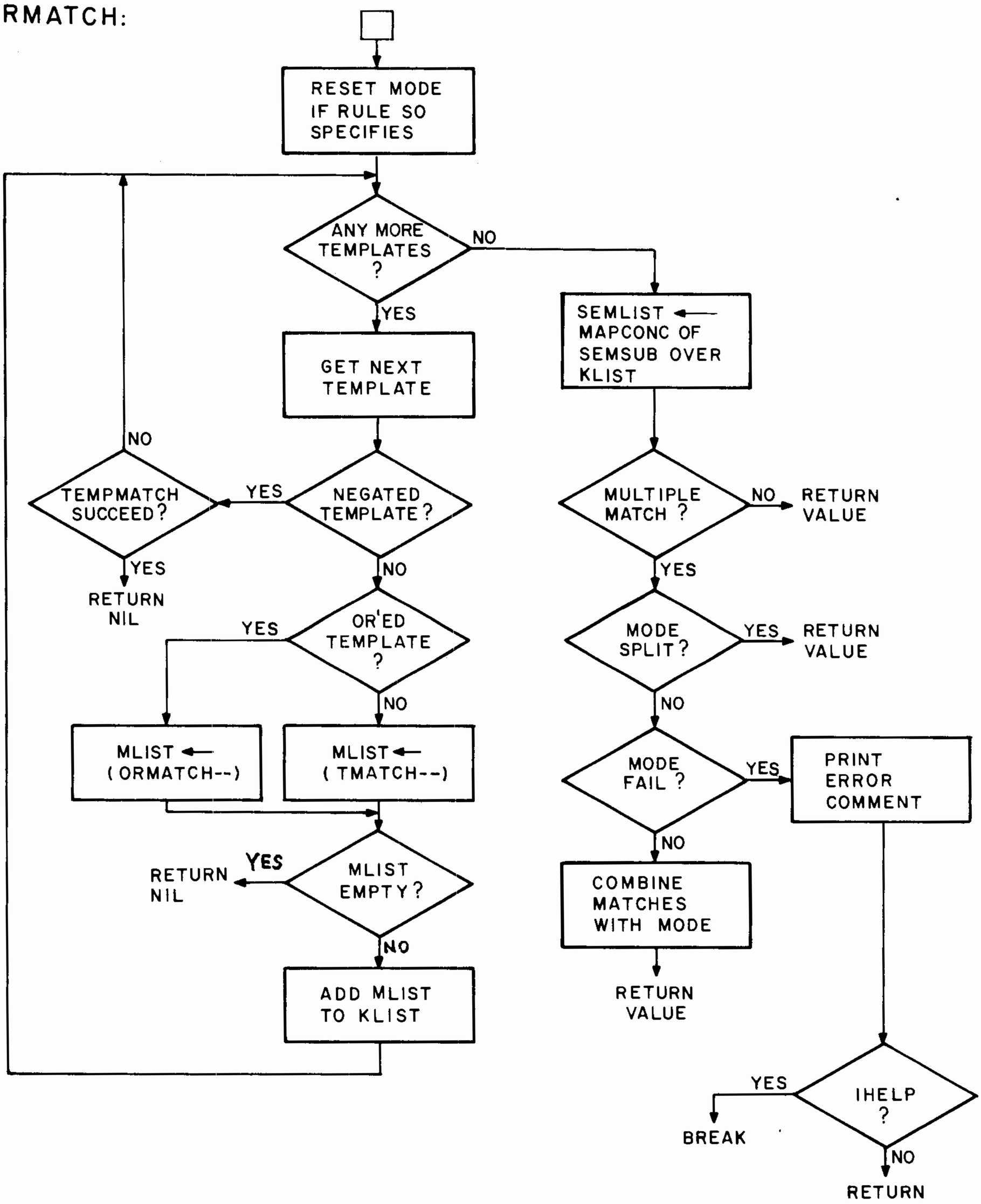


Figure 2-8. The Function RMATCH

2.3.6 An Example

As an example of the operation of the semantic interpreter consider the sentence:

(HOW MANY SAMPLES ARE THERE?)

which has the following syntactic structure assigned to it by the grammar:

```
S  Q
     NP  DET  HOWMANY
         N  SAMPLE
         NU  PL
   AUX  TNS  PRESENT
   VP  V  EXIST      .
```

Semantic interpretation begins with a call to INTERP looking at the topmost S node with typeflag NIL. The head of the construction is the verb EXIST, and the function RULES looking at an S node with typeflag NIL returns the global list of rules PRERULES. These rules look for such things as yes/no question markers, sentential negations, etc. In this case, a rule PR6 matches and the right-hand side (PRED (# 0 SRULES)) specifies a call to INTERP for the same node with typeflag SRULES.

RULES looking at an S node with typeflag SRULES returns a list of semantic rules which it gets from the dictionary entry for the head of the sentence (in this case EXIST), and in this case a rule SS41 matches. Its right-hand side (PRED (EXIST (1 . 1))) specifies a pattern into which the interpretation of the node (1 . 1) is to be inserted (where the matching node in question is the subject noun phrase).

The semantic interpreter now begins to look at the subject noun phrase with typeflag NIL. In this case, RULES is smart enough to detect the HOWMANY determiner and return the single rule D:HOWMANY, which matches successfully. The right-hand side of D:HOWMANY is:

```
(QUANT (FOR THE X / (# 0 NUMBER) : T ; (PRINTOUT X)))
```

which specifies that a quantifier is to be constructed by substituting in the indicated place the interpretation of this same node with typeflag NUMBER. (In the case of howmany questions, the rule assumes that the syntactic structure above containes only the dummy verb EXIST and therefore leaves no opening in the quantifier for the later insertion of the higher proposition.

RULES with typeflag NUMBER returns only the single rule D:NUMBER whose right-hand side is:

(SSUNIONF (SEQL (NUMBER X / (# 0 NRULES) : (# 0 RRULES)))).

Here, the function SSUNIONF is a function which can grab quantifiers like PRED but which would insert them inside the NUMBER function instead of around the outside. SEQL is an enumeration function which will show up in the final interpretation. This rule calls for the interpretation of the NP node with typeflags NRULES and RRULES which end up returning (SEQ SAMPLES) and the default restriction T, respectively, with no quantifiers arising from either source. After the insertion of these values and the evaluation of SSUNIONF the result of this call to INTERP is (SEQL (NUMBER X15 / (SEQ SAMPLES) : T)) with no additional quantifier.

The right-hand side of the rule D:HOWMANY now gets resumed and after substitution and evaluation of the QUANT, the resulting SEM is X15 with an associated QUANT of:

(FOR THE X15 / (SEQL (NUMBER X15 / (SEQ SAMPLES) : T)): T ;
(PRINTOUT X15))

(Normally such a QUANT would contain a marker DLT indicating the place where the interpretation of the higher sentence was to be inserted, but because of the special nature of the howmany determiner this quantifier is completely self-contained.) This quantifier is returned to the higher level S interpreter whose PRED grabs it, and from there it ripples its way to the top where it becomes the final interpretation.

2.4 The Retrieval Component

2.4.1 The Function Execute

In the NASA LSNLIS, the retrieval component resides in a separate fork of the TENEX time-sharing system which we will call the lower fork or retrieval fork. This fork is under the control of the language processing fork.

When the semantic interpretation component has finished constructing the interpretation of a request, it calls the function EXECUTE with this interpretation as its argument. The function EXECUTE passes the interpretation to the retrieval fork by means of a buffer file QBUF (for query buffer) and wakes up the retrieval fork. When the retrieval fork has completed processing the query, it will have written the answer(s) onto a file HITFILE, and it will then write the number of hits into a buffer file ABUF and return control to the upper fork. The function EXECUTE then prints out the answer if there are fewer than 5 hits, or notifies the user of the number of hits otherwise and asks him whether he wishes to see the answers. The function EXECUTE, thus serves as the access port to the lower fork.

2.4.2 The Data Base Tables

The Data Base of the system consists of two types of information--chemical analysis data on the lunar samples, and keyphrase indexing of the publications concerning the samples. Examples of these two types of data are given in figures 2-9 and 2-10.

; MINTABLE.;6 THU 7-JAN-71 10:21AM

SAMPLE	PHASE	CONSTIT.	CONTENT	UNIT	CITATION	TAG
S10002	OVERALL	AL26	120.0	DPM/KG	D70-237	0
		BE7	80.0			
		C	190.0	PPM	D70-228	0
			230.0		D70-234	0
		CO56	40.0	DPM/KG	D70-237	0
		C13	8.7999999	DEL	D70-228	0
		H	.83999999	CC/G	D70-249	0
		H3	314.0	DPM/KG		
		K20	.13250999	PCT	D70-237	0
		MN54	28.0	DPM/KG		
		N	125.0	PPM	D70-234	0
		NA22	51.0	DPM/KG	D70-237	0
		S	.10699999	PCT	D70-228	0
		SC46	8.0	DPM/KG	D70-237	0
		S34	3.5	DEL	D70-228	0
		TH	1.9200000	PPM	D70-237	0
		TI44	2.5	DPM/KG		
		U	.48999999	PPM		
S10003	OVERALL	AL203	10.429999	PCT	D70-205	0
			9.6364499		D70-208	0
			10.203300		D70-216	0
			11.0		D70-244	0
		AL26	74.0	DPM/KG	D70-237	0
			75.0		D70-241	0
			74.0		D70-260	0
		BA	160.0	PPM	D70-203	0
			106.0		D70-215	0
			220.0		D70-216	0
		BE	1.5		D70-203	0
		BE7	100.0	DPM/KG	D70-237	0
		CAO	11.129999	PCT	D70-205	0
			11.613360		D70-216	0
			11.0		D70-244	0
		CE	45.5	PPM	D70-215	0
			37.0		D70-216	0
			41.300000		D70-220	0
		CO	15.0		D70-203	0
			14.100000		D70-216	0
		CO56	43.0	DPM/KG	D70-237	0
		CO57	43.0		D70-241	0
		CO60	1.0			
		CPX	50.199999	***	D70-154	0
			51.699999		D70-173	0
		CR203	.27181999	PCT	D70-203	0
			.20312999		D70-216	0
			.25999999		D70-244	0

Figure 2-9. A Sample of the Chemical Analysis Data

```
;  PHRTABLE.;1    WED 16-DEC-70 2:01PM

((ABRASION) (D70-086 D70-096))
((ABSORBED GAMMA RADIATION) (D70-097))
((ABSORBED GAS) (D70-129))
((ABSORPTION) (D70-022 D70-068 D70-071 D70-072 D70-097 D70-099 D70-107
D70-108 D70-117 D70-122 D70-126 D70-131))
((ABSORPTION BAND) (D70-068 D70-108 D70-115 D70-122 D70-126 D70-135))
((ABSORPTION COEFFICIENT) (D70-071 D70-117))
((ABSORPTION COEFFICIENT MEASUREMENT) (D70-011 D70-117))
((ABSORPTION PEAK) (D70-107))
((ABSORPTION SPECTROMETRY) (D70-136))
((ABSORPTION SPECTRUM) (D70-126 D70-131 D70-135))
((ABUNDANCE ANOMALY) (D70-020))
((ABYSSAL BASALT) (D70-020 D70-027))
((ABYSSAL SUBALKALINE BASALT) (D70-027))
((ACCELERATING POTENTIAL) (D70-057))
((ACCESSORY CHROMITE) (D70-086))
((ACCESSORY ILMENITE) (D70-086 D70-096))
((ACCESSORY OLIVINE) (D70-083))
((ACCESSORY PHASE) (D70-062 D70-071))
((ACCRETION) (D70-010 D70-012 D70-055 D70-087 D70-127))
((ACCRETION STAGE) (D70-012 D70-017 D70-066 D70-087))
((ACCRETIONARY LAPILLI) (D70-028 D70-085))
((ACCRETIONARY RIM) (D70-085))
((ACCRETIONARY STRATIGRAPHY) (D70-087))
((ACCUMULATION SEQUENCE) (D70-085))
((ACHONDRITE) (D70-005 D70-006 D70-008 D70-012 D70-015 D70-018 D70-022
D70-023 D70-024 D70-027 D70-029 D70-032 D70-049 D70-063 D70-066 D70-122))
((ACHONDRITE METEORITE) (D70-012 D70-014))
((ACICULAR CRYSTAL MODE) (D70-071))
((ACICULAR ILMENITE) (D70-060))
((ACICULAR PLAGIOCLASE) (D70-060))
((ACICULAR SILICON) (D70-060))
((ACID BASALT) (D70-069))
((ACID GLASS) (D70-057))
((ACID HYDROLYZATE) (D70-135))
((ACID LEACHING) (D70-054))
((ACID SOLUTION) (D70-132 D70-139 D70-140))
((ACIDIC GLASS) (D70-069))
((ACTINIUM) (D70-021 D70-088))
((ACTIVATION CROSS SECTION) (D70-127))
```

Figure 2-10. A Sample of the Keyphrase Indexing Data

In the system, these two files are stored in different ways. The keyphrase information is stored symbolically on a disk file in essentially the form that appears in figure 2-10. Keyphrases are looked up with a binary search of this file in order to obtain the associated list of references. The chemical analysis data on the other hand is stored in a compressed, bit-coded form on a binary file which is windowed into an array in the virtual core memory by the hardware page-mapping facilities of TENEX. The use of this type of coding provides an extremely compact and rapidly accessible representation. Detailed descriptions of the data structures and retrieval functions of the data base are included in Appendix F.

2.4.3 The Formal Query Language

The data base of the LSNLIS system is accessed by means of a formal query language into which the input English requests are translated by the language analysis component. Examples of this language have already been seen in previous sections. The language is essentially a generalization of the predicate calculus which could either be manipulated as a symbolic expression by a formal theorem prover to derive intensional inferences or be executed directly on the data base to derive extensional inferences. Only the latter, extensional inference facility is used in the current LSNLIS.

The query language contains essentially three kinds of constructions:

designators, which name objects or classes of objects in the data base (including functionally determined objects),

propositions, which are formed from predicates with designators for arguments, and

commands, which take arguments and initiate actions.

For example, S10046 is a designator for a particular sample, OLIV is a designator for a certain mineral (Olivine), and (CONTAIN S10046 OLIV) is a proposition formed by substituting designators as arguments to the predicate CONTAIN. TEST is a command function for testing the truth value of a proposition. Thus, (TEST (CONTAIN S10046 OLIV)) will answer yes or no depending on whether sample S10046 contains Olivine. Similarly, PRINTOUT is a command function which prints out a representation for a designator given as its argument.

The major power and usefulness of the formal query language comes from the use of a quantifier function FOR and special enumeration functions for classes of data base objects to carry out extensional quantification over the data base. The format for a quantified proposition is:

(FOR QUANT X / CLASS : PX ; QX)

where QUANT is a type of quantifier (EACH, EVERY, SOME, THE, numerical quantifiers, comparative quantifiers, etc.), X is a variable of quantification, CLASS determines the class of

objects over which quantification is to range, PX specifies a restriction on the range, and QX is the proposition or command being quantified. (Both PX and QX may themselves be quantified expressions.)

The specification of the CLASS over which quantification is to range is performed in the system by special enumeration functions which (in addition to whatever other parameters they might have) take a running index argument which is used as a restart pointer to keep track of the state of the enumeration. Whenever FOR calls an enumeration function for a member of the class, it gives it a restart pointer (initially NIL) and each time the enumeration function returns a value it also returns a new restart pointer to be used to get the next member. Enumeration can terminate either by returning NIL indicating that there are no more members or by returning a value and a NIL restart pointer indicating that the current value is the last one. (This latter can save one extra call to the enumeration function if the information is available at the time the last value is returned--e.g. for single valued functions.)

The enumeration function formulation of the quantifier problem frees the FOR function from explicit dependence on the structure of the data base--the values returned by the enumeration function may be searched for in tables, computed dynamically, or merely successively accessed from a precomputed list. A general purpose enumeration function SEQ can be used to enumerate any precomputed list, and a similar function SEQL can be used

to enumerate singletons. For example:

(FOR EVERY X1 / (SEQ TYPECS) : T ; (PRINTOUT X1))

is an expression which will printout the sample numbers for all of the samples which are type C rocks (i.e. breccias).

The bread and butter enumeration function for the chemical analysis data base is the function DATALINE which takes as arguments designators for a data file, a sample, a phase name, and a constituent and enumerates the lines of the data file which deal with the indicated sample/phase/constituent triple. Other complex enumeration functions are NUMBER and AVERAGE which take an argument format similar to the FOR function and perform counting and averaging functions. Detailed descriptions of these and other retrieval functions are given in Appendix F and examples of the interpretations of various requests are given in Appendix G.

Chapter 3

THE GRAMMAR

The translation of an English request into an appropriate retrieval expression proceeds in three main stages: first, the English sentence is converted into a "canonical form" in which the syntactic relationships holding between constituents are made explicit; next, the canonical form, or parse, is mapped into a semantic interpretation which highlights the logical connections between terms; and finally, the semantic interpretation is executed in the data base to produce the answer to the query. In this section we discuss in some detail the first stage of the translation process, the parsing of the input string of English words.

3.1 MOTIVATION AND OVERVIEW

It is a well-known fact about natural languages that sentences which have different words in different orders can have essentially the same meanings, while superficially similar sentences can have very different meaning; this insight is the cornerstone of the transformational theory of grammar (Chomsky, 1957, 1965). For example, active sentences (1) have corresponding passive sentences (2) which are virtually synonymous, and sentences with existential

"there" subjects (3) are synonymous to sentences with ordinary subjects (4):

(1) We need some information.

(2) Some information is needed by us.

(3) There are many documents in the file.

(4) Many documents are in the file.

On the other hand, sentences (5) and (6) have similar sequences of "parts-of-speech", but the syntactic and logical relationships between the words are different:

(5) John is eager to please.

(6) John is easy to please.

In (5), John is to do the pleasing, whereas in (6) John is to be pleased by someone. Linguists account for these facts by positing a form of syntactic description more abstract than just a specification of the linear arrangements of words in sentences. In brief, transformational grammarians characterize the syntactic relationships in a sentence in terms of a "deep structure" -- a structural description, usually in the form of a tree with labeled nodes, from which the string of words can be derived by applying a sequence of formal rules called transformations (Chomsky, 1965). Thus the similarity in meaning of actives and their corresponding passives is due to the fact that the sentences have the same deep structure; the different strings result from the application of slightly different sequences of transformations. Sentences with similar superficial

characteristics but different syntactic relationships, such as (5) and (6), result from the application to different deep structures of transformation sequences with similar outputs. Finally, ambiguous sentences can be derived from more than one deep structure, while ungrammatical word-strings have no corresponding deep structure. In this context, the major task in the syntactic analysis of an input request may be seen as the problem of determining the appropriate deep structure (s) for a given string of English words.

Unfortunately, linguistic theory is not of much help here. Transformational grammars are designed to enumerate the class of possible deep structures (by using a context-free phase-structure grammar) and then to generate all and only the sentences of a language from the set of deep structures. Very little has been said about how to find the deep structure for a given string, and, in fact, the few attempts at "reversing" the sequence of transformational operations that have been made have not been very successful (cf. Petrick, 1965; Zwicky, et al. 1965). Thus, for the syntactic analysis component of the LSNLIS English processor we have used an augmented recursive transition network parser, described elsewhere in this report and in Woods (1970), which surmounts many of the difficulties encountered in earlier efforts at transformational recognition. Our goal is still to map the

input request into a deep-structure-like representation.

A transition network grammar consists of a set of states connected by a set of labelled directed arcs. The label on an arc determines whether the transition can be made, based on the current input word and the previous analysis history of the input string, and also specifies a set of actions to be executed if the transition is permitted (section 2.2 of this report gives a detailed description of the grammatical notation and the operation of the parser). The sequence of transitions taken in the course of an analysis reflects the superficial arrangement of words in the string; the actions on the arcs are used to build up sections of the deep-structure tree and hold them in "registers" until they are combined into larger sections and, eventually, into the complete representation of the input string. Analysis of the input string thus proceeds from left to right on two related levels: words in the string are identified by arc transitions, and at the same time, the deep-structure is being fashioned in the registers.

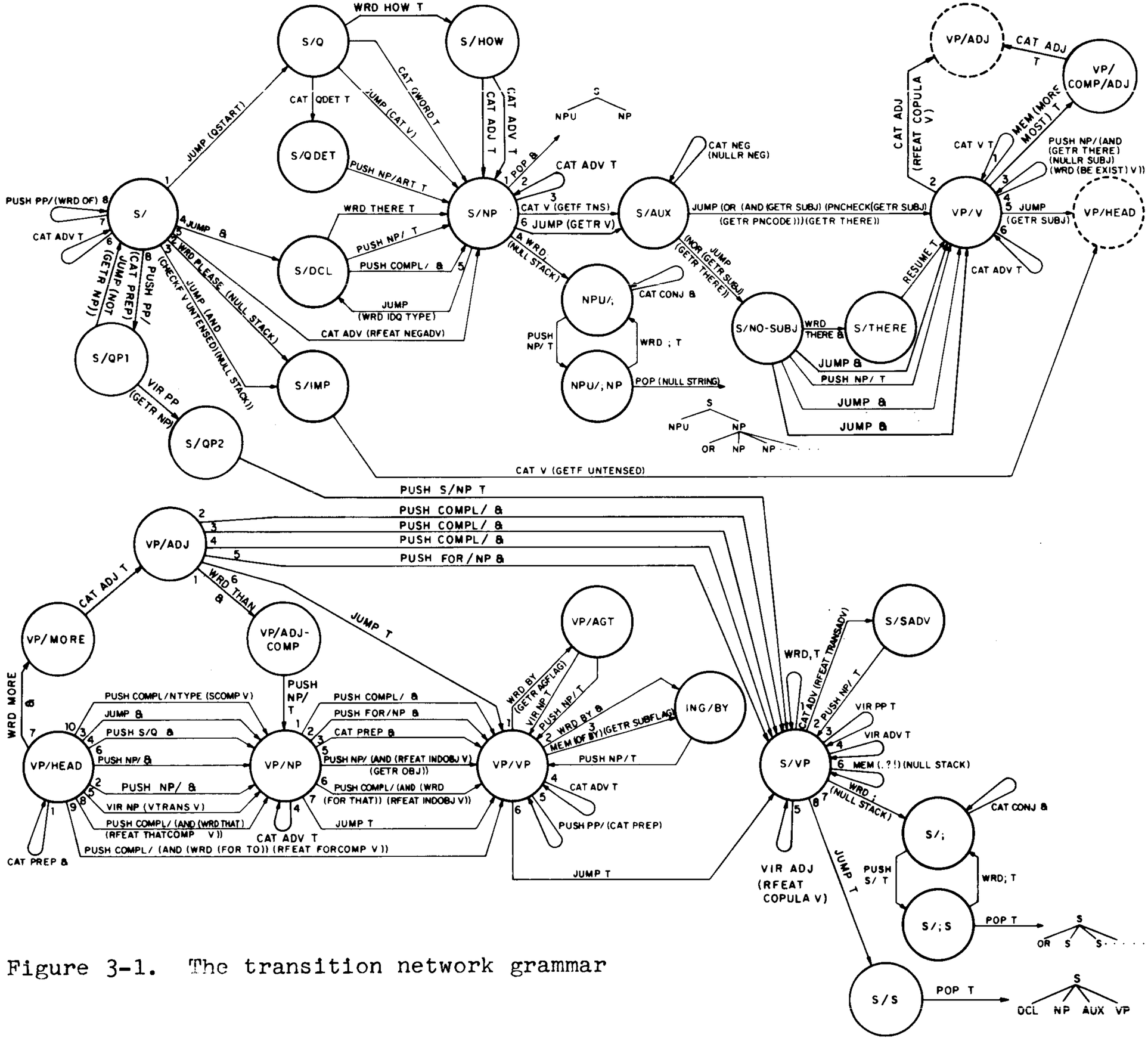


Figure 3-1. The transition network grammar

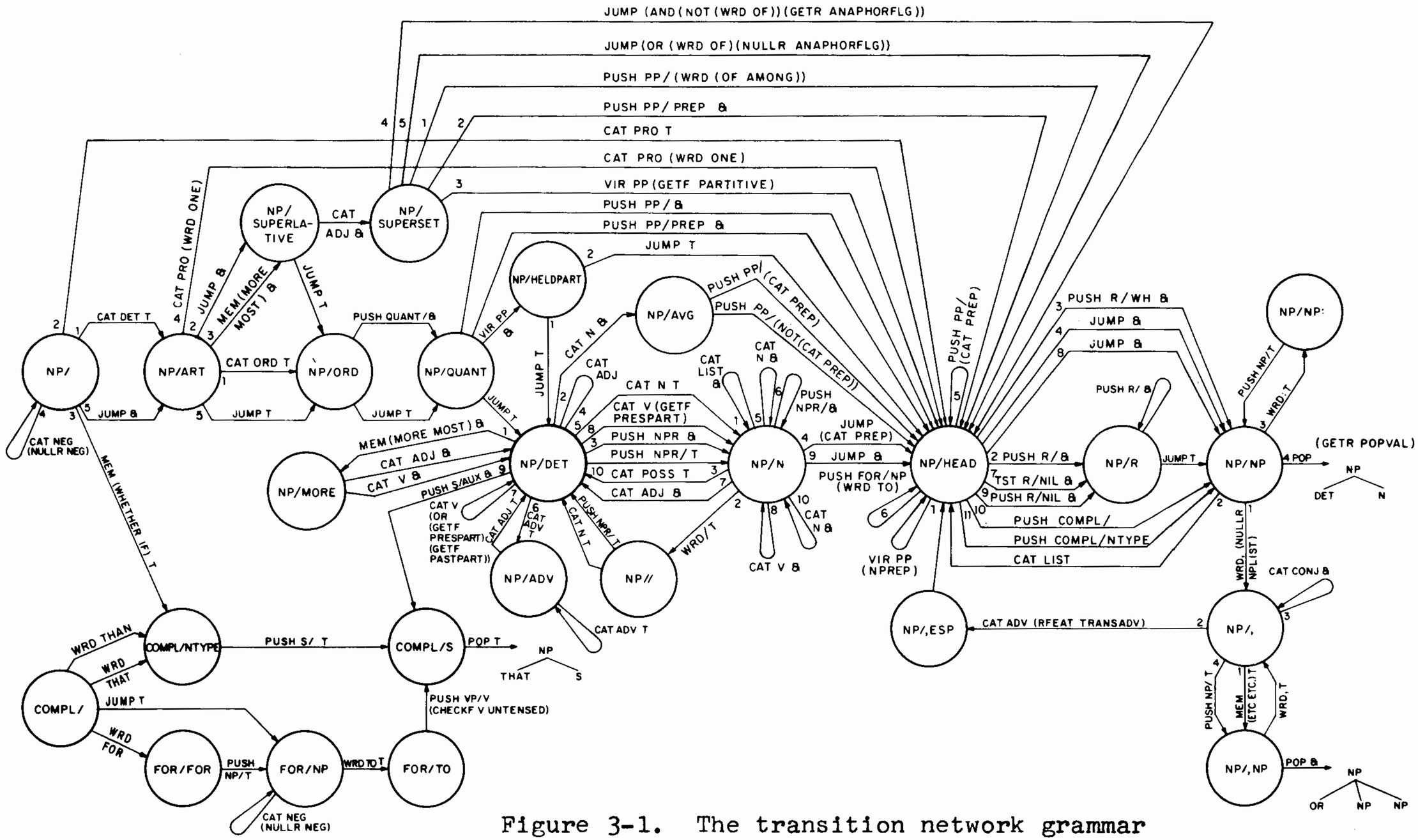
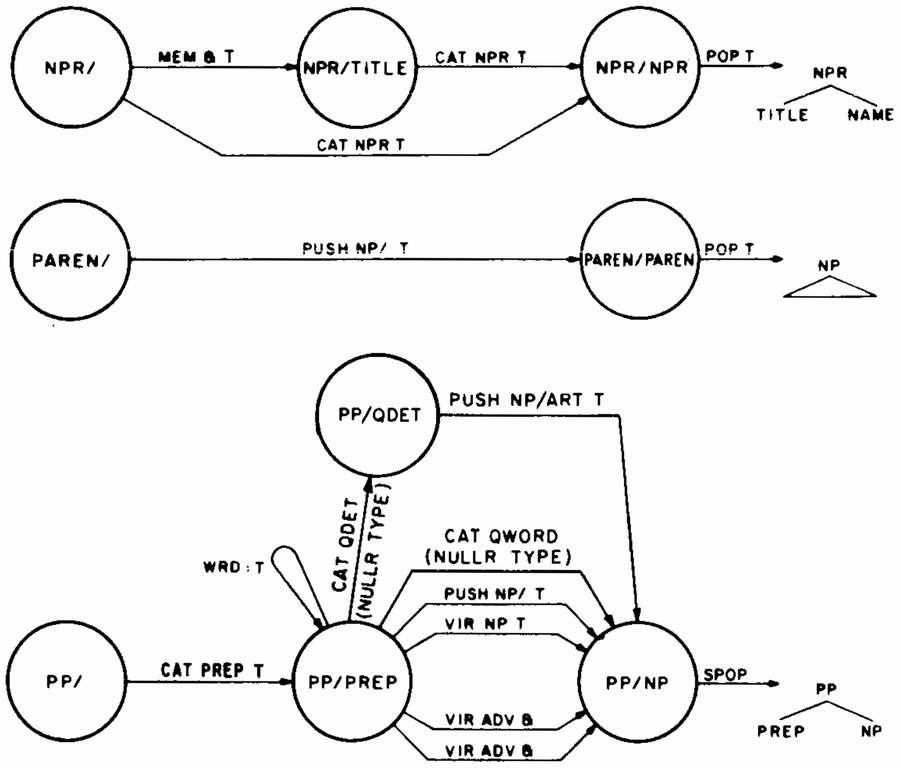


Figure 3-1. The transition network grammar (continued)



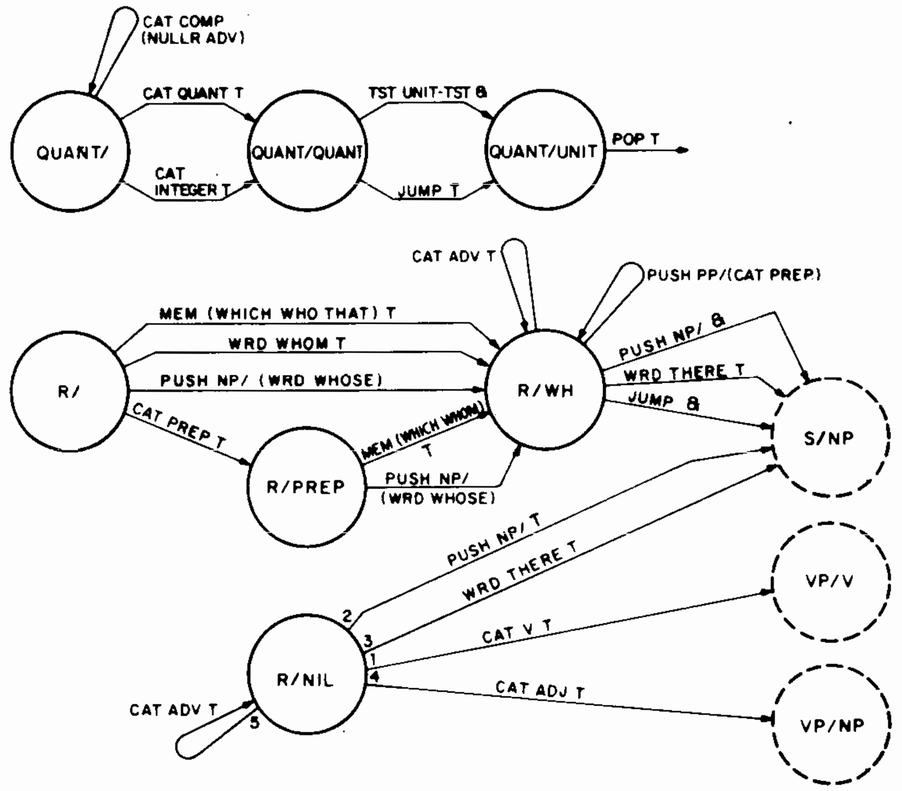


Figure 3-1. The transition network grammar (continued)

The configuration of arcs and states in the grammar is shown in Figure 3-1. Unless the order of the arcs is explicitly indicated by numbers on the arcs, they are ordered clockwise from the top of the state. The symbol & on an arc indicates that there is a condition associated with the arc which is not included in the figure. See the grammar listing in Appendix B for the details of these conditions.

The parses developed by the grammar resemble the deep-structures described by Chomsky (1965), with some elements borrowed from Stockwell et al. (1968) and some included because of special characteristics of the lunar sample requests. Briefly, a sentence consists of a subject noun-phrase, an auxilliary-verb constituent specifying the tense, modality, and aspect of the sentence, and a verb-phrase containing the main verb, the direct and indirect objects and predicate complements (if any), and optional adverbial and prepositional-phrase modifiers. A noun-phrase consists of an optional determiner and adjectival modifiers, a head noun, and optional restrictive and non-restrictive post-nominal modifiers. A precise specification of the form of deep-structures is contained in the listings of the grammar actions SBUILD, NPBUILD, and DETBUILD in Appendix D and in the annotated listing of the grammar itself (Appendix B). Below we will focus on the grammatical strategies used to identify the various

constituents and not on the structures in which they are placed.

3.2 GENERAL DESCRIPTION OF THE GRAMMATICAL STRATEGIES

In this section we discuss and illustrate the overall organization of the grammar and indicate in some detail the strategies used to deal with particular syntactic constructions. A bird's eye view of the grammar was given in Figure 3-1, in which states are represented by circles enclosing the state name (for example, S/DCL) and arcs are represented by arrows connecting the states. The arcs are labelled in the diagram with their types (CAT, WRD, MEM, VIR, JUMP etc.) and frequently, with their conditions also. The actions on the arcs and the detailed specification of ccmplicated conditions may be found in the annotated listing in Appendix B.

The parser allows state names to be arbitrary LISP atoms, but we have adopted the convention that state-names indicate the unit of the sentence being analyzed and constituents of the unit already identified, separated by a slash ("/"). Thus S/AUX signifies that the S-level of the parse is being developed and that we have succeeded either in finding an auxilliary verb or in establishing the fact that the sentence has no auxilliary. The diagram also expresses another convention: unless the arcs leaving a

state are explicitly numbered in the diagram, the clock-wise order of arcs from the top of a state-circle corresponds to the order of arcs in the grammar listing and the order in which transitions are attempted. By convention, the initial state of the whole grammar is state S/.

3.2.1 THE SENTENCE LEVEL NETWORK

1. The basic strategy

With these conventions established, we can examine the way in which the parser uses this grammar to analyze sentences: Consider the simple sentence (7):

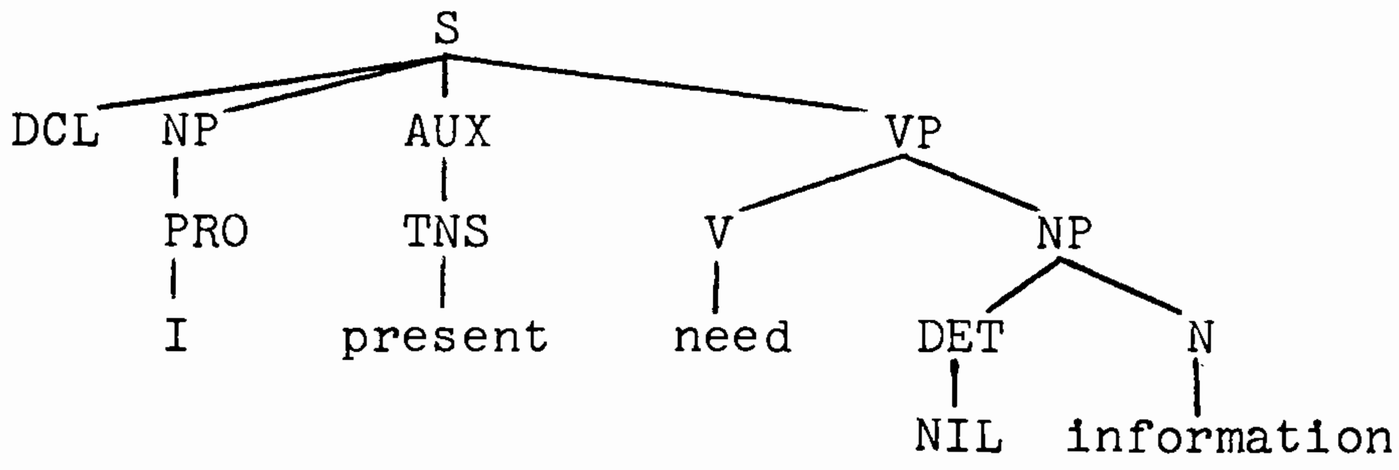
(7) I need information.

The underlying structure of (7) is intuitively obvious, given only a slight familiarity with high-school grammar. The word I is the subject, NEED is the verb, and INFORMATION is the object. The parser begins by comparing the string to the grammar at state S/. The first word of the string (I) cannot start an English question, so the predicate QSTART fails, ruling out the first arc but permitting the third. Since I is not PLEASE, the second arc is also excluded, and so the third arc is the first transition. We jump to state S/DCL, having established that the sentence is declarative. Since a JUMP transition does not advance the input string, we are still looking at I. At S/DCL we try to find the subject noun-phrase, the word THERE in subject position, or

a subject complement. In this case, the push to the noun-phrase network (arc 2) is successful, returning the structure (8):

(8)


We enter state S/NP with NEED as the current word and (8) in the register SUBJ. Since NEED is the tensed verb, the CAT V transition is permitted, and the actions save the tense (present), the person-number code (X3SG = "anything except third-person singular"), and the root form of the verb (need) in the apropriate registers. We enter S/AUX looking at the last word of the sentence. Since we have already identified the subject and since its person and number agree with those of the verb, we jump to VP/V, and from there we jump to VP/HEAD. VP/HEAD is a landmark: whenever we reach it, we have identified the main verb (the head of the verb phrase) and the subject, and we can begin to look for post-verbal constituents. In this case, since need is transitive, we push for the object noun-phrase at arc 3, and successfully return with the object, INFORMATION. This is the end of the string, so we continue jumping through the grammar from VP/NP to VP/VP and then to S/VP, from which we pop the recovered deep structure:



This is the basic strategy for simple, active, declarative, transitive sentences: at S/DCL, we have decided that the sentence is declarative; at S/NP, we have the subject; at S/AUX, we have the first (and only) verb, which carries us through to VP/HEAD; at VP/NP we have the direct object, and we then jump all the way to S/VP, where we pop the completed parse. For intransitive sentences such as (9),

(9) I went.

the jump arc (arc 1) is taken from VP/HEAD instead of the PUSH NP/arc, and the resulting structure does not have the NP node in the VP.

From this basic strategy, more complicated sentences are analyzed by varying and elaborating one or more segments of the analysis path.

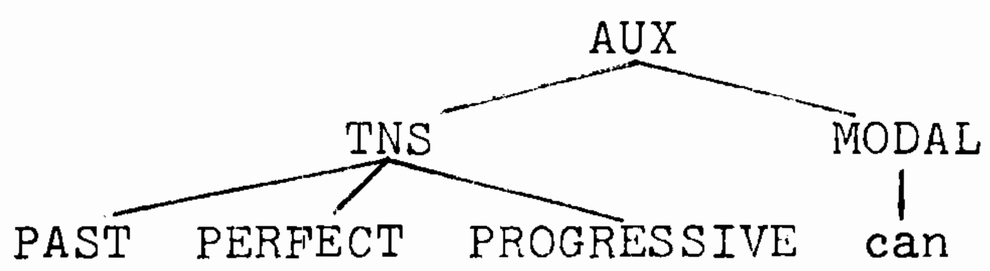
2. Auxiliary verbs

If the sentence has one or more auxiliary verbs beside the main verb, as in (10):

(10) I could have been going.

the analysis path is embellished at state VP/V: the CAT V arc at state S/VP picks up the modal verb COULD and stores it in the register MODAL instead of V. Then at state VP/V, HAVE satisfies the CAT V arc, so the loop is taken, making HAVE the main verb. Since BEEN also satisfies the CAT V arc, the loop is taken again, and with HAVE in V and BEEN marked as a past participle, PERFECT is added to the aspect register and BE is placed in the V register, and we re-enter state VP/V with GOING as the current word. Again, the CAT V arc is permitted, GO is the main verb and PROGRESSIVE is added to ASPECT. Finally we make the jump to VP/HEAD, having identified, as before, the main verb and the subject. The rest of the analysis resembles that of the simple intransitive (9), and the deep structure is similar except that the node AUX has been expanded to (11):

(11)



3. Passives

It was pointed out earlier that passive sentences (12) have the same meanings as their corresponding actives (7). We now show how the grammar maps them into the same deep structure.

(12) Information is needed by me.

The same sequence of transitions is taken for the passive as for the active, up to state VP/V, although the constituents identified and saved in registers differ. Upon entering state VP/V, the register SUBJ contained I and V contained NEED for the active, while for the passive, SUBJ holds INFORMATION and BE is the main verb as in (13):

(13) Information is available

At VP/V the analyses diverge. For the active the current word is INFORMATION, ruling out the CAT V loop, so the jump arc is taken to VP/HEAD, where the object is picked up. For the passive, the current word is NEEDED, the past participle of NEED. In this case, the CAT V arc is allowed, the subject INFORMATION is placed on the hold list by the conditional action, the indefinite noun-phrase SOMETHING is placed in SUBJ, and BE is replaced by NEED in V. At this point, we have identified the main verb, we have partially undone our previous assignment of INFORMATION as the subject, and the current input word is BY. AGFLAG has been set to indicate the possibility that the real subject is in a by-phrase later on. We now make the jump to VP/HEAD, but the push for the object noun-phrase fails with BY. Instead, the VIR NP arc removes INFORMATION from the hold list and places it in the object register. None of the arcs at VP/NP can deal with BY, so we jump to VP/VP, where we take the WRD BY transition to VP/AGT, since AGFLAG is set. Here we push

for a noun-phrase, find ME, and override the indefinite subject SOMETHING that we set up at VP/V. We continue along the basic analysis path and pop a structure at S/VP identical to that for the active. If the by-phrase had not been found in the sentence, the subject at this point would still be the indefinite SOMETHING, which agrees with our intuitions about the meaning of passivized sentences with missing agents.

4. Questions

In English, questions introduce a number of variations in the usual subject-verb-object sentence forms handled by the basic strategy. The states emanating from S/Q, and also some from S/, are designed to cover these possibilities. We recognize three major types of questions, yes-no (14a), question-pronouns and question-adverbs (14b and c), and question-determiners (14d and e).

(14)

a. Does each type/A rock contain krypton?

b. What is the average krypton concentration in type/A rocks?

c. How old is sample 10003?

d. Which rocks contain olivine?

e. How much olivine does each rock contain?

A yes-no question is characterised by the fact that a modal or auxiliary verb occurs before the subject. This may be at

the beginning of the sentence or after any number of fronted prepositional phrases. The predicate QSTART at state S/ precludes any other pre-subject verb, so the jump arc at S/Q brings us to S/NP with only the register TYPE set and with "be", "have" or a modal as the current word. We pick up the verb in the ordinary way, and arrive at S/AUX with a verb but no subject. Hence, we transfer to S/NO-SUBJ where, for (14a), the PUSH NP/ arc is successful, returning "each type/A rock". This agrees with the person-number code of the verb, and so becomes the subject. From state VP/V, the analysis is identical to the corresponding declarative, and the structures are identical except that the question structure has a type node "Q" instead of "DCL".

A question-pronoun or adverb is a WH-word that can stand by itself as the object of interrogation, for example, WHO, WHAT, WHEN, WHERE, WHY, and HOW. The root forms in the dictionary for these items are complete NP or ADV structures, and the CAT QWORD arc makes a copy of this structure (so that other parts of the grammar do not do permanent damage to the dictionary). Most of the pronouns can serve as either the subject or an object of the sentence; these are saved in the register WHO until further information determines whether they are to be moved into SUBJ or held for the post-verb modifier arcs. The question-adverbs, WHEN, WHERE, and HOW, and the pronoun WHOM cannot serve as the subject, so they are held immediately,

to be picked up later by VIR arcs. If the question-adverb is "how", a detour is made to check if the following word is an adjective or an adverb, as in (14c) above. If so, the adjective or adverb is also held, to be picked up later by a VIR arc. In any case, we enter state S/NP looking at the first verb. If there is a potential subject in WHQ, and the first verb is not an auxiliary or modal, then the WHQ word must be the subject, as in (15), so we rearrange the registers.

(15) Who wants the information?

If the verb is a modal or auxiliary, then the WHQ word is still a possible object, as in (14b), so we postpone a decision and enter S/AUX without a subject. Here, we transfer to S/NO-SUBJ, where for sentences such as (14b), we push and recover the full noun-phrase subject. This means that WHQ contains an object, so we add it to the hold-list. For sentences such as (16),

(16) What is available?

where there is no noun-phrase in this position, we know at last that the WHQ must be the subject, and the registers are rearranged on the jump arc. From VP/V, the analysis for QWORD questions follows the basic strategy, except that VIR NP or VIR ADV arcs are taken if the QWORD structure was held.

If an adjective was held from a "how<adjective>" construction, and the main verb is "be", in state VP/V, the

adjective replaces "be" as the main verb. Thus, the structure built from "How old is Sample 10003?" is:

```
S   Q
    NP   NPR   SAMPLE
                10003
    AUX   TNS   PRESENT
    VP   V   ADJ   OLD
         ADV   HOW
```

Finally, questions with question-determiners fall into two groups, those associated with count nouns (with QDETs "which", "what" and "how many") and those asociated with mass nouns (with QDET "how much"). The first group are analysed as questioned noun phrases, with the noun phrase containing the question-determiner becoming the deep structure subject, independent of its surface structure function. This is very similar to a predicate calculus representation and makes it clear that the questioned noun phrase, in general, has the widest scope. For example,(14d) is analysed as:

```
S   NPQ
NP   DET   WHQ
     N   ROCK
     NU   PL
     S   QREL
         NP   DET   WHR
              N   ROCK
              NU   PL
         AUX   TNS   PRESENT
         VP   V   CONTAIN
              NP   NPR   OLIVINE
```

i.e. "Which rocks such that they contain olivine (exist)?",

and (17):

(17) In which phases does S10005 contain krypton?

is analysed as:

```
S    NPQ
     NP    DET    WHQ
           N    PHASE
           NU    PL
           S    QREL
                NP    NPR    S10005
                AUX    TNS    PRESENT
                VP    V    CONTAIN
                      NP    NPR    KRYPTON
                      PP    PREP    IN
                NP    DET    WHR
                      N    PHASE
                      NU    PL
```

i.e. "Which phases such that S10005 contains krypton in those phases (exist)?"

When a question-determiner like "what", "which" or "how many" starts a sentence, it involves a push from state S/QDET for a full noun phrase structure with a question-determiner. Since the NP/ network does not recognize WH-words at the beginning of a noun phrase, the determiner must be picked up at the S-level and sent down into the DET register. Also sent down is a flag indicating that the remainder of the sentence, after the noun phrase containing the question-determiner should be made a relative clause of type QREL on that noun phrase. When the complete noun phrase is returned, it is placed in WHQ for S/NP to pop. This completes the analysis.

The noun phrase containing a question-determiner can also be in a prepositional phrase at the beginning of the

sentence. In this case, we push for a prepositional phrase in state S/, but note at state PP/PREP whether the following noun phrase begins with a question-determiner. If so, the determiner is again sent down into the NP/ network and put into the DET register. When the noun phrase is returned, it is lifted up to the S/ network and put into the NP register there. The prepositional phrase is also held, with the feature FRONTED. States S/QP1 and S/QP2 relativize the held prepositional phrase (i.e. replace the determiner of the embedded noun phrase with WHR), then push for a relative clause of type QREL. The relativized prepositional phrase is sent down into the relative clause as a verb phrase modifier (VMOD). When the relative clause is returned, it is attached to the original noun phrase containing the question-determiner, and the whole phrase is placed in the WHQ register for S/NP to pop. This completes the analysis.

The second group of questions containing question-determiners comprises those questions asking "how much". They are analyzed along the lines of QWORD questions, though, again, the question-determiner must be picked up at the S-level and sent down into the NP network. For example, (14e) is analyzed as:

```
S    Q
     NP    DET    EACH
           N    ROCK
           NU    SG
     AUX    TNS    PRESENT
     VP    V    CONTAIN
           NP    DET    POSTART    COMP    ADV    HOW
                                   MUCH
                 N    OLIVINE
                 NU    SG
```

The reason for using this analysis, rather than the more elaborate one discussed previously for the other question-determiners, is that the determiner "how much" does not interact with other determiners to cause scope problems. This was one of the reasons for adopting the previous, more elaborate analysis.

5. Existential THERE.

Sentences in which a form of the verb BE (or EXIST) occurs often have counterparts in which the subject is replaced by the word THERE and the real subject occurs after the BE as in (18):

(18) There is a document.

If BE is in fact the main verb, the sentence is interpreted as asserting the existence of its real subject. For sentences of this type, the WRD THERE arc is taken at state S/DCL, setting the register THERE but leaving SUBJ empty. We are still allowed to jump from S/AUX to VP/V, but we cannot go on to VP/HEAD without the subject. Thus, we push

for a noun-phrase on the second arc from VP/V, having seen a THERE-BE combination. The noun-phrase A DOCUMENT is returned, and since it agrees with the verb's person-number code, it becomes the subject. The rest of the analysis is ordinary, except that when we finally do jump to VP/HEAD, if the main verb is still BE, we convert it to EXIST. Thus the structure for (18) is:

```
S    DCL
NP   DET   A
     N     DOCUMENT
     NU    SG
AUX  TNS   PRESENT
VP   V     EXIST
```

Notice that this strategy allows the real subject to occur at any position in the string of auxiliary verbs following THERE, as long as the immediately preceding verb is BE or EXIST. Thus the sentences in (19) can be parsed properly:

(19)

a. There could be a document.

b. There could have been a document.

c. There could have been a document telling about...

An existential THERE can also occur in the subject position of a question; hence the WRD THERE arc at state S/NO-SUBJ. If there is a WHQ in this situation, it becomes the subject. Otherwise, the subject is found with the push arc from VP/V, as above. If the WHQ resulted from a QDET question, the DO arc at state S/THERE allows for the resumption of an

extraposed noun modifier, as in (20):

(20) How many men were there who wanted the document?

6. Do-support.

In English the verb DO can occur as a main verb (21a), as a modal verb with the connotation of emphasis (21b), or as an auxiliary verb in questions and negations with no apparent meaning (21c-d).

(21)

a. They did it.

b. The sample does contain Plagioclase.

c. Did they want it?

d. The document does not contain the information.

The cases exemplified in (21c-d) correspond to the general rules that subjects and verbs can be inverted only if the first verb is an auxiliary; if it is a regular main verb, DO is inserted. Similarly a modal or auxiliary must precede the sentential negation operator, and DO is inserted if there is no other possibility. In transformational theory, the process of inserting DO is called DO-support.

The strategy for interpreting DO in the LSNLIS grammar is as follows: At state S/NP, where the first verb is picked up, DO is placed in the MODAL register, since it satisfies

the predicate MODAL. If another verb is not found (as in 21a)), then an action on the jump arc to VP/HEAD moves DO from MODAL to V, making it the main verb.

In sentences where the subject and verb have been inverted, the subject is sought at state S/NO-SUBJ. If the PUSH NP/ arc is successful and if DO is in MODAL, it is deleted and does not appear in the final parse. Likewise, if the CAT NEG arc is taken at state S/AUX, a DO in MODAL is again removed. Thus semantically empty occurrences of DO are correctly eliminated, while emphatic and main-verb DO's are preserved.

7. Imperatives

The strategy for imperatives is basically simple. An untensed (infinitive) verb beginning a sentence, optionally preceded by PLEASE, marks the sentence as a command. Imperatives usually have no overt subject and no modal or auxiliary verbs, so the arc from S/IMP sets up the understood subject YOU and the tense-indicator PRESENT, and terminates at VP/HEAD where post-verbal constituents are analyzed in the normal way.

8. Objects and complements.

We have already seen in the basic strategy, how simple transitive and intransitive sentences are analyzed.

Syntactic features on verbs can require other types of predicate complement structures; the set of paths leading from VP/HEAD to VP/VP allow for the various possibilities, and the annotated listing of the grammar should be consulted to determine precisely how a given sentence will be analyzed. Here we discuss a few common predicate complement forms.

The CAT ADJ arc (arc 2) from VP/V allows a predicate adjective to follow a copula verb such as BE, BECOME, APPEAR:

(22)

a. The concentration of krypton in S10007 is large.

b. The basalts are older than the breccias.

c. S10003 appears glassy under UV light.

The adjective is placed in the verb register, preceded by ADJ and followed by its features. The features come from both the adjective itself and the copula verb. For example, the verbs constructed from the sentences in (22) would be:

(22')

a. (ADJ LARGE)

b. (ADJ OLD COMPARATIVE)

c. (ADJ GLASSY SEEMING)

Predicate adjectives which form their inflections by joining "more" and "most" to the uninflected form will also

be recognized in state VP/V on the MEM (MORE MOST) arc (arc 3). The uninflected adjective will be recognized in the following state VP/COMP-ADJ, where the adjective will be placed in the verb register, again preceded by ADJ and followed by its features.

The paths for one- and two-word adjectives reconverge at state VP/ADJ, where a variety of complements can be recognized. For simple comparatives like (22b) above, the word "than" causes one to move to state VP/ADJ-COMP, where a simple noun phrase is sought. If found, it is made the sentential object. (22b) is analysed as:

```
S    DCL
     NP    DET    THE
           N      BASALT
           NU     PL
     AUX    TNS     PRESENT
     VP    V      ADJ     OLD     COMPARATIVE
           NP     DET   THE
                  N     BRECCIA
                  NU    PL
```

This path would not be successful if a sentence followed the word "than", rather than just a noun phrase. In that case, the sentence would be analysed as a sentential complement in the verb phrase. Other allowable complements recognized via pushes from VP/ADJ are seen in such familiar sentences as "John is easy to please" and "John is eager to please."

The arcs which push to the COMP/ network permit a variety of complement sentence structures. Some verbs can have as their direct object a complete sentence, often

preceded by the complementizer THAT:

(23) I believe that the document is important.

Other verbs can have complements beginning with the complementizers FOR or TO, (24) and arc 7 is taken for these constructions.

(24)

a. The document seems to be important.

b. We arranged for the document to be sent.

Indirect-direct object combinations are handled by a sequence of arcs from VP/HEAD to VP/VP. In the simplest case, the verb is followed by two noun phrases:

(25) Give me the information.

The first noun phrase is picked up by the PUSH NP/ arc at VP/HEAD, and our initial guess is that it is the direct object, along the lines of (26).

(26) Give the information to me.

When we find the second noun-phrase on arc 4 from VP/NP, we rearrange the registers, making the previous object the object of a dative prepositional phrase, as in (26), and making the second noun-phrase the direct object. The superficial difference between (25) and (26) are thus removed. The grammar allows for various combinations of noun-phrases and sentential complements in the direct and indirect object positions, but we shall not discuss the

details here.

9. Verb Modifiers

The grammar allows for two types of optional verb modifiers--adverbs and prepositional phrases. These usually occur after the predicate complement structures have been analyzed, and the loops at state VP/VP pick them up. They are added to the list of modifiers in VMODS. Adverbs and prepositional phrases can also occur at the beginning of the sentence, before the normal subject-verb constituents, and the loops at state S/ permit this possibilty. Finally, adverbs can occur at other places in the sentence, most often within the sequence of auxiliary and modal verbs; thus, the CAT ADV loop at state VP/V. Adverbs can sometimes occur in other positions, but the grammar currently will not recognize them.

There is a special arc leaving S/ to deal with negative adverbs such as HARDLY and BARELY. When these adverbs occur at the beginning of a sentence such as (27),

(27) Barely was there enough information.

the subject and verb are inverted as in a question. Instead of looking for the subject in the normal declarative position, the negative adverb arc leads directly to S/NP where the first verb is picked up. The rest of the analysis resembles that of a yes-no question, except that the type is

DCL instead of Q.

Finally, in WH-questions where the question word is an adverb (WHEN, HOW), the adverb is held and picked up on the VIR ADV arc at state S/VP, where it is added to VMODS.

3.2.2 THE NOUN-PHRASE LEVEL

The second major component of the grammar is the noun-phrase level (with state names beginning with NP/). It is entered by pushes from the S-level and prepositional-phrase (PP) states, and it also has recursive calls to itself. We now describe some of the strategies used in the analysis of noun-phrases.

1. The basic strategy

Consider the simple noun-phrase:

(28) the information

At state NP/ the determiner-article THE permits the CAT DET transition to state NP/ART. From there a number of arcs permitting optional constituents are by-passed by a series of jumps to state NP/DET. The CAT N arc picks up the noun INFORMATION, carrying us to NP/N. We then jump to NP/HEAD and finally to NP/NP, from which we pop the completed noun-phrase structure (29):

(29) NP DET THE

N INFORMATION

NU SG

The significant milestones in the analysis of a noun-phrase are thus as follows: at NP/DET, the series of determiner constituents (a simple article in (28)) has been analyzed and the appropriate structure has been built (by the function DETBUILD) and saved in the register DET. At NP/N a potential head of the noun phrase has been found, while at NP/HEAD the ultimate head has been determined. Finally, at NP/NP, the complete noun-phrase has been recognized. As at the S-level, more complicated noun-phrases are recovered by variations and elaborations of this basic analysis path.

2. Determiner structures

In English it is possible to omit all constituents before the head of the noun-phrase. For example, abstract, mass and proper nouns, and plural forms of common nouns, do not even require preceding articles, and the jump arc from NP/ to NP/ART is provided for such instances. On the other hand, noun- phrases permit more than just an artcile in the determiner structure. Following the analysis of Stockwell et al. (1968), we recognize ordinals and quantifiers (with accompanying partitives) as part of the "post-article" structure of determiners. We also recognize magnitudes (e.g. 5 ppm, 7.2 percent) as part of this structure.

Ordinals indicate the position of the object denoted by the noun-phrase in a sequence of objects (e.g. FIRST, LAST, NEXT). The constraint is that ordinals precede quantifiers and other prenominal modifiers, so that (30a) is acceptable but (30b) is not:

(30)

a. the next five samples

b. *the five next samples

The CAT ORD arc from NP/ART picks up ordinals and saves them in the register POSTART.

Following an ordinal, a quantifier is allowed as in (30a). The grammar of quantifiers is fairly complicated, and a separate level (QUANT/) is provided for their analysis. This level is called by the first arc leaving NP/ORD. The QUANT/ states recognize simple cardinals, magnitudes, and some comparative constructions (MORE THAN, LESS THAN); this is an area of the grammar that needs further expansion. If found, the quantifier is added to the ordinal structure in POSTART. If a post-article has been identified, a partitive can follow which can indicate the set from which the particular object was drawn (31a-b) or (in the case of mass nouns) the mass term which it quantifies (31c).

(31)

a. five of the samples

b. the last of the measurements

c. five pounds of lead

The partitive is usually introduced by OF, and the first arc leaving NP/QUANT looks for an OF prepositional phrase. If one is found, the head of the noun-phrase becomes the dummy element ONES, and the partitive phrase becomes the first post-nominal modifier. With the head firmly decided, we transfer to NP/HEAD to look for other modifiers. For certain quantifiers (e.g. ALL, BOTH) the OF can be missing; the second arc at NP/QUANT takes care of this case. Finally, the partitive structure can sometimes be fronted to the beginning of a sentence such as (32):

(32) Of the documents, how many are about ...

where a loop at S/ puts it on the hold list for a VIR PP arc to find. If there is no POSTART or if no partitive is found, the jump to NP/DET is taken, and DETBUILD puts the contents of the DET and POSTART registers into the final determiner structure.

3. Pre-nominal modifiers

After the determiner, a sequence of modifiers can occur before a potential head is found. These may include adjectives, participial forms of verbs, and adverb-adjective phrases. Arcs 1, 4, 6, and 7 at state NP/DET, together with the arcs at NP/ADV, pick up these constituents, saving them in the register ADJS. Examples of noun-phrases with these

modifiers are given in (33):

(33) a. the lunar samples

b. a folded schist

c. an intriguing fact

d. a very large vesicle

4. Other potential heads

In the basic strategy, the head of the noun-phrase is a noun, picked up on the CAT N arc from NP/DET to NP/N. Three other arcs parallel the CAT N arc, permitting the head of the noun phrase to be a title (arc 3) a proper noun (34a) (arc 10) or a gerund (34b) (arc 8).

(34) a. sample 10026

b. John

c. the processing of information

In any case, the ~~three~~ four arcs place the potential head (embedded in a structure indicating its type) in the register N.

A pronoun may also be picked up as the head of the noun phrase, as in (35).

(35) a. The one which contains kryptonite

b. Either one of the phases

c. What is it for s10003

The pronoun may or may not be preceded by a determiner (such

as "the", "either", "some", "any"), and the CAT PRO arcs from NP/ and NP/ART to NP/HEAD are there to recognize both possibilities.

We have departed from Stockwell with regard to our analysis of superlative adjectives. This is an area in which we are still working, so the following, while it represents the current state of the grammar, is not necessarily final.

Superlatives may function both as identifiers, (36a), which point to a single specific thing, and as predicates, (36b), like ordinary adjectives.

(36) a. The oldest type/A rock

b. Rocks which are most representative of the Apennine Region.

When they function as identifiers, they point to the one member of some set which satisfies some requirements. In (36a), the criteria require the one type/A rock to be older than all the other type/A rocks in the set. We recognize a superlative identifier by the definite determiner "the" preceding it, and analyze it as the head of its own noun phrase. The remainder of the original surface structure noun-phrase is analyzed as a partitive construction on the superlative. This indicates the set over which the superlative ranges. For example, (36a) is parsed as:

```
NP  DET  THE
    N OLD SUPERLATIVE
    NU SG
    PP  PREP   OF
        NP  DET  THE
            ADJ  TYPE/A
            N  ROCK
            NU PL
```

The JUMP NP/SUPERLATIVE and WRD (MORE MOST) arcs from NP/ART catch one- and two-word inflected adjectives, respectively, provided the adjective has been preceded by "the". (We also treat determined comparatives as identifiers and analyze them in the same way as superlative identifiers. For example, "the older sample" is analyzed as "the older of the samples". Here we still have implicit in the structure the information that the set over which the comparative ranges has exactly two members.

The above analysis of inflected adjectives as specifiers is incomplete in the following sense: it does not allow for such plural constructions as:

(37) a. The oldest samples

b. The largest TiO2 concentrations

where the criteria for "oldest" and "largest" have been changed so that more than one member of the set can meet those criteria. It is as if we had scales for the different properties and a threshold for each property beyond which that property was considered "most" itself. For example, "the oldest samples" might include all those samples older than 3 billion years, while "the oldest buildings" might

include all those buildings over 1000 years. As can be seen above, the threshold can be influenced by the set. We do not understand this use of inflected adjectives well enough yet to have incorporated it into the current LSNLIS system.

When an inflected adjective does not function as an identifier, we treat is as an ordinary adjective with the feature "comparative" or "superlative" as appropriate. This also differs from Stockwell's analysis of superlatives, which he considers part of the post-determiner structure. The WRD (MORE MOST) arc and the CAT ADJ arc on NP/DET pick up undetermined one- and two-word inflected adjectives.

When an inflected adjective occurs as a predicate adjective, we make it the main verb of the sentence, replacing the copula. In this case, the verb would get the feature "comparative" or "superlative", plus any features from the copula which it replaces. For example, (36b) is parsed as:

```
NP  DET  NIL
N   ROCK
NU  PL
S   REL
    NP   DET   WH
         N   ROCK
         NU  PL
    VP   V   ADJ   REPRESENTATIVE
                   SUPERLATIVE
         PP   PREP  OF
              NP   DET  THE
                   ADJ  APENNINE
                   N   REGION
                   NU  SG
```

Inflected predicate adjectives are caught in state VP/V on the CAT ADJ and MEM (MORE MOST) arcs. From this point on, they follow the course of normal adjectives, looking for verb phrase complements and modifiers.

There is a further set of words which may be picked up as the head of a noun phrase. This set now includes the words "average", "most", "least", "maximum" and "minimum", but it seems reasonable that the ordinals should be included in this set too. A member of this set is analyzed in a similar fashion to the determined inflected adjectives. It is made the head of a deep structure noun phrase, while the remainder of the original surface structure noun phrase is put into a partitive construction following it. The partitive indicates the set over which one of these function words ranges. For example, "the average concentration of iron in breccias" is analyzed as "the average of the concentrations of iron in breccias".

The first CAT N arc from NP/HEAD catches words in the aforementioned set and jumps to NP/AVG, where the partitive is constructed. From NP/AVG, we continue with the regular NP processing at NP/HEAD.

5. Noun-noun modification

It is very often the case that the first potential head in a noun-phrase is not the real head. For example, nouns

and proper nouns can be modifiers on other heads (38):

(38) a. Olivine analysis

b. data processing

c. Apollo 11 sample

When a potential head is first encountered at state NP/DET, we make the tentative assumption that it is, in fact, the head of the noun-phrase. At NP/N, another potential head implies that the previous head is actually a modifier on the new head; the loops at state NP/N (arcs 1, 5, 6, 7, 8 and 10) pick up the new head and add the old one to the list of modifiers in ADJS. This process can be repeated several times, as in (39). The CAT ADJ arc leading back to NP/DET means that the series of potential-head-modifiers can have regular adjectives interspersed.

(39) a. NASA mission control operations staff

b. Apollo 11 lunar samples

A possessive marker ('s) is also allowed after a potential head has been found, and the CAT POSS arc at NP/N picks it up, converts the previous head to a modifier, and returns to NP/DET. A gerund at this point might confirm that the noun-phrase is really a POSS-ING complement as in (40),

(40) John's winning of the race

and arc 9 at NP/DET pushes into the S/ level to look for this structure.

6. Relative clauses

The jump to NP/HEAD is taken only when the head of the noun phrase has been definitely determined. At this point a variety of post-nomial modifiers is permitted. If we are looking at a relative pronoun (41a) or a preposition followed by a relative pronoun (41b), we know that there is a relative clause, and we push to R/ to parse it.

(41) a. the samples which contain Olivine

b. the samples in which Olivine was found

Essentially, a relative clause is a sentence with a missing noun phrase, with the head and number of the noun-phrase that intuitively should fill the empty slot being the same as those of the noun-phrase in which the relative clause occurs. Thus we send down a copy of the noun-phrase to be used in the analysis of the relative clause.

The states R/ and R/WH pick up the relative pronoun and decide whether the copy of the noun-phrase sent down (in WH) can be the subject of the relative clause. If so, the noun-phrase is placed in SUBJ. Otherwise, it is held for later use. We then enter the S/ network at S/NP to complete the analysis. The path through R/PREP handles the preposition-relative pronoun clause.

The relative pronoun can also be left out of relative clauses, and arcs 7 and 9, which push to R/NIL, handle such "reduced" relative clauses. The reduced relative can begin

with a noun-phrase (42a), a participial verb (42b), or an adjective (42c).

(42) a. the elements the sample contained

b. the elements contained in the sample

c. the information available

A noun-phrase is still sent down to WH, and its ultimate destination is decided at R/NIL. Notice that there are two arcs that push to R/NIL -- the arc usually taken is arc 9, which follows the jump to NP/NP. This means that we will not look for reduced relatives unelss we have failed on other paths; this strategy improves the efficiency of the grammar, since R/NIL can lead to long blind alleys. However, this also implies that a reduced relative on a noun-phrase in a prepositional-phrase within another noun-phrase would be attached to the first noun-phrase in the first parse. Thus in (43) the reduced relative would be tried first as a modifier of MAN instead of PARK:

(43) the man in the park the girl frequented

The TST R/NIL arc (arc 7), which calls the SUSPEND mechanism, is included to provide the correct analysis in these cases. It can only be taken if a prepositional-phrase (IN THE PARK) has been found in the current noun-phrase.

Whenever we find a relative clause, we move to state NP/R, where additional full relatives are allowed until we

finally jump to NP/NP.

7. Other post-nominal modifiers

The grammar handles other types of post-nominal modifiers, including prepositional phrases, sentential complements, and parenthetic comments. The PUSH PP/ loop recognizes a sequence of prepositional-phrases and places them, along with the relative clauses, in the register NMODS. The register PPFLAG is set, which enables the TST R/NIL arc and prohibits the normal PUSH R/NIL.

The PUSH FOR/NP loop handles TO-completments, such as (44),

(44) the way to do it

which can appear on a wide variety of nouns. The arcs which push to the COMPL/ network can only be taken for certain head nouns, those which take a THAT-complement (e.g. FACT, STATEMENT, CLAIM) as in (45):

(45) the fact that the documents are not available

Notice that THAT can also introduce a relative clause; the difference between a relative clause and a THAT-complement is that in the relative clause there is an empty noun-phrase slot to be filled. The THAT-complements are complete sentences.

Other arcs leaving state NP/NP allow for such constructions as a colon following a noun-phrase and a comma either indicating the beginning of a conjoined sequence of noun-phrases, or introducing a transitive adverb (ESPECIALLY, PARTICULARLY) and its following noun-phrase. The annotated listing describes these arcs in detail.

Chapter 4

SEMANTIC INTERPRETATION STRATEGIES

4.1 Motivation

In Section 2.3, we presented an outline of the operation of the semantic interpretation component. In this Chapter, we will discuss the particular semantic strategies embodied in the semantic rules and the way in which they produce semantic interpretations from the syntactic structures being interpreted.

The semantic rules used in the system fall into two classes. One class deals with grammatical constructions which have real meaning to the system (determiners, verbal constructions such as "give me", "I need", etc., and noun constructions such as "analyses", "average...", "ratios", "references", etc.). The other class deals with constructions whose meaning is not apparent to the system, but which are instead to be interpreted as subject indicators or other restrictions on the references to be retrieved. The system generally attempts to interpret a sentence or request in terms of its semantic rules for specific constructions, but if it fails to find an interpretation of the sentence in terms of the constructions which it knows, it types a comment to that effect and attempts to interpret the request as a Boolean combination of terms for retrieval from the keyphrase table. The set of semantic rules which perform this Boolean interpretation are called topicrules and may either be invoked explicitly by constructions such as "references on", "bibliography of", etc. or it may be invoked by default when a request fails to interpret normally.

4.2 The General Semantic Framework

The general semantic format of the LSNLIS system is essentially that described in Woods (1967, 1968). That is, the retrieval component of the system consists of a set of primitive commands, functions, and predicates which may be combined and quantified to produce semantic interpretations which are essentially retrieval programs for computing the truth values of propositions or for carrying out commands. The task of the semantic interpreter is to translate the parse tree of the sentence into an expression in this formal query language which can then be executed to retrieve or compute the answer.

4.3 Semantic Representation

The fundamental components of the retrieval component are the primitive functions, commands, and predicates which the machine understands. These include specific retrieval functions for concepts such as "average", "ratio", "analysis", "mineral", "isotope", etc. and general functions such as the quantifier function FOR and the list enumeration function SEQ. The typical retrieval operation is based on the quantification of propositions and commands by quantifiers of the form:

(FOR QUANT X / CLASS : P(X) ; Q(X)) where QUANT is a quantifier (EACH, EVERY, SOME, etc.), X is the variable of quantification, CLASS is the class of objects over which the variable is to range, P(X) is a restriction on this range (i.e. the only objects in CLASS which are of interest are those for which P(X) is true), and Q(X) is the proposition or command being quantified. The class of objects is specified by a special enumeration function (such as DATALINE) for enumerating the objects in the class (e.g. by searching the table) or by the function SEQ which takes a list as an argument and enumerates the elements of that list. Typically, Q(X) will be the command (PRINTOUT X) which prints out a representation of the object X on the teletype. For example, the expression

(FOR EVERY X1 / (SEQ PHASES) : T ; (PRINTOUT X1)) is a retrieval program which will print out the names of all of the members of the list PHASES (the list of all the names of phases of samples which are recorded in the system's data base). This framework for semantic representation provides a powerful formal language for the expression of requests for the retrieval component.

4.4 Interpreting Sentence Nodes

The interpretation of a sentence node occurs in two phases distinguished by the use of different values for TYPEFLAG. The first phase, with TYPEFLAG NIL determines whether there are any governing operators or commands such as NOT, TEST, etc., which govern the sentence. It is essentially a preprocessing phase prior to the actual examination of the sentence itself and consists in matching rules from the global list PRERULES (which is independent of the particular verb which governs the sentence). Of these rules, S:AND and S:OR interpret conjoined sentences; S:DCL deals with the interpretation of declarative sentences; S:IMP interprets imperative sentences; and S:WHQ and S:YES/NO deal with questions (the former with questions containing question words such as "which" or "what" and the latter with simple yes/no questions). S:NPU deals with noun-phrase utterances (i.e. sentences which consist only of a single noun-phrase with no verb). Other prerules interpret negative sentences and different syntactic formats.

All of the PRERULES (with the exception of S:NPU and a few others) specify the subsequent interpretation of the same node with the TYPEFLAG SRULES (the exceptions call for the interpretation of specific lower nodes). This second call for the interpretation of the node begins the second phase of processing. In this case,

the rules to be tried are taken from the property list (i.e. dictionary entry) of the head of the sentence (i.e. the verb) under the property SRULES. Alternatively, if the verb does not itself have any SRULES, but has as one of its semantic markers a word which has SRULES on its property list, then the rules to be matched will be taken from the list associated with the marker. For example, the rule S:GIVE which interprets sentences of the form "give me information on ..." is used for the interpretation of many words which are synonyms of "give" in this context. This is indicated by putting the semantic marker GIVE in their dictionary entries, and therefore enabling them to use the SRULES from the dictionary entry for the word "give".

4.5 Interpreting noun-phrases

The semantic rules which interpret sentences generally require the interpretation of one or more noun phrases as subconstituents of their interpretation. The rule S:NPU for example, requires only the interpetation of its single constituent noun phrase. Like the interpretation of sentences, the interpretation of noun-phrases occurs in several phases. The first of these, with NIL TYPEFLAG interprets the determiner structure of the noun phrase to determine what type of quantifier is to govern it. This phase consists of matching rules from the global list DRULES. These rules examine the determiner and number of the noun phrase and assign a basic quantifier structure. They also call for the interpretations of the same node in two different modes--NRULES and RRULES--for the other two phases. NRULES and RRULES both are taken from the property list of the head of the noun-phrase (i.e. the noun) or from the property lists of words which occur in the list of markers for the head noun. NRULES interpret the noun of the phrase and any arguments which it may require (i.e. if it is a function); RRULES interpret any further restrictive modifiers which may occur in the noun phrase. Modifiers which do

not match any RRULES are ignored, and relative clauses are handled by a special mechanism which tags the relative pronoun of the relative clause with the variable of quantification and calls for its interpretation as a sentence.

4.6 Interpreting Topics

The above description of the interpretation of noun phrases applies only to noun phrases which have direct and understandable meaning to the system (e.g. "documents on ..." "analyses of ...", etc.) Other noun phrases consist of topic descriptions and are treated in an entirely different manner by the system. In the latter case, a list TOPICRULES specifies a global list of rules for translating syntax trees into Boolean combinations of key phrases. These rules are grouped on TOPICRULES into AND and OR groups in the way in which any resulting matches are to be combined. Each topic rule corresponds to a particular type of key phrase which may be present in a syntactic construction, and specifies in its left-hand side the proper context and structure for the extraction of that key phrase.

The list of TOPICRULES is used to interpret a noun phrase instead of the usual sequence of DRULES, NRULES, and RRULES whenever the call to interpret the noun phrase is made with TYPEFLAG TOPIC instead of TYPEFLAG NIL. Semantic rules of the ordinary type are used to invoke this special type of interpretation whenever they locate a context which is definitely a topic. For example, the rule R:DOC-ON interprets restrictions on a noun which is semantically marked DOCUMENT that begin with the preposition "on". It interprets such constructions as "data on X" by calling for the interpretation of X with TYPEFLAG TOPIC and constructing an instance of the ABOUT predicate indicating that the documents in questions are about the topic X.

Since the data base of the system is quite limited in scope, it is quite likely that the user may ask the system for something which it does not understand. In this case, the system will attempt to interpret the unknown thing as a topic for which references are required. For this purpose, noun phrases in the environment of a "give me ..." sentence or a noun-phrase utterance are interpreted with a TYPEFLAG REFS? which first trys to interpret the noun phrase in the normal way, and failing that, produces a call to the rule REFERENCES which prints a comment to the user and interprets the noun phrase as a topic.

4.7 An Example

As an example of the semantic interpretation procedure, consider the "sentence" (actually a noun-phrase utterance):

(ANALYSES OF SAMPLE S10046 FOR HYDROGEN)

This sentence produces the following tree structure when processed by the parser:

```
S    NPU
     NP      DET     NIL
             N       ANALYSIS
             NU      PL
             PP      PREP     OF
                     NP       DET     NIL
                              NPR     SAMPLE
                                      S10046
                              NU      NIL
PP           PREP       FOR
             NP         DET     NIL
                        N       HYDROGEN
                        NU      SG
```

The function INTERP does the interpretation. It first attempts an interpretation of the whole sentence, S. Using the list of rules, PRERULES, it finds that the sentence is a noun phrase utterance (NPU) and should receive the interpretation attached to its main noun phrase (NP). At this point, the right-hand side of the rule (PRED (PRINTOUT (# 1 1 REFS?))) indicates that the interpretation is a predicate (which may later be quantified) governing the command PRINTOUT. It indicates that the things to be printed out are to be determined by interpreting the noun-phrase (# 1 1) with the typeflag REFS? (i.e. the noun phrase may be either a topic description or a noun phrase whose head is semantically interpretable. The function PRED will be executed after the substitution of the interpretation on the noun-phrase has been made in the right-hand side, and it will grab any quantifiers which have been produced by the constituent interpretations.

The interpreter now begins the interpretation of the noun phrase using the two rules REFRULE? and REFRULE (determined by the TYPEFLAG REFS?). The first attempts the interpretation in the normal mode beginning with the global list DRULES. Since the noun phrase does have an interpretable head ("analysis") this interpretation will succeed, and the rule REFRULE will never be tried.

As we mentioned, the DRULES are used to interpret the determiner and number of the noun phrase and determine the type of quantifier to be produced. This includes the case of no determiner (determiner NIL). In this case, the special generic quantifier GEN is produced, and placed in a buffer string for the function PRED to grab. A variable of quantification (X13) is assigned to the noun phrase, and the interpreter is called with the TYPEFLAG NRULES to interpret the noun.

4.7.1 Interpreting the Noun Phrase

In the dictionary, the word "analysis" contains the semantic marker ANALYSIS, and under the property NRULES it contains the list (N:ANALYSIS N:MODAL-ANALYSIS). The two NRULES specify the interpretations of the two types of analyses which the system recognizes--the chemical analysis of some element in some phase of a sample, and the modal analysis of some mineral in a sample. The first rule applies when there is no adjective "modal" present in the noun phrase, and the second applies when there is such an adjective. The rule which will be applicable in this case is thus N:ANALYSIS. This rule is shown in figure 4-1.

The rule N:ANALYSIS specifies constituents which must be present (or absent) in a noun phrase in order for the rule to match, and specifies the enumeration function which is to be used for the quantification if the rule match is successful. First, it specifies that the noun of the noun phrase be a member (MEM) of the semantic class ANALYSIS (i.e. that its dictionary entry contain the semantic marker ANALYSIS). This is true not only of the word "analysis" itself, but also of other words which can behave as synonyms for "analysis" in this context (such as "concentration", "composition", etc) Secondly, the rule specifies that there must be no modifier "modal" (in this case, the rule N:MODAL-ANALYSIS would apply). The next three components of the pattern part of the rule specify the "arguments" of the head noun -- the sample, phase, and constituent of interest. These may be specified syntactically several ways -- either as an adjectival modifier, a prepositional modifier, or by default. Thus, the constituent of an analysis (the fifth component of the pattern) can be specified by a prepositional phrase whose object is either an element, an oxide, or an isotope; by an adjectival modifier

```
[N:ANALYSIS
     (NP.N (MEM 1 ANALYSIS))
     (NOT (NP.ADJ (EQU 1 MODAL)))
     (OR (NP.PP (MEM 2 (SAMPLE ROCK)))
         (NP.PP.PP (MEM 2 (SAMPLE ROCK)))
         (NP.PP.PP.PP (MEM 2 (SAMPLE ROCK)))
         (DEFAULT (2 NP (DET ALL)
                        (N SAMPLE)
                        (NU PL))))
     (OR (NP.PP (MEM 2 (PHASE MINERAL)))
         (NP.PP.PP (MEM 2 (PHASE MINERAL)))
         (NP.PP.PP.PP (MEM 2 (PHASE MINERAL)))
         (NP.ADJ#2 (MEM 2 (PHASE MINERAL)))
         (NP.PP.ADJ-N (AND (OR (EQU 2 FINE)
                               (EQU 2 COARSE))
                           (MEM 1 DUST)))
         (DEFAULT (2 NP (NPR OVERALL))))
     (OR (NP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
         (NP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
         (NP.PP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
         (NP.ADJ#2 (MEM 2 (ELEMENT OXIDE ISOTOPE)))
         (DEFAULT (2 NP (DET EVERY)
                        (ADJ MAJOR)
                        (N ELEMENT)
                        (NU SG))))
     (OR (NP.ADJ (EQU 1 CHEMICAL))
         (DEFAULT (1 NPR NIL)))
         -> (SSUNIONF (DATALINE (WHQFILE (# 3 2 SSET)) (# 3 2 SSET)
(# 4 2) (# 5 2 SSET)))  ]
```

Figure 4-1 The NRULE N:ANALYSIS

which is one of the above three types; or by default in which case quantification over the major elements is assumed. These alternatives are represented in the semantic rule by a group of templates OR'ed together with the default option at the end of the OR. The default option applies if and only if none of the other components of the OR are satisfied.

The last component of the pattern specifies the optional presence of the adjective "chemical". This is done so that when the user requests "chemical analyses", the Semantic Interpreter does not apologise for being unable to interpret "chemical" as a modifier of "analysis". (See the section on User Aids, Chapter 1, for a discussion of the interpreter's reaction to things it cannot "understand".) In the current data base, all analyses are basically chemical ones, so "chemical" does not add anything to the interpretation of "analysis". However, the Semantic Interpreter should know the difference between harmless optional modifiers and ones which are important.

4.7.1.1 Prepositional Arguments

It would be nice if the parser provided a syntax tree in which the various prepositional arguments of a noun phrase were attached directly to the noun phrase where they make sense semantically, and we have experimented elsewhere with a rudimentary facility for using the information in the semantic rules to guide the parser in the placement of prepositional modifiers. In the present system, however, we have taken the opposite tack and provided semantic rules which can locate the necessary prepositional arguments even when the parser has placed them in the wrong place. Thus, the templates which match prepositional modifiers in the rule N:ANALYSIS make use of the tree fragments NP.PP,NP.PP.PP, and NP.PP.PP.PP. which can locate a prepositional phrase one, two, or three levels deep in a noun phrase.

Note that the templates which check prepositional phrases make no checks on the preposition itself. The rule was originally written this way as an expedient since there are many possible combinations of prepositions which may occur in this context and their enumeration was tedious. However, the rule in this form has been very successful--we have encountered no cases in which a sentence was falsely interpreted because of this failure to check the prepositions, and if no such cases arise, we will probably leave the rule in its present form (since it is faster without the additional checks).

4.7.1.2 The Right-Hand Side of the Rule

The right-hand side of the rule specifies the enumeration function which is to be used to enumerate the analyses to which the request refers. This will be a call to the data base function DATALINE which takes arguments specifying the sample, phase, and constituent of interest and enumerates the lines of the table which correspond to the values of these arguments. The first argument to DATALINE is a call to the function WHQFILE with the sample as its argument. WHQFILE returns the name of the file on which the analyses of the sample are located. This is so that DATALINE will know which file to search for them. The rule specifies the interpretation of node (# 3 2), with typeflag SSET, as being the sample required. The final three arguments are the sample, phase and constituent, respectively, and these positions are to be filled respectively with the interpretations of the nodes (# 3 2), (# 4 2) and (# 5 2). The typeflag SSET with which the sample and constituent are to be interpreted specifies the nodes to be interpreted as sets, if possible, and not as quantified variables. Nodes can be interpreted as sets if they are plural ("the samples", "all the halogens") or determined by "every" ("every type A rock"). Otherwise, if any of these argument positions receive direct proper noun interpretations, then these interpretations are inserted directly in place of the expressions (# 3 2) etc., while if any of them are quantified by "each" ("each rare earth element"), their interpretation will be the variable of quantification and the governing quantifier will be passed up the tree to the sentence which dominates the noun phrase.

4.7.1.3 Completing the NRULE Interpretation

In the particular case at hand, the third template will match "of sample S10046", the second will default to "overall", and third will match "for Hydrogen". Thus, in order to complete the interpretation of the node, the interpretations of the nodes (# 3 2)(sample S10046), (# 4 2)(overall), and (# 5 2)(Hydrogen) will be called for, producing the interpretations (NPR* X1 / (QUOTE S10046)), (QUOTE OVERALL), and (NPR* X2 / (QUOTE H)), respectively. No quantifiers are produced by any of these sub-interpretations. The result of the rule N:ANALYSIS is thus the enumeration function:

```
(DATALINE (WHQFILE (NPR* X1 /(QUOTE S10046)))(NPR* X1 /
     (QUOTE S10046))(QUOTE OVERALL)(NPR* X2 / (QUOTE H)))
```

At this point, we return to the DRULE which called for the NRULE interpretation of this node and INTERP begins another interpretation of the same node with typeflag RRULES to pick up any possible modifiers. It again consults the property list of the head noun ("analysis") and finds the RRULES, R:ANALYSIS-REF and R:ANALYSIS-TAG. These rules allow for the optional restriction of "analysis" by some tag or reference, e.g. "Analyses of hydrogen in D70-246 with tag 2". In addition to these RRULES, RULES sends the function MATCHER the four universal RRULES, R:REL, R:QREL, R:ADJ and R:PP to match against the node. These rules see if there are any relative clauses, special relative clauses of type QREL (see section 3.2.1.4, where these clauses are discussed), unused adjectives or uninterpretted prepositional phrases. The latter two rules are used to check that all adjectives and PP's on the node have been interpreted or have contributed to the interpretation of the node. If not, they inform the user of what the Semantic Interpreter has ignored and ask him what to do about it. The former two rules call for the interpretation of any relative clauses on the node, but follow the latter two rules if any of them are uninterpretable.

In this case, none of the RRULES apply and the result of the interpretation is the vacuous restriction T. Control again returns to the DRULE looking at "analyses ..." and the quantifier:

```
(FOR GEN X13 / (DATALINE (WHQFILE (NPR* X1 / (QUOTE S10046)))
 (NPR* X1 / (QUOTE S10046))(QUOTE OVERALL)(NPR* X2 / (QUOTE H)))
 : T ; DLT )
```

is constructed. This quantifier is returned to the higher sentence (the NP utterance) which called for the interpretation of this node, and the semantic interpretation X13 is attached to the node.

4.7.2 Completing the Interpretation of the Noun-Phrase Utterance

Recall that the semantic rule which interpreted the top level noun-phrase utterance was left pending with the right-hand side (PRED (PRINTOUT (# 1 1 REFS?))). The interpretation that is returned by the call for the interpretation of (# 1 1 REFS?) is now X13, and the quantifier governing this variable has been placed in the quantifier string QUANTS which is being passed up along the tree. After the substitution, the resulting expression

(PRED (PRINTOUT X13)) is executed and results in the 'grabbing" of the quantifier(s) in the string QUANTS to produce the resulting interpretation:

```
(FOR GEN X13 / (DATALINE (WHQFILE (NPR* X1 / (QUOTE S10046)))
                         (NPR* X1 / (QUOTE S10046))
                         (QUOTE OVERALL)
                         (NPR* X2 / (QUOTE H))) : T ; (PRINTOUT X13))
```

This expression is a retrieval program which will range over the set of table lines specified by the call to DATALINE, binding X13 to each in turn, and executing the expression (PRINTOUT X13) for each such line. The result of the execution will be a printout of the lines reporting overall hydrogen analyses of sample S10046.

4.8 Variations on the Example

The interpretation of sentences such as "give me all analyses of sample S10046 for Hydrogen" are interpreted in exactly the same manner as the preceding example except that instead of the rule for noun phrase utterances, a rule S:GIVE (for various paraphrases of "give me ...") will apply to produce the right-hand side (PRED (PRINTOUT (# 2 1 REFS?))). The interpretation of phrases such as "overall Hydrogen analyses of S10046", "Hydrogen analyses of S10046, "analyses of Hydrogen in S10046", etc. will differ only in which components of the OR'ed templates of N:ANALYSIS are chosen. The interpretations of phrases such as "analyses of major elements in S10046" will differ only in that additional quantifiers will be passed up by the interpretation of the embedded quantified noun phrases.

4.9 Anaphoric Reference

LSNLIS has been designed as a conversational system. Thus, it must be prepared to deal with such a common conversational device as anaphoric reference. Several examples of anaphoric reference are shown in the following sets of requests.

(1) a. Give me all analyses of Sample 10046 for hydrogen.
    b. Give me them for oxygen.

(2) a. Do any breccias contain aluminum?
    b. Which are those breccias?

(3) a. Which coarse-grained rocks have been analysed for cobalt?
    b. Which ones have been analysed for strontium?
    c. Which ones have been analysed for strontium too?

(4) a. How much TiO_2 is in type B rocks?
    b. How much silicon is in them?

The problem is in finding the referent of each anaphoric element: e.g. "them" in (1) and (4), "those breccias" in (2) and "ones" in (3).

The resolution of anaphoric reference is done in LSNLIS

by the Semantic Interpreter, and not by the Parser. Thus, we will find pronouns like "one" and "they" and determiners like "those" and "that" in the parse tree given the semantic interpreter. Whether it would be more efficient to let the Parser resolve anaphoric reference itself, or let the two phases share the burden is not yet clear, but we do not claim that the strategies we use now should be in any way final. Much remains to be done in this area.

Our main device for dealing with anaphoric reference makes use of the "variables of quantification" mentioned earlier in the chapter. During interpretation, every noun phrase that the Semantic Interpreter attempts to interpret becomes associated with a variable of quantification. These variables are very much like Chomsky's referential indices. After the interpretation of the request is completed, each variable also becomes associated with the interpretation of its noun phrase, as well as its syntactic structure. This latter association is done by the function SCOPEFINDER, called by INTERP. The above information is stored on the property list of each variable, under the properties NODE and INTENSION, respectively. For example, the following shows the property list of variable X13, at the completion of our example request:

```
X13
NODE      (NP   (DET NIL)
                (N ANALYSIS)
                (NU PL)
                (PP (PREP OF)
                    (NP (DET NIL)
                        (NPR SAMPLE
                             10046)
                        (NU SG)))
                (PP (PREP FOR)
                    (NP (DET NIL)
                        (N HYDROGEN)
                        (NU SG))))
INTENSION (GEN  (DATALINE (WHQFILE (NPR* X1 /
                                        (QUOTE S10046)))
                          (NPR* X1 / (QUOTE S10046))
                          (QUOTE OVERALL)
                          (NPR* X2 / (QUOTE H))) T)
```

Also at the completion of a request, the function SPROC adds the variables used in its interpretation to the top of the list ANTECEDANTS, for use in resolving future anaphoric reference.

We distinguish two types of anaphoric reference in LSNLIS, partial anaphoric reference and complete anaphoric reference. (1b.) is an example of partial anaphoric reference in that "them" refers to only part of the previous noun phrase "analyses of Sample 10046 for hydrogen", that is, to "analyses of Sample 10046". The prepositional phrase "for oxygen" replaces "for hydrogen" in the original request.

The remaining examples illustrate complete anaphoric reference of two types: anaphoric reference to the question set (i.e. the phrase used in the request) and to the answer set (i.e. the set of answers to the request). Example (3) illustrates these two types best. "Ones" in (3b.) refers to the question set in (3a.) "coarse-grained rocks", while "ones" in (3c.) refers to the answer set in (3a.), "coarse-grained rocks which have been analysed for cobalt". We take the word "too" in (3c.) as signalling this difference.

We shall give in what follows, a brief sketch of how the Semantic Interpreter treats each type of anaphoric reference, then go on to discuss its limitations in this area.

There are two semantic rules for interpreting anaphoric pronouns and determiners, D:ANAPHORA and D:SEMI-ANAPHOR. The former matches anaphoric pronouns and determiners which do not have any prepositional phrase or relative clause modifiers, while the latter matches those that do. The former represents complete anaphoric reference, while the latter, partial anaphoric reference. There are also two semantic rules for interpreting the pronoun "one", which can be influenced by the words "too", "also", and "in addition" in selecting the referent.

Consider example (1b) first. The right-hand side of the rule S:GIVE applies to this request and calls for the interpretation of the direct object of "give", in this case, "them in oxygen". We proceed to interpret its determiner structure, and since anaphoric pronouns are matched in the same cycle as determiners, we find that the rule D:SEMI-ANAPHOR matches the node. This rule calls for the application of the function SEMIANAPHOR to the entire node, as is. What SEMIANAPHOR does is to search through the list of antecedant noun phrases for one which has a syntactic and semantic structure parallel to the given node. In this case, it looks for one with a dependent prepositional phrase whose preposition is "of" and whose head noun has the same markers as "oxygen", that is, (ELEMENT). The noun phrase "analyses of sample 10046 for hydrogen" meets this description. SEMIANAPHOR then replaces the prepositional phrase "for hydrogen" with that "for oxygen", and returns, as the interpretation of "them for oxygen", the interpretation of "analyses of sample 10046 for oxygen".

SEMIANAPHOR, as it now stands, is only a first approximation to the problem of resolving partial anaphoric reference. To begin with, it is only applicable to anaphora with a single prepositional phrase parallel to one belonging to its antecedant, as in the example above. This also requires having correct modifier placement, a stage which we have not yet reached. Follow-up requests to (1a) which would be beyond the system's current capacity are ones like:

(5) a. Give me the oxygen ones.
b. How about them for oxygen.
c. Give me those that have been done for oxygen.

The following example shows the parsing and interpretation of the two requests in example (1).

```
SENTENCE:
(GIVE ME ALL ANALYSES FOR HYDROGEN IN SAMPLE 10046)
PTIMING:
1262 CONSES
4.796 SECONDS
PARSINGS:
S IMP
  NP PRO YOU
  AUX TNS PRESENT
  VP V GIVE
     NP DET ALL
        PRO ONES
        NU SG/PL
        PP PREP OF
           NP DET NIL
              N ANALYSIS
              NU PL
              PP PREP FOR
                 NP DET NIL
                    N HYDROGEN
                    NU SG
              PP PREP IN
                 NP DET NIL
                    NPR SAMPLE
                        10046
                    NU SG
     PP PREP TO
        NP PRO I
           NU NIL

ITIMING:
2280 CONSES
 0.805 SECONDS
INTERPRETATIONS:
(DO (FOR EVERY X14 / (DATALINE (WHQFILE (NPR* X15 / (QUOTE S10046)))
(NPR* X15 / (QUOTE S10046)) (NPR* X16 / (QUOTE OVERALL)) (NPR* X17
/ (QUOTE H))) : T ; (PRINTOUT X14)))
```

↑L

```
SENTENCE:
(GIVE ME THEM FOR OXYGEN)
PTIMING:
664 CONSES
3.61 SECONDS
PARSINGS:
S IMP
   NP PRO YOU
   AUX TNS PRESENT
   VP V GIVE
      NP PRO THEY
         NU PL
         PP PREP FOR
            NP DET NIL
               N OXYGEN
               NU SG
      PP PREP TO
         NP PRO I
            NU NIL

ITIMING:
2244 CONSES
9.345 SECONDS
INTERPRETATIONS:
(DO (FOR GEN X19 / (DATALINE (WHQFILE (NPR* X20 / (QUOTE S10046)))
(NPR* X20 / (QUOTE S10046)) (NPR* X1 / (QUOTE OVERALL)) (NPR* X2 /
(QUOTE C))) : T ; (PRINTOUT X19)))
```

Examples (2) and (4) illustrate basic complete anaphoric reference. The semantic rule which handles this type of anaphoric reference is D:ANAPHOR. We consider the analysis of (2b) first. (Both requests in example (2) can be found as examples in Appendix G.)

Request (2b) is a "What(Which) is X?" question, which is interpreted by rule SS30. This rule calls for the interpretation of X, in this case "those breccias". The rule D:ANAPHOR recognizes that "those" is anaphoric, and notes that the noun phrase is not modified by either a prepositional phrase or relative clause modifier, in which case, the rule D:SEMI-ANAPHOR would apply. The right hand side of D:ANAPHOR is a sequence of instructions to resolve the anaphorism. The first function called, ANTECEDANT, finds the variable associated with the antecedant of "those breccias", while the calls to ANTEQUANT construct the quantifier to be returned as its interpretation. (Since the antecedant of "those breccias" may be within the scope of some other quantifiers (SCOPEVARS), they must also be included in the interpretation of

"those breccias".) The primary strategy used here in finding the antecedant of "those breccias" is to look for one whose head noun is also "breccia".

The antecedant of "those breccias" in example (2b) is "breccias which contain aluminum". It is a general observation captured by the Semantic Interpreter that a questioned existentially quantified sentence like (2a) implies an intensional noun phrase containing among its restrictions those of the main verb of the request. Thus the interpretation of (2a) produces an intensional noun phrase equivalent to "breccias which contain aluminum?" and it is this intensional object which is the antecedant of "those" breccias.

D:ANAPHOR is also used to resolve the anaphoric reference in example (4b), but the strategy used by ANTECEDANT to find the antecedant of "them" is slightly different. At the sentence level, the rule S:BE-IN2 would match if the antecedant of "them" had the semantic markers (SAMPLE). (The constitutents of S:BE-IN2 are a subject noun phrase which is marked either ELEMENT, OXIDE, ISOTOPE, PHASE or MINERAL, a verb which is either BE, OCCUR or EXIST, a prepositional phrase whose head is marked SAMPLE, and another optional prepositional phrase whose head is marked either PHASE or MINERAL.) When the template (S.PP (AND (EQU 1 IN)(MEM 2 (SAMPLE))) is matched against the top S node, MEM calls the function ANTECEDANT to find out if there is a possible antecedant for "them" which fits this description, i.e. (MEM 2 (SAMPLE)). ANTECEDANT finds "type b rocks" as a possible antecedant noun phrase for "them", which also satisfies the requirement that its head have markers SAMPLE. ANTECEDANT also records on TAGLIST that the antecedant of "them" is X3, the variable associated with the noun phrase "type b rocks". (Where this strategy differs from the one used for finding the antecedant of "those breccias" is in using semantic markers, rather than a specific word like "breccias" as a requirement on the head of the antecedant.) When S:BE-IN2 later calls for the interpretation of the phrase "in them", D:ANAPHOR calls ANTECEDANT which picks off the TAGLIST the antecedant for "them" it found previously. ANTEQUANT then constructs the proper quantifier for it. The following

example illustrates the parsing and interpretation of the two requests in example (4).

```
SENTENCE:
(HOW MUCH TITANIUM IS IN TYPE B ROCKS)
PTIMING:
1315 CONSES
7.655 SECONDS
PARSINGS:
S Q
  NP DET POSTART COMP ADV HOW
                      MUCH
     N TITANIUM
     NU SG
  AUX TNS PRESENT
  VP V BE
     PP PREP IN
        NP DET NIL
           ADJ TYPE/B
           N ROCK
           NU PL

ITIMING:
1089 CONSES
4.595 SECONDS
INTERPRETATIONS:
(FOR GEN X3 / (SEQ TYPEBS) : T ; (CONTAIN' X3 (NPR* X4 / (QUOTE TIO2))
(QUOTE NIL) (HOW)))


SENTENCE:
(HOW MUCH SILICON IS IN THEM)
PTIMING:
1074 CONSES
6.282 SECONDS
PARSINGS:
S Q
  NP DET POSTART COMP ADV HOW
                      MUCH
     N SILICON
     NU SG
  AUX TNS PRESENT
  VP V BE
     PP PREP IN
        NP PRO THEY
           NU PL

ITIMING:
 899 CONSES
5.227 SECONDS
INTERPRETATIONS:
(FOR GEN X3 / (SEQ TYPEBS) : T ; (CONTAIN' X3 (NPR* X6 / (QUOTE SIO2))
(QUOTE NIL) (HOW)))
```

Example (3) illustrates anaphoric reference with the word "ones". "One" and "ones" are peculiar anaphoric pronouns in that they are influenced by the words "too", "also", and "in-addition" in establishing their antecedants. "Them" in examples (1) and (4), on the other hand, is not so influenced. The antecedant of "them" does not change, whether one says "Give me them for oxygen." or "Give me them for oxygen too.". As mentioned previously, when "too" and similar words occur with "ones" or "one", the pronoun's antecedant is the answer set, while without "too", its antecedant is the question set. "One" and "ones" are also peculiar pronouns in that they can occur with determiners, which must also be considered in forming their interpretations: "which ones" has a different interpretation from "the ones".

A noun phrase whose head is the anaphoric pronoun "one" or "ones" is interpreted by the normal DRULES to find its determiner structure. The rules N:ONE and R:ONE are then used to interpret the class of the noun phrase and its restrictions. (It should be pointed out here that "one" and "ones" can also be used in a non-anaphoric, partitive sense, e.g. "Which one of the boys", and in this case, the rules N:ONEOF and R:ONEOF are used to get the class and restrictions of the noun phrase from the head of the partitive construction.)

In the interpretation of example (3b), S:NPQ identifies the parse tree as a questioned noun phrase, and calls for its interpretation with typeflag REFS? We try to interpret it normally, and not as a topic, and in doing so, match the DRULE D:WHQ-PL to the node. D:WHQ-PL calls for the class and restrictions in making up its interpretation, and the rules N:ONE and R:ONE are invoked. The right-hand side of N:ONE calls ANTECEDANT to find the antecedant of "one" and return its associated variable. ANTEQUANT again brings into the interpretation all the quantifiers in whose scope the antecedant of "ones" was located. The function NEWCLASS then adds into the interpretation the class from the antecedant of

"ones" picked off its INTENSION. The rule R:ONE returns all the restrictions on the antecedant of "ones" which did not come from the verb phrase - in this case none. If the word "too" or one like it were present, all the restrictions on the antecedant, including those from the verb phrase, would be returned by NEWPX. The parsing and interpretation of examples (3a) and (3b) is as follows:

```
SENTENCE:
(WHICH COARSE GRAINED IGNEOUS ROCKS HAVE BEEN ANALYZED FOR COBALT)
PTIMING:
884 CONSES
3.647 SECONDS
PARSINGS:
S NPQ
  NP DET WHICHQ
     ADJ COARSE
     ADJ GRAINED
     ADJ IGNEOUS
     N ROCK
     NU PL
     S QREL
       NP PRO SOMETHING
       AUX TNS PRESENT
               PERFECT
       VP V ANALYZE
          NP DET WHR
             N ROCK
             NU PL
          PP PREP FOR
             NP DET NIL
             N COBALT
             NU SG

ITIMING:
 964 CONSES
7.834 SECONDS
INTERPRETATIONS:
(FOR EVERY X13 / (SEQ TYPEBS) : (AND (DATALINE (WHQFILE X13) X13 OVERALL
(NPR* X15 / (QUOTE CO))) (AND T T)) ; (PRINTOUT X13))

↑L
```

```
SENTENCE:
(WHICH ONES HAVE BEEN ANALYZED FOR STRONTIUM)
PTIMING:
734 CONSES
2.899 SECONDS
PARSINGS:
S NPQ
  NP DET WHICHQ
     PRO ONE
     NU PL
     S QREL
       NP PRO SOMETHING
       AUX TNS PRESENT
               PERFECT
       VP V ANALYZE
          NP DET WHR
             PRO ONE
             NU PL
          PP PREP FOR
             NP DET NIL
             N STRONTIUM
             NU SG

ITIMING:
1581 CONSES
7.131 SECONDS
INTERPRETATIONS:
(FOR EVERY X16 / (SEQ TYPEBS) : (AND (AND T T) (DATALINE (WHQFILE
X16) X16 OVERALL (NPR* X18 / (QUOTE SR)))) ; (PRINTOUT X16))
```

Our present anaphorism facility is still very rudimentary and contains a number of deficiencies which will have to be rectified before it can be extended. First, because the intension of a variable and its associated noun phrase is not computed until after the interpretation of the entire request, intra-sentence anaphorism like:

(6) Is the average titanium concentration in S10046 larger than that in S10047?

cannot be interpreted correctly, if at all.

Secondly, we do not save enough of the things in the environment which can serve as antecedants. Consider for example the interchange:

(7) User: Which samples contain magnesium in glass?
LSNLIS: --S10047
USER: Does it contain zirconium too?

To resolve the anaphorism in the above exchange, we should save the information that S10047 is available as an antecedant for the second question. The current system does not do this. It resolves the

anaphorism in a make-shift manner by not insisting on number agreement between anaphorism and antecedant. It takes "samples which contain magnesium in glass as the antecedant for "it", and produces an interpretation which tests whether each sample which contains the above also contains zirconium. That the retrieval component has already found that S10047 is the only sample meeting the above description is ignored by the Semantic Interpreter.

Both of the above problems require the provision of an appropriately varying dynamic environment of possible candidates for antecedants which extends not only between sentences, but within the processing of a single sentence, and includes entities mentioned by both participants in the dialog.

In addition to the above limitations of the current system's "possible antecedant environment", there are many other aspects of the anaphoric reference problem which we have not even begun to investigate. For example, our attempts at partial anaphoric reference have just begun to scratch the surface; much more work is required in this area. Other aspects of anaphorism that have not been incorporated in the system, even on a limited scale, include treatment of words like "other" as anaphoric expressions. For example, the system should be able to find the referent of phrases such as "other rocks" in the following exchanges:

(8) Does S10017 contain magnesium in glass?
Do other rocks contain it?

(9) Which basalts contain aluminum?
Which other rocks contain it?

"Other rocks" in (8) refers to ones other than S10017. "Other rocks" in (9) refers to ones other than the basalts (type A rocks).

Anaphorism is a very interesting and subtle problem, but a crucial one to conveient man-machine communication. More research in this area is required.

Chapter 5

CONCLUSION

5.1 Goals

The long range goals of the LSNLIS project are to develop a system for man-machine communication in natural English which is so natural and convenient that the task of formulating requests for the machine need not distract the scientist from his tasks of hypothesis formation and testing. We would like to be able to understand the scientist's requests in whatever form they occur to him, without requiring him to rephrase them into a constrained and artificial language. Although English is not necessarily the only means of achieving this degree of naturalness, we feel that any artificial language which meets the above criteria will have to share many features of natural language such as vagueness, ambiguity, etc. and that it is more fruitful to try to deal with these problems in English than to try to devise an artificial language which is both as easy for people to think in as English, and at the same time more easy to process by machine.

In addition to the long range goal of making such a system possible in the distant future, we have the additional goal of making some more limited version of the goal available in the next few years. That is, as our knowledge of the linguistic processes involved in the understanding of natural language increases, it should be possible to harness this knowledge into a system which, although more limited than the ultimate goal, will nevertheless perform useful work. We believe that the state of the art in natural language processing is at the point where such applications are possible, and the current LSNLIS prototype is an attempt to carry out such an application.

5.2 Demonstration of the Prototype

At the Second Annual Lunar Science Conference, held in Houston, Texas, January 11-13, 1971, the LSNLIS system was run as a demonstration twice a day for three days. During this time the lunar geologists attending the conference were invited to ask questions of the system. Approximately 110 requests were processed, many of which were questions whose answers would contribute to the work of the requestor and not merely "toy" questions to see what the system would do. These requests were limited to those questions asked which in fact dealt with the data base of the system (many people asked their questions before they could be told what the data base contained) and were restricted to not contain comparatives (which we did not handle at the time, the contract being only 6 months old) by filtering out those requests which contained comparatives. The requests were freely expressed, however, without any prior instructions as to phrasing and were typed into the system exactly as they were asked.

Of 111 requests entered into the system during the three days, 10% of them failed to perform satisfactorily because of parsing or semantic interpretation problems. Another 12% failed due to trivial clerical errors such as dictionary coding errors which were easily corrected during or immediately after the demonstration. The remaining 78% of the requests were handled to our complete satisfaction, and with the correction of the dictionary coding errors and other trivial errors, 90% of the

questions expressed fell within the range of English handled by the system. This performance indicates that our grammar and semantic interpretation rules, which were based on the information of a single geologist informant, did indeed capture the essential details of the way that geologists would refer to the objects and concepts contained in our data base. Examples of the requests which were received are:

(GIVE ME THE AVERAGE SM ANALYSIS OF TYPE A ROCKS)
(WHAT IS THE AVERAGE MODAL CONCENTRATION OF ILMENITE
IN TYPE A ROCKS?)
(GIVE ME EU DETERMINATIONS IN SAMPLES WHICH CONTAIN ILM.)
(GIVE ME ALL K / RB RATIOS FOR BRECCIAS.)
(WHAT BE ANALYSES ARE THERE?)
(GIVE ME OXYGEN ANALYSES IN S10084)
(WHAT SAMPLES CONTAIN CHROMITE?)
(WHAT SAMPLES CONTAIN P2O5?)
(GIVE ME THE MODAL ANALYSES OF P2O5 IN THOSE SAMPLES)
(GIVE ME THE MODAL ANALYSES OF THOSE SAMPLES FOR ALL PHASES)
(DOES S10046 CONTAIN SPINEL?)
(WHAT PHASES DOES S10046 HAVE?)
(WHAT IS THE AVERAGE CONCENTRATION OF IRON IN ILMENITE)
(GIVE ME REFERENCES ON SECTOR ZONING)
(GIVE ME REFERENCES ON ABYSSAL BASALTS)
(GIVE ME ALL IRON / MAGNESIUM RATIOS IN BRECCIAS)
(GIVE ME ALL SC46 ANALYSES)
(WHAT SOILS CONTAIN OLIV)
(GIVE ME ALL OLIV ANALYSES OF S10085)
(WHAT ARE ALL TUNGSTEN ANALYSES?)
(GIVE ME IRON ANALYSES FOR PLAGIOCLASE IN S10022)
(GIVE ME ALL ZIRCONIUM CONCENTRATIONS IN ILMENITES)

5.2 What We Have Accomplished

The current LSNLIS prototype represents a significant step in the direction of the goals discussed above. Within the range of its data base, the system permits a scientist to ask questions and request computations in his own natural English in much the same form as they arise to him (or at least in much the same form that he would use to communicate them to another human being). This is borne out by the performance of the system during the demonstration at the Second Annual Lunar Science Conference. The system answered most of the questions dealing with its data base which were asked by the investigators during the demonstration. The effort required to recast the request into a form suitable for execution in the data base is assumed by the natural English preprocessor, which translates the English requests into compact "disposable" programs which are then executed in the data base. The Englist preprocessor therefore functions as an automatic programmer which will convert the user's request into a tailor-made program for computing or retrieving the answer. The English processor knows the ways in which geologists habitually refer to the elements, minerals, and measurements contained in its data base; it knows the specific details of the data base table layouts; and it knows the correspondence between the two. Thus, for example, the user need not know that the mineral Olivine is abbreviated OLIV in the data base, that the concentrations of Titanium are recorded in terms of the percentage of TiO_2, that the class of rocks referred to variously as "type A", "high alkali", or "fine grained crystalline" are encoded as "TYPEAS" in the data base. These facts are "known" by the natural English processor, and the user's request is automatically translated from the form in which he may ask it into the proper form for the data base. Thus an appreciable portion of the goals of the system are met by the prototype (at least for the current limited data base).

5.3 Where We Stand

Although our current system does indeed exhibit many of the qualities that we have outlined as our goals, we are still far from achieving the goal as stated. The knowledge that the current system contains about the use of English and the corresponding meanings of words and phrases is limited to those English constructions which pertain to the system's data base of chemical analysis data; (which has a very limited and simple structure). Indeed this data base was chosen as an initial data base because its structure was simple and straightforward. In order to incorporate additional data bases into the system, it will be necessary to provide the system with information about the ways that the users will refer to those data bases in English, the vocabulary they will use, the ways they will use that vocabulary, and the "meanings" of the words and constructions in terms of the data base tables. For some tables, (those whose structure is as simple and direct as the chemical analysis table) this process may be a direct extension of the current facility and may require only the addition of new semantic rules for interpreting the new words and constructions. For other applications, however, this will require much greater sophistication in both the linguistic processing and the underlying semantic representations and inference mechanisms. One type of data which will require considerable advancement in the state of the art is the representation and use of data which describes surface and structural features of the samples. This data does not fit conveniently into a table or a paradigm, and the techniques for storing it, indexing it, and providing access to it for retrieval and inference remain to be developed. Indeed, it is in the handling of such information that natural language querying may hold its greatest promise, but such potential is as yet undeveloped.

5.3.1 Linguistic Fluency and Completeness

There are two scales which can be used to measure the performance of a system such as LSNLIS. We can call them completeness and fluency. A system is logically complete if there is a way to express any request which it is logically possible to answer from the data base. The scale, of fluency measures the degree to which virtually any way of expressing a given request is acceptable. The two scales of completeness and fluency are somewhat independent in that it is possible to have a fluent system which will accept virtually any variations on the requests which it accepts, but which is nevertheless incomplete. Likewise, a system may be logically complete but very restricted in its syntax. A natural language system which is incomplete cannot answer certain questions, while such a system that is not fluent is difficult to use.

5.3.1.1 Fluency of LSNLIS

The LSNLIS prototype is quite fluent in a few specific constructions. It will recognize a large number of variations on requests of the form "give me all analyses of constituent x in phase y of sample z." It knows many variations of "give me" and many different variations on "analysis". However, there are other requests which (due to limitations in the current grammar) must be stated in a specific way in order for the grammar to parse them and there are others which are only understood by the semantic interpreter when they are stated in certain ways. Most of the limitations of fluency in the current system are simply due to the fact that the necessary grammar rules and semantic interpretation rules have not been put into the system. Continued development of the grammar and semantic rules will result in continued improvements in fluency, and there is no visible ceiling other than an economic one to the fluency which can be achieved.

5.3.1.2 Completeness of LSNLIS

The criteria for logical completeness is a level of achievement that is not generally met by currently available data management systems using artificial request languages, much less by a system that recognizes natural language. The request language used for the retrieval component of LSNLIS fares better than most data management systems in this respect since it is fundamentally an extension of the predicate calculus of quantificational logic, but there are still some extensions which the language requires in order to fully achieve logical completeness. In addition to this incompleteness of the formal request language, there are limitations in the logical completeness of the subset of English handled by the system. This arises largely from the difficulties of parsing conjunction constructions in English, but there are also problems in the ambiguity of the scopes of quantifiers. However, the subset of English which is currently handled is adequate for expressing most questions which have arisen in practice, and with some further work on conjunctions should become a very convenient language to use.

5.4 Problems for Further Research

5.4.1 Modifier Placement

The semantic rules for the interpretation of the queries in the current system are written in a fairly powerful format which allows a great deal of flexibility. However, there are a number of aspects of the problem which have been surmounted in the prototype by brute force, or by ad hoc procedures. One of these is the syntactic ambiguity of modifiers, as in "Give me the average analysis of breccias for all major elements." In this sentence, there are three syntactic possibilities for the modifier "for all major elements" (it can modify "breccias", "analysis", or "give"). In this case, our understanding of the semantics of the situation tells us that it modifies "analysis", since one can analyze a sample

for an element, and "breccias for all major elements" doesn't "make sense." Without a similar semantic understanding of the situation, the computer has no criteria to select which of these three cases to use. We have in our present system, embodied in the semantic rule for interpreting analyses, the equivalent of the knowledge that "one can analyze a sample for an element." Unfortunately, this information is not in a format which makes it conveniently available to the parser for use in deciding where to put the prepositional phrase. The parser in our present system, therefore, uses a crude consistency check between verbs and the prepositional modifiers they may take to make an initial placement of modifiers. Since this approximation may sometimes be in error or may not entirely determine modifier placement, the semantic rules have been made smarter in order to find the modifier "for all major elements" when interpreting the phrase "average analysis of breccias for all major elements" even though it appears as a modifier of "breccias" and not where it should be. This mechanism, while adequate for the present level of the system, carries inherent difficulties, since it is now possible for several semantic rules to use the same modifier for different purposes.

5.4.2 Retrieval Component

Another area of research has to do with the retrieval component to which the natural language processor interfaces. To take full advantage of the natural language communication, it is clear at this point that there are requirements on the facilities which the retrieval component must possess. For example, the natural language processor contains a facility for dealing with a number of anaphoric expressions (such as pronouns) and filling in their antecedents. That is, when the semantic interpreter produces an interpretation of a noun phrase (a potential antecedant), it remembers that interpretation together with the syntactic structure of the English phrase. Subsequently, when an anaphoric expression is encountered

which could take this phrase as an antecedant, this semantic interpretation is filled in for the anaphoric expression. However, the semantic interpretation is merely a form which enables the retrieval component to compute the actual members of the set denoted. Unless the retrieval component has remembered the result of its previous computation, it will have to perform the computation again. There are thus two problems in dealing with anaphoric expressions--one is recognizing them and identifying the antecedant, and the second is remembering the actual set of data base objects denoted by the antecedant. If the computation which determines the extension of a phrase is sufficiently simple and cheap, then it is advantageous to compute it over again, rather than to save the result. Only if the computation is complex or expensive is it worth the cost in memory space to store the result. Thus the retrieval component needs some appropriate mechanism for determining whether to save such results. (It would be nice if the system were smart enough to know which types of computations might be used as antecedants. It is not clear, however, that there is any recognizable feature which distinguishes such computations.)

5.4.3 Optimizing the Retrieval Expression

At the present, the retrieval programs which are written by the English language processor contain a number of inefficiencies that are due to the way they were generated from natural English, and which would be avoided by a human programmer. Although the cost savings of having the program produced automatically (compared to paying a human programmer to write it) will more than offset the additional cost of computer time in all but extremely long computations, there are undoubtedly improvements that could be made by imposing an optimization phase between the query generated by the semantic interpreter and the actual retrieval operation. The situation is very analogous to the early Fortran compilers

(and many that are still being written) where the quality of code generated by the compiler was inferior to that produced by a human programmer, but the savings in programmer time more than offset the costs associated with the inefficiency for most programs—especially if the program was not to be run many times. In our case, the programs which are constructed will be executed only once, and so the cost of a human programmer could not be amortized over many runnings.

5.4.4 Semantic Representation

The previously mentioned problem areas have all dealt with essentially efficiency questions that would make a system with the current capabilities more economical to run. There are, however, more serious problems having to do with the limitations of the system's current capabilities. Although the procedural approach to semantics and meaning that has been taken here appears to be generalizable to any concept admisable to empiricist philosophy, the fact remains that there are many English constructions for which no effective procedural characterizations have yet been formulated. For example, the linguistic and semantic understanding of processes as fundamental as adverbial modification and mass nouns remain very much obscure and no effective mechanical semantics exists for such concepts.

As discussed previously, the current LSNLIS deals only with extensional inferences that can be computed from well-formatted data bases. The ability to deal with more complex types of data entities--especially descriptions of shape and textural features of the lunar samples will require the use of intensional inference procedures and will raise as a more pressing issue the question of appropriate notational representations and structures for these intensional entities. In short, much basic research in the semantics of natural language remains to be done before a fully general LSNLIS

can become a reality.

5.5 Prospects for the Near Future

As we have just pointed out, there are still many technical and theoretical problems yet to be overcome before the long term goals envisioned by this project can be achieved. However, we feel that the language processing technology that is embodied in the current prototype is such that certain types of limited applications could be feasible in the near future. In those areas where the semantically relevant concepts can all be formally specified in terms of well-formatted data bases such as the chemical analysis data base, and where only English querying and not English updating and data input are required, then the language processing techniques embodied in LSNLIS are capable of providing a fluent language understanding system which removes almost all of the burden of learning artificial conventions from the user. Moreover, the time required for processing requests in the current LISP implementation (approx. 30 seconds of cpu time per request on a hardware paged PDP-10) could easily be cut by an order of magnitude by careful implementation in a language such as FORTRAN or in machine language. At such a level, the cost of such processing would not be exorbitant.

5.6 Summary

The LSNLIS project has made significant progress in its two years of development. We now have a working prototype which demonstrates many of the features which were the objectives of the project and which demonstrates the technical feasibility of natural English querying in the NASA MSC and other similar environments. The system enables a working scientist to ask questions and request computations in a natural and convenient medium--his own natural language--in much the same form in which they arise to him, with the effort required to recast his request into a form suitable for execution in the data base being assumed by the system.

Appendix A

THE LSNLIS USER'S GUIDE

Negotiating with TENEX

An experimental LSNLIS system is currently operational on the BBN-TENEX Time-Sharing System in Cambridge, Mass. In order to use the system, it is necessary to log into the TENEX system. This is done as follows: After establishing a telephone connection with the computer by dialing the computer's number from a data set or acoustically coupled teletype, the TENEX system will type something like:

```
BBN TENEX 1.21.00 5-APR-71 EXEC 1.28
@
```

The "@" sign is the TENEX executive's symbol which indicates that it is waiting for the user to type something. The user should now type:

```
LOGIN WARNER
```

followed by a space, followed by a secret password (which will not print on the teletype), followed by another space, followed by an account number, followed by a carriage return. (The password and account number will be given to authorized users.) For a hypothetical account number 777777, the line on the teletype would look like:

```
@LOG WARNER 777777
```

If you have logged in successfully, the system will type some information relating to your teletype and job number, and will type another "@" waiting for your input. If not, it will give an error comment and wait for you to try again. If you do not succeed in logging in within a reasonable period of time, the system will automatically log you out and break the telephone connection.

Once logged in, the user can call the execution of the English preprocessor as described in the following section. He can return to the TENEX executive at any time by typing control C (i.e. by depressing the control key on the teletype and typing C). To log out of TENEX at the end of the session, return to the TENEX executive and type:

LOGOUT

followed by a carriage return. The system will type some accounting information and automatically break the telephone connection.

Note: If for some reason the telephone connection should be broken accidentally by some difficulty with the telephone line or for any reason other than a normal logout, the job will be held by the TENEX system in a "detached" status and can be resumed. This can be done by dialing up the machine again and instead of typing LOGIN, type ATTACH (space) (password) (space) (the job number that the system assigned you when you logged in)(carriage return). If you are successful, TENEX will type "@" with no further comment and you will be reattached to your old job. If you had lost the connection while in the English preprocessor subsystem, you can resume it by typing CONTINUE followed by a carriage return. If you have any trouble reattaching, call BBN by telephone. A detached job continues to be charged for computer hookup time until you reattach to it and log it out normally.

Using the English Preprocessor

After logging into the TENEX system, the user enters the English preprocessor by typing:

```
RUN DEMSYS.SAV
```

followed by a carriage return. The system will then type a comment something like:

```
BBN LISP-10 11-31-70
```

and will then type a left arrow indicating that LISP is waiting for input. The user should then type:

```
SETUP(LOWFORK.SAV)
```

This sets up the lower (retrieval) fork which contains the data base. When the system again types the left arrow, the user should type:

```
TALKER()
```

to invoke the English preprocessor executive. Notice that LISP will echo a carriage return as soon as the parentheses in its input string ballance. TALKER will identify itself, and will then proceed to accept queries for processing or LISP commands for execution. The former consist of English sentences enclosed in parentheses, while the latter consist of LISP commands followed by arguments enclosed in parentheses.

TALKER indicates that it is waiting for input by typing its "system symbol", which consists of two asterisks.

To leave TALKER and return to the TENEX executive at the end of the session, one can either type control C as described before, or one can type the LISP command.

```
LOGOUT()
```

Control Characters in LISP

There are a number of special control characters which make life easy in the interactive LISP system in which the English processor runs. These characters are typed by depressing the control key on the teletype and typing the corresponding character. If printed on the teletype, a control character is preceeded by an upward arrow, however, most of the control characters do not print when they are typed, but cause a side effect. The following characters are useful.

Control A deletes the preceeding typed character.
    It indicates the deleted character by echoing a backwards slash followed by the deleted character.

Control Q deletes the current contents of the input buffer.
    (Generally the input buffer is the same as the current typed line--the exception being automatic carriage returns generated by the system when the user types beyond the end of the line. These exceptions are indicated by two asterisks beginning the new "line".)

Control R retypes the current contents of the input buffer
    (useful when echos from control A have made the current line unreadable).

Control D aborts whatever you are doing, and returns to the top level LISP executive. (useful whenever you get into trouble or want to discontinue a sentence and type another sentence. It throws you out of TALKER, however, and it is necessary to retype TALKER()).

Control E is less drastic than Control D and will usually abort a sentence and return to TALKER to await another sentence. It should usually be used before trying Control D. Control E will not break out of an embedded help loop, however.

Control C interrupts whatever you are doing and returns to the TENEX executive under which the LISP system runs. This is used in order to return to TENEX to logout, but can also be used to return to TENEX for any other reason. (The interrupted LISP system can be continued by typing

CONTINUE to the TENEX executive as long as it has not been supplanted by a call to some other subsystem.)

CAUTION:

Typing control characters D or E while the lower fork is operating will yank control away from the lower fork and return it to the language processor fork at the top level, leaving the file buffers open and exiting from the TALKER routine. If this happens, the user should type EXECUTE(NIL) to close the file buffers in the lower fork (and type out whatever had been written into the HITFILE buffer before the interrupt) and then type TALKER() to reenter the language processing executive.

Entering Queries for Processing

Queries are entered into the LSNLIS system as normal English sentences enclosed in parentheses. For example:

**(GIVE ME ALL ANALYSES OF SAMPLE S10046)

(TALKER types the double asterisks to indicate that it is ready for input.) Terminal punctuation is optional.

The system understands such concepts as "give me ...", "analyses of a sample for an element in a phase", "modal analyses", "Potassium / Rubidium ratios", "average analyses", etc. It knows both the full chemical element names and their abbreviations, it knows which of the elements are measured in their oxide form as opposed to the elemental form, and it knows a number of variant names for many minerals and rock types. Thus, the user need not concern himself with knowing the particular standard form of the mineral or element names used in the data base, since this standardization takes place during the semantic interpretation of his request. Also, the system makes an effort to allow for all of the reasonable paraphrases of a given request so that the user need not concern himself unduly with the problem of formulating his request in a way which will be acceptable to the system. All of the expressions "Olivine analyses of Sample S10046 for Hydrogen", Hydrogen analyses of Sample S10046 in Olivine", "analyses of S10046 for Hydrogen in Olivine", etc. are equivalent.

In addition, the system knows reasonable default assumptions when the requestor fails to mention the phase or element of concern in his request. If he fails to mention the phase, the overall phase (i.e. the entire sample) is assumed. Likewise, if he fails to mention a specific element, then a quantification of the request over the major elements is assumed.

Depending on the setting of a number of mode variables, the system can display various intermediate results in the course of the processing. It can show the time spent in parsing and in interpreting, the parse tree that results, the intermediate semantic interpretation, and finally the answers that are generated by the sentence. In the normal mode, only the interpretation and the answers will be displayed. However, any of the other displays can be obtained by setting the corresponding mode variable to "T" (the LISP system's symbol for "true") and they can be turned off again by setting the mode variable to "NIL" (the LISP system's equivalent of "false"). For example, the commands:

```
SETQ(PPRINT T)
SETQ(ITIMEFLAG NIL)
```

will set the mode variable PPRINT to "T" indicating that parsings should be printed and will set ITIMEFLAG to "NIL" indicating that the interpretation time is not be be printed. The mode variables which govern these functions are PPRINT (print parsing), IPRINT (print intermediate semantic interpretation), PTIMEFLAG (time the parsing), and ITIMEFLAG (time the semantic interpretation). There are other mode variables which govern other aspects of the system, but they are not necessary to the typical user.

Encountering Unknown Words

The previous sections cover all of the basic information needed by a user as long as he uses only words that are in the vocabulary of the system, -- currently three or four thousand words. However, it is inevitable that a user, asking unconstrained questions about technical subjects will use words which the system has not previously encountered and are not in its vocabulary. We will therefore give here a brief description of the system's operation in that case.

When the system encounters a word which is not in its dictionary and is not an inflected form of a word in its dictionary (it performs inflectional morphology on the most common types of regularly inflected words), it announces this fact to the user with a comment:

```
I DON'T KNOW THE WORD XXXXXX
PLEASE TYPE ITS DICTIONARY ENTRY
D*
```

(where XXXXXX will be the word in question).

at this point, the system is in a special subsystem executive (which identifies itself with the system symbol "D*") waiting for the user to give it a dictionary entry for the indicated word. The user may at this point do any of several things. If he wants to give up on the sentence and try again from scratch, he can type QUIT followed by carriage return, and the system will return to ask for another sentence. If he knows the correct form for the needed dictionary entry, he can type DDEF followed by the proper dictionary entry to add the word to the dictionary, and then type OK followed by a carriage return to tell the system to continue its parsing. If the user does not know the proper format for

the dictionary entry, he can look at the dictionary entry for a similar word and copy it. The command DICT? followed by a word in parentheses will type the dictionary entry for a word. For example:

```
DICT?(REPORT)
```

would result in a typeout of the dictionary entry for the word "report", which would look like:

```
(REPORT
     N -S)
```

The dictionary entry will print out with indenting for easier reading, but the indenting is not necessary for a dictionary entry which the user types in. To type in this same dictionary entry, the user would type:

```
DDEF(REPORT N -S)
```

If the word which the system requests is the root form of the word or if it is an inflected form of a word which undergoes regular inflection, then the user need only give a dictionary entry for the root of the word. If, however, the word is an inflected form of a word which does not undergo regular inflection, then he should give entries for both the root word and the inflected form. The following interchange is an example:

```
I DON'T KNOW THE WORD MICE
PLEASE TYPE ITS DICTIONARY ENTRY
D*DDEF(MOUSE N IRR)
MOUSE
D*DDEF(MICE N (MOUSE (NUMBER PL)))
MICE
D*OK
```

Here, the computer typed everything except the lines that begin with "D*", and the computer typed the "D*"'s which begin those lines. For more complete information on the format for dictionary entries, see the writeup on dictionary formats.

One frequent source of words which are not in the dictionary are misspelled words. If a misspelling is not caught by the user at the time he types it, the system will generate an "I DON'T KNOW THE WORD" comment and wait for a dictionary entry. If this is the case, the user can change the word to the correct word by typing CHANGEWORD followed by the new word or words enclosed in parentheses. For example:

```
I DON'T KNOW THE WORD MODOL
PLEASE TYPE ITS DICTIONARY ENTRY
D*CHANGEWORD(MODAL)
```

The system will respond by typing the remainder of the sentence to be parsed with the new substitution. Note that CHANGEWORD can be used to delete a word (by including no words in the parentheses) or to replace a single word by several (by including several words in the parentheses). Its use is in no way restricted to correcting spelling errors, however, and it can conveniently be used to substitute an equivalent word to see if the system knows it. (If not, the system will again respond with an "I DON'T KNOW THE WORD" message.) When the user is satisfied with the change, typing OK will cause the system to resume parsing on the changed string.

Logging Out

Although this information has already been covered in previous sections, I will cover it again here for easy reference. When the user has completed a session with the system and is ready to log out, he must first return to the TENEX executive. He can do this either by typing LOGOUT() or by typing control C. When the system responds with an "@", he need only type LOGOUT followed by a carriage return, and the system will automatically logout and break the telephone connection.

Appendix B.

The Transition Network Grammar

```
  (PROGN (LISPXPRIN1 (QUOTE "FILE CREATED ")
                     T)
         (LISPXPRIN1 (QUOTE "12-JUN-72 21:54:28")
                     T)
         (LISPXTERPRI T))
  (LISPXPRINT (QUOTE ANNGRAMCOMS)
              T)
  (RPAQQ ANNGRAMCOMS ((G: NASAGRAMMAR)))
(LISPXPRINT (QUOTE (G: NASAGRAMMAR)) T)
(RPAQQ NASAGRAMMAR (COMPL/ COMPL/NTYPE COMPL/S FOR/FOR FOR/NP FOR/TO
ING/BY NP/ NP/, NP/,ESP NP/,NP NP// NP/ADV NP/ART NP/AVG NP/DET NP/HEAD
NP/HELDPART NP/MORE NP/N NP/NP NP/NP: NP/ORD NP/QUANT NP/R
NP/SUPERLATIVE NP/SUPERSET NPR/ NPR/NPR NPR/TITLE NPU/; NPU/;NP PAREN/
PAREN/PAREN PP/ PP/NP PP/PREP PP/QDET QUANT/ QUANT/QUANT QUANT/UNIT
R/ R/NIL R/PREP R/WH S/ S/; S/;S S/AUX S/DCL S/HOW S/IMP S/NO-SUBJ
S/NP S/Q S/QDET S/QP1 S/QP2 S/S S/SADV S/THERE S/VP VP/ADJ VP/ADJ-COMP
VP/AGT VP/COMP-ADJ VP/HEAD VP/MORE VP/NP VP/V VP/VP))
(DEFINEG


(COMPL/
   (WRD FOR T
        (TO FOR/FOR

           (* START OF COMPLEMENT NETWORK;
           TRANSFERS TO THE PROPER STATE FOR THE IDENTIFIED
           COMPLEMENTIZER.)

))
   (WRD THAT T
        (SETQ NTYPE THAT)
        (TO COMPL/NTYPE))
   (WRD THAN T
        (SETQ NTYPE THAN)
        (TO COMPL/NTYPE))
   (JUMP FOR/NP T
        (COND
          ((NULLR SUBJ)
           (SETR SUBJ (BUILDQ (NP (PRO SOMETHING))))))))


(COMPL/NTYPE
   (PUSH S/ T
        (SETR S *)

           (* LOOK FOR A COMPLETE S WHEN THE COMPLEMENTIZER
           (IN NTYPE) SO SPECIFIES)


        (TO COMPL/S)))
```

```
(COMPL/S
   (POP (BUILDQ (NP + +)
                NTYPE S                                  (* LAST STATE OF
                                                         COMPLEMENT NETWORK;
                                                         POPS A COMPLEMENT NP
                                                         STRUCTURE.))

        T))

(FOR/FOR
   (PUSH NP/ T
        (SETR SUBJ *)

            (* IF THE COMPLEMENTIZER IS 'FOR', LOOK FOR THE
            SUBJECT NP OF A FOR-TO COMPLEMENT)

        (TO FOR/NP)))

(FOR/NP
   (WRD TO T
        (TO FOR/TO                                       (* LOOK FOR 'TO' OR 'NOT
                                                         TO'.)))
   (CAT NEG (NULLR NEG)
        (SETR NEG *)
        (TO FOR/NP)))

(FOR/TO
   (PUSH VP/V (CHECKF V UNTENSED

            (* IF 'TO', 'NOT TO', OR 'FOR' + NP WAS FOUND, LOOK
            FOR THE REMAINDER OF THE FOR-TO COMPLEMENT.)

)
        (SENDR SUBJ (GETR SUBJ))
        (SENDR OBJ (GETR OBJ))
        (SENDR NEG (GETR NEG))
        (SENDR TNS (GETR TNS 1))
        (SENDRQ TYPE FOR-TO)
        (SETRQ NTYPE NOM)
        (SETR S *)
        (TO COMPL/S)))
```

```
(ING/BY
   (PUSH NP/ T
        (SETR SUBJ *)
        (TO VP/VP

           (* IF THE SUBJECT WAS NOT PROPERLY DETERMINED IN A
           POSS-ING COMPLEMENT, LOOK FOR IT HERE.)

)))


(NP/
   (CAT DET T

         ((GETF POSSPRO                                    (* START OF THE NP
                                                           NETWORK.))
          (ADDL ADJS (BUILDQ (POSS (NP (PRO *)))))
          (SETRQ DET THE

          (* IF THE DETERMINER IS A POSSESSIVE PRONOUN
          (MY, YOUR), CONSTRUCT THE POSSESIVE MODIFIER AND USE
          'THE' FOR THE DETERMINER)

))
          (T (SETR DET *)))
        (TO NP/ART))
   (CAT PRO T
        (SETR N (BUILDQ (PRO *))                           (* A PRONOUN MAY PICK UP
                                                           PP MODIFIERS IN NP/HEAD)
                )
        (SETR NU (GETF NUMBER))
        (TO NP/NP))
   (MEM (WHETHER IF)
        T
        (SETR NTYPE *)
        (TO COMPL/NTYPE

           (* CONSTRUCT THE COMPLEMENT STRUCTURE FOR SENTENCES
           SUCH AS 'I DON'T KNOW WHETHER HE LEFT.')

))
   (CAT NEG (NULLR NEG)
        (SETR NEG *)
        (TO NP/))
   (JUMP NP/ART (OR (WRD MOST)
                    (NOR (CAT (DET PRO NEG)

           (* SINCE A PRONOUN OR DETERMINER IS NOT REQUIRED TO
           BEGIN AN NP, CONTINUE PROCESSING.)


                         )
                    (WRD (WHOSE WHO WHAT WHETHER IF))))))
```

```
(NP/,
   (MEM (ETC. ETC)
        T
                                                  (* AFTER A COMMA AT THE
                                                  END OF A NP.)
        (ADDR BODY *                              (* 'ETC' FILLS OUT A
                                                  CONJOINED SERIES.))
        (SETRQ CONJ AND)
        (TO NP/,NP))
   (CAT ADV (RFEAT TRANSADV)
                                                  (* A TRANSITIVE ADVERB
                                                  ('PARTICULARLY',
                                                  'ESPECIALLY') MAY
                                                  INTRODUCE A POST-NOMINAL
                                                  MODIFIER.)
        (SETR ADV *)
        (TO NP/,ESP))
   (CAT CONJ (NOR (WRD ,)
                  (GETR CONJ))
                                                  (* A CONJUNCTION CAN
                                                  LEAD INTO THE FINAL ITEM
                                                  IN A SERIES.)
        (SETR CONJ *)
        (TO NP/,))
   (PUSH NP/ T
                                                  (* LOOK FOR THE NEXT
                                                  ITEM IN THE SERIES.)
        (SENDR NPLIST T)
        (ADDR BODY *)
        (TO NP/,NP)))

(NP/,ESP
   (PUSH NP/ T
        (ADDL NMODS (BUILDQ (ADVP (ADV +)
                                  *)
                            ADV)                  (* ANALYZE THE
                                                  POST-NOMINAL MODIFIER
                                                  INTRODUCED BY THE
                                                  TRANSITIVE ADVERB)
        )
(COND
  ((NULLR PPFLAG)
    (SETRQ PPFLAG T)))
(TO NP/HEAD)))
```

```
(NP/,NP
   (WRD , T
        (TO NP/,

           (* AN ITEM IN A COMMA-SPLICED NP SERIES HAS BEEN
           FOUND. IF THE NEXT WORD IS ',', LOOK FOR SUBSEQUENT
           ITEMS. OTHERWISE, POP THE CONJOINED STRUCTURE.)

))
   (POP (COND
          ((AND (NULLR CONJ)
                (WRD (AND OR)
                     (CADAR (LAST (GETR BODY)))))
            (PROG (BODY CONJ LAST TEMP)

           (* IF NO CONJ WAS FOUND AT THIS LEVEL BUT THE LAST
           NP IN THE SERIES WAS ITSELF A CONJOINED NP WITH A
           CONJ, PROMOTE THE CONJ UP TO THIS LEVEL:
           (NP1 NP2 (NP AND NP3 NP4)) -->
           (NP AND NP1 NP2 NP3 NP4))


                  (SETQ BODY (APPEND (GETR BODY)))
                  (SETQ LAST (LAST BODY))
                  (SETQ CONJ (CADR (SETQ TEMP (CAR LAST))))
                  (RPLACD LAST (CDDDR TEMP))
                  (RPLACA LAST (CADDR TEMP))
                  (RETURN (CONS (QUOTE NP)
                                (CONS CONJ BODY)))))
           (T (BUILDQ (@ (NP #)
                         +)
                      (COND
                        ((GETR CONJ))
                        (T (QUOTE OR))

           (* IF THE LAST ITEM WAS NOT A CONJOINED NP AND THERE
           WAS NO CONJ AT THIS LEVEL, INSERT 'OR')
                      )
                      BODY)))
           (OR (GETR CONJ)
               (WRD (AND OR)
                    (CADAR (LAST (GETR BODY))))
               (NULL STRING))))
```

```
(NP//
   (CAT N T
        (SETR N (BUILDQ (N + / (N *))
                        N)                          (* FIND THE SECOND TERM
                                                    IN A RATIO)
              )
        (TO NP/DET))
   (PUSH NPR/ T
        (SETR N (BUILDQ (N + / *)
                        N))
        (TO NP/DET)))


(NP/ADV
   (CAT ADJ T
        (ADDL ADJS (BUILDQ (@ (ADJP)
                              #
                              ((ADJ *)))
                           (REVERSE (GETR ADVS)))

           (* AN ADVERB HAS BEEN FOUND AFTER THE DETERMINER
           STRUCTURE HAS BEEN BUILT, OR AFTER 1 OR MORE
           PRENOMINAL MODIFIERS HAS BEEN PROCESSED.
           A SEQUENCE OF ADDITIONAL ADVERBS IS ALLOWED, UNTIL
           THE ADJECTIVE THEY MODIFY IS FOUND, COMPLETING THIS
           PARTICULAR PRENOMINAL MODIFIER.)


              )
        (TO NP/DET))
   (CAT ADV T
        (ADDL ADVS (BUILDQ (ADVS *)))
        (TO NP/ADV)))
```

```
(NP/ART
   (CAT ORD T
        (SETR POSTART (BUILDQ ((ORD *)))

           (* AN ARTICLE (POSSIBLY NULL) HAS BEEN FOUND;
           LOOK FOR AN OPTIONAL ORDINAL MODIFIER)


              )
        (TO NP/ORD)


           (* THE NEXT TWO ARCS CATCH DEFINITELY DETERMINED
           INFLECTED ADJECTIVES LIKE: "THE MOST ANCIENT ROCK",
           "THE OLDEST ROCK", "THE MORE ANCIENT ROCK")
)
   (JUMP NP/SUPERLATIVE (AND (CAT ADJ)
                             (WRD THE DET)))
   (WRD (MORE MOST) (AND (GETP (NEXTWRD)
                               (QUOTE ADJ))
                         (WRD THE DET))
        (SETR MORE-MOST *)
        (TO NP/SUPERLATIVE))
   (CAT PRO (WRD ONE)
        (SETR N (BUILDQ (PRO *))                    (* THE PRONOUN "ONE" CAN
                                                    FOLLOW A DETERMINER,
                                                    E.G. "THE ONES WHICH..."
                                                    AND "WHICH ONDS")

              )
        (SETR NU (GETF NUMBER))
        (SETR DET (DETBUILD))
        (SETR HEAD (CADR (GETR N)))
        (TO NP/HEAD))
   (JUMP NP/ORD T))
```

```
(NP/AVG
   (PUSH PP/ (CAT PREP)
        (SENDR V (GETR V))
        (ADDL NMODS *)
        (SETR HEAD (CADR (GETR N)))
        (TO NP/HEAD)

           (* THE FUNCTION WORDS "AVERAGE", "MAXIMUM", "MINIMUM",
           "MOST", AND "LEAST" ARE PARSED AS NP'S, WHEN THEY
           APPEAR IN ADJECTIVE POSITION.
           THE REST OF THE SS NP IS MADE THE OBJECT OF A
           DEPENDENT PP. IT INDICATES THE SET OVER WHICH THE
           FUNCTION IS TO BE APPLIED. E.G. "THE OLDEST ROCK" IS
           ANALYSED AS "THE OLDEST OF THE ROCKS")
   )
      (PUSH PP/PREP (NOT (CAT PREP))
           (SENDRQ PREP OF)
           (SENDRQ NU' PL)
           (SENDR V (GETR V))
           (ADDL NMODS *)
           (SETR HEAD (CADR (GETR N)))
           (TO NP/HEAD)))

(NP/DET
   (WRD (MORE MOST) (AND (GETP (NEXTWRD

              (* HERE AFTER THE COMPLETE DETERMINER STRUCTURE
              (INCLUDING ART, ORD, AND QUANT) HAS BEEN PROCESSED.
              LOOK FOR POSSIBLE PRENOMINAL MODIFIERS
              (ADJECTIVES OR PARTICIPLES
              (WITH ADVERBS)) AND THEN LOOK FOR A POTENTIAL
              HEAD--AN N, NPR, OR GERUND, OR EVEN A POSS-ING
              NOMINALIZATION.)

)
                                 (QUOTE ADJ)

   (* WE RECOGNIZE UNDETERMINED TWO-WORD INFLECTED
   ADJECTIVES. THE THIRD ARC BUILDS AN ADDITIONAL NP
   NODE FOR "AVERAGE", "MAXIMUM", ETC.
   (SEE NP/AVG FOR FURTHER DETAIL.))


                                 )
                           (NOT (WRD THE DET)))
(SETR MORE-MOST *)
(TO NP/MORE))
```

```
(CAT N (WRD (AVERAGE MAXIMUM MINIMUM MOST LEAST))
     (SETR N (BUILDQ (N *)))
     (SETR NU (GETF NUMBER))
     (TO NP/AVG))
(PUSH NPR/ (WRD (SAMPLE ROCK LINE LINE# APOLLO))
     (SETR N *)
     (SETRQ NU SG)
     (TO NP/N)
                                          (* PICKS UP TITLES LIKE
                                          "APOLLO 11", "LINE 5"))
(CAT ADJ T
     (ADDL ADJS (BUILDQ (@ (ADJ)
                           #                (*)
                           )
                        FEATURES))
     (TO NP/DET))
(CAT N T
     (SETR N (BUILDQ (N *)))
     (SETR NU (GETF NUMBER))
     (TO NP/N))
(CAT ADV T
     (SETR ADVS (BUILDQ ((ADVS *))))
     (TO NP/ADV))
(CAT V (OR (GETF PASTPART)
           (GETF PRESPART))
                                          (* PRENOMINAL
                                          PARTICIPLES)
     (ADDL ADJS (BUILDQ (ADJ (PARTICIPLE #))
                        LEX))
     (TO NP/DET))
(CAT V (GETF PRESPART)
                                          (* GERUND HEAD, AS IN
                                          'FREEZE DRYING')
     (SETR N (BUILDQ (N #)
                     LEX))
     (SETRQ NU SG)
     (TO NP/N))
(PUSH S/AUX (OR (CAT (NEG ADV))
                (CHECKF V PRESPART))

(* PUSH FOR A POSS-ING NOMINALIZATION.
('JOHN'S FALLING ...') IF A POSSESSIVE MODIFIER HAS
BEEN FOUND, IT BECOMES THE SUBJECT OF THE COMPLEMENT
SENTENCE, OTHERWISE THE SUBJECT IS 'SOMETHING')
```

```
        (SENDR SUBJ (COND
                    ((WRD POSS (CAAR (GETR ADJS)))
                      (SETQ TEMP (CADAR (GETR ADJS)))
                      (SETR ADJS (CDR (GETR ADJS)))
                      TEMP)
                    (T (SENDR SUBFLAG T)
                       (BUILDQ (NP (PRO SOMETHING))))))
        (SENDRQ TYPE POSS-ING)
        (SETRQ NTYPE NOM)
        (SETR S *)
        (TO COMPL/S))
    (PUSH NPR/ T
        (SETR N *)
        (SETRQ NU SG)
        (TO NP/N)))

(NP/HEAD
   (VIR PP (NPREP)

          (* HERE WHEN THE HEAD OF THE NP HAS BEEN POSITIVELY
          DETERMINED. LOOK FOR POST-NOMINAL MODIFIERS:
          PREPOSITIONAL PHRASES, RELATIVE CLAUSES, TO- AND
          THAT- COMPLEMENTS. PPFLAG IS T AFTER A PP HAS BEEN
          FOUND; IT INSURES THAT SUBSEQUENT REDUCED RELATIVES
          WILL MODIFY THE NEAREST NP.)

        (ADDL NMODS *                                  (* RECOGNIZES FRONTED
                                                       PP'S WHICH BELONG TO THE
                                                       NP))
        (COND
          ((NULLR PPFLAG)
            (SETRQ PPFLAG T)))
        (TO NP/HEAD))
    (PUSH R/ (AND (OR (WRD (WHO WHOM WHOSE WHICH THAT))
                      (AND (WRD (WHICH WHOM WHOSE)
                                (NEXTWRD))
                           (CAT PREP)))
                  (OR (CADR (GETR DET)                 (* THIS RESTRICTION
                                                       DISALLOWS
                                                       NON-RESTRICTIVE RELATIVE
                                                       CLAUSES)
                            )
                      (WRD PL NU)))
        (SENDRQ TYPE REL)
        (SENDR WH (BUILDQ (NP (DET WHR)
                              +
                              (NU +))
                          N NU))
        (SENDR ANAPHORFLG (CADR (GETR N)))
        (ADDL NMODS *)
        (TO NP/R))
```

```
(PUSH R/WH (AND (CAR V)
                (GETR RELVPFLG))
    (! (COND
         ((WRD (WHICHQ HOWMANY)
               (CADR (GETR DET)))
           (SENDR ANAPHORFLG (COND
                   ((WRD ONES (CADR (GETR N)))
                     (GETR ANAPHORFLG))
                   (T (CADR (GETR N))))))))
    (SENDRQ TYPE QREL)
    (SENDRQ RELVPFLG T

       (* THE QUESTIONED NP IS MADE THE DS SUBJECT AND THE
       REMAINDER OF THE SENTENCE IS MADE A RELATIVE CLAUSE
       OF TYPE QREL ON THE SUBJECT)

)
       (SENDR WH (BUILDQ (NP (DET WHR)
                             #
                             (NU +))
                         (COND
                           ((WRD ONES (CADR (GETR N)))
                             (GETR ANAPHORFLG))
                           (T (GETR N)))
                         NU))
       (ADDL NMODS *)
       (TO NP/NP))
  (JUMP NP/NP (OR (NOT *)
                  (AND (WRD TO)
                       (RFEAT INDOBJ V

          (* THE JUMP NP/NP ARC CAN BE TAKEN IN TWO PLACES,
          DEPENDING ON THE REGISTERS AND THE CURRENT WORD)

)
                       (VPREP *)
                       (NOR (NPREP *)
                            (WRD (OF FOR))
                            (WRD IT (CADR (GETR N))))))
                 (LIFTR NPFEATURES (RESUMETAG NP/HEAD))
                 (COND
                   ((GETR PARTFLAG)
                     (LIFTR ANAPHORFLG (GETR N)
                            2

(* IF NP OCCURS IN A PARTITIVE CONSTRUCTION, ITS
HEAD IS LIFTED UP TO THE HIGHER NP FOR FURTHER USE.)


                            )))))
```

```
(PUSH PP/ (CAT PREP)
     (SENDR V (GETR V))
     (ADDL NMODS *)
     (COND
       ((NULLR PPFLAG)
          (SETRQ PPFLAG T)))
     (TO NP/HEAD))
(PUSH FOR/NP (WRD TO)
                                                  (* LOOK FOR
                                                  TO-COMPLEMENTS: 'THE WAY
                                                  TO DO IT...')
     (ADDL NMODS (BUILDQ (COMPL *)))
     (COND
       ((NULLR PPFLAG)
          (SETRQ PPFLAG T)))
     (TO NP/HEAD))
(TST R/NIL (AND (GETR PPFLAG)
                (NOR (WRD (WHAT WHO WHOM WHICH THAT WHOSE))
                     (GETR QDET)
                     (AND (WRD BE (GETROOT * V))
                          (NOT (EQ * (QUOTE BEING))))))))
     (SUSPEND 1)

        (* LOOK FOR REDUCED RELATIVES AFTER SEEING A PP.
        THE SUSPEND MEANS THAT THE ACTIONS
        (INCLUDING THE PUSH) WILL GET DONE AFTER THE
        FOLLOWING JUMP ARC HAS BEEN TAKEN AND LEADS TO A
        BLOCK; THE SUBSEQUENT BACKUP WILL CAUSE THE RELATIVE
        IN 'THE MAN NEAR THE GIRL I SEE' TO MODIFY 'GIRL'
        INSTEAD OF 'MAN' IN THE FIRST PARSE.)

     (SENDRQ TYPE REL)
     (SENDR WH (BUILDQ (NP (DET WHR)
                           +
                           (NU +))
                       N NU))
     (PUSH R/NIL)
     (ADDL NMODS *)
     (TO NP/R))
(JUMP NP/NP (NOR (NOT *)
                 (AND (WRD TO)
                      (RFEAT INDOBJ V))
                 (VPREP *)
                 (NOR (NPREP *)
                      (WRD OF))))

        (* THIS SETS UP THE REGISTER NPFEATURES AT THE LEVEL
        ABOVE THIS SO THAT THE RESUME MACHINERY WILL RETURN
        AN EXTRAPOSED RELATIVE CLAUSE TO THIS POSITION)
```

```
      (LIFTR NPFEATURES (RESUMETAG NP/HEAD))
      (COND
        ((GETR PARTFLAG)
          (LIFTR ANAPHORFLG (GETR N)
                 2))))
   (PUSH R/NIL (NOR (GETR PPFLAG)
                    (WRD (WHAT WHO WHOM WHICH THAT WHOSE))
                    (GETR QDET)
                    (AND (WRD BE (GETROOT * V))
                         (NOT (EQ * (QUOTE BEING))))))
      (SENDRQ TYPE REL)
      (SENDR WH (BUILDQ (NP (DET WHR)
                            +
                            (NU +))
                        N NU))
      (ADDL NMODS *)
      (TO NP/R))
   (PUSH COMPL/ (AND (WRD THAT)
                     (WRD (A THE)
                          (CADR (GETR DET))

         (* FOR NOUNS MARKED FACTN (E.G. 'STATEMENT', 'FACT'),
         THIS ARC LOOKS FOR A THAT-COMPLEMENT
         ('THE FACT THAT I ARRIVED...'))


                          )
                     (RFEAT FACTN HEAD))
      (ADDL NMODS (BUILDQ (COMPL *)))
      (TO NP/NP))
   (PUSH COMPL/NTYPE (RFEAT FACTN HEAD)
                                                  (* LOOK FOR A
                                                  THAT-COMPLEMENT WITH
                                                  DELETED 'THAT')

      (SENDRQ NTYPE THAT)
      (ADDL NMODS (BUILDQ (COMPL *)))
      (TO NP/NP)))

(NP/HELDPART
   (JUMP NP/DET T
                                                  (* A PARTITIVE HAS BEEN
                                                  TAKEN OFF THE HOLD LIST
                                                  AT STATE NP/QUANT.)
                                                  (* LOOK FOR A REGULAR
                                                  HEAD: 'OF THESE HOW MANY
                                                  MEN WENT ...'))
   (JUMP NP/HEAD T

         (* THE HEAD WAS DELETED; INSERT AN APPROPRIATE
         DUMMY: 'OF THESE HOW MANY WENT ...')

      (SETRQ N (PRO ONES))
      (SETRQ NU SG/PL)))
```

```
(NP/MORE
   (CAT ADJ (GETR MORE-MOST)
       (ADDL ADJS (BUILDQ (@ (ADJ)                    (*)
                             #)
                          (COND
                            ((WRD MORE MORE-MOST)
                              (QUOTE (COMPARATIVE)))
                            (T (QUOTE (SUPERLATIVE)))))))
       (TO NP/DET)

          (* NP/MORE RECOGNIZES TWO-WORD INFLECTED ADJECTIVES
          AND PAST PARTICIPLES. E.G. "MORE BRECCIATED",
          "MORE METALLIC", "MOST REPRESENTATIVE")
)
   (CAT V (AND (GETR MORE)
               (GETF PASTPART))
       (ADDL ADJS (BUILDQ (ADJ (PARTICIPLE #)
                               COMPARATIVE)
                          LEX))
       (TO NP/DET)))

(NP/N
   (CAT LIST (AND (WRD SG NU                          (* ADJUST NU FOR AN
                                                      ALTERNATIVE PLURAL
                                                      SPECIFICATION 'BOY
                                                      (S) ')

          (* A TENTATIVE HEAD HAS BEEN FOUND, BUT IT MAY BE
          ONLY THE FIRST PART OF A NOUN-NOUN OR
          NOUN-ADJECTIVE-NOUN SEQUENCE.)

                  )
              (WRD (S ES)
                   (CAR *)))
       (SETRQ NU SG/PL)
       (TO NP/N))
(WRD / T

          (* '/' FOLLOWS THE TENTATIVE HEAD, INDICATING THAT
          IT WAS THE FIRST TERM OF A RATIO)

       (TO NP//))
(CAT POSS T

          (* POSS ('S) MARKS THE PRECEDING HEAD AS A GENITIVE
          MODIFIER ON A HEAD WHICH IS TO FOLLOW.
          SET UP THE PROPER STRUCTURE AND LOOP TO NP/DET.)
```

```
        (SETR ADJS (BUILDQ ((POSS #))
                           (NPBUILD)))
        (SETRQ DET THE)
        (TO NP/DET))
    (JUMP NP/HEAD (CAT PREP)
        (COND
          ((GETR NU')
            (SETR NU (GETR NU')

             (* AS SOON AS WE HAVE SEEN A PREPOSITION, WE KNOW WE
             HAVE SEEN THE HEAD. THE HEAD OF THE PP INDICATING
             THE SET OVER WHICH "AVERAGE", "MAXIMUM", ETC. RANGE
             (SEE NP/DET) IS ALWAYS PLURAL.
             THIS INDICATION IS PASSED DOWN IN THE REGISTER NU',)
                  )))
        (SETR HEAD (CADR (GETR N))))
    (CAT N (NOR (WRD PL NU)
                (EQ LEX (QUOTE BEING)))

             (* A NEW HEAD IS FOUND, IMPLYING THAT THE PRECEDING
             HEAD IS A NOUN-MODIFIER.)

        (ADDL ADJS (BUILDQ (ADJ +)
                           N))
        (SETR N (BUILDQ (N *)))
        (SETR NU (GETF NUMBER))
        (TO NP/N))
    (PUSH NPR/ (NOR (CAT V)
                    (CAT PREP)
                    (NULL STRING))

                                                (* THE NEW HEAD IS A
                                                PROPER NOUN MODIFIED BY
                                                THE OLD HEAD.)
        (ADDL ADJS (BUILDQ (ADJ +)
                           N))
        (SETR N *)
        (SETRQ NU SG)
        (TO NP/N))
    (CAT ADJ (NOT (WRD PL NU))

                                                (* AN ADJECTIVE AFTER A
                                                TENTATIVE HEAD IMPLIES
                                                AN N-ADJ-N STRUCTURE.)
        (ADDL ADJS (BUILDQ (ADJ +)
                           N))
        (ADDL ADJS (BUILDQ (@ (ADJ)
                              #
                              )
                           FEATURES))                (*)
        (TO NP/DET))
```

```
(CAT V (AND (GETF PRESPART)
            (NOT (WRD PL NU))
            (NOT (VPARTICLE * (NEXTWRD)))
            (NOT (AND (GETR N)
                      (EQ LEX (QUOTE BEING))))))

    (* A GERUND HERE IS TAKEN AS THE HEAD--A SUBSEQUENT
    N WILL MOVE IT TO A PARTICIPIAL MODIFIER POSITION.)

    (ADDL ADJS (BUILDQ (ADJ +)
                       N))
    (SETR N (BUILDQ (N #)
                    LEX))
    (SETRQ NU SG)
    (TO NP/N))
(JUMP NP/HEAD (NOT (CAT PREP))
    (SETR HEAD (CADR (GETR N))                    (* SEE EARLIER JUMP
                                                  NP/HEAD ARC FOR
                                                  EXPLANATION OF NU'.)
          )
    (COND
      ((GETR NU')
        (SETR NU (GETR NU')))))
(CAT N (OR (WRD PL NU)
           (EQ LEX (QUOTE BEING)))

    (* A SPECIAL ARC TO HANDLE N-N MODIFIERS WHERE THE
    FIRST NOUN IS PLURAL: 'OPERATIONS RESEARCH',
    'SYSTEMS ANALYSIS'. THIS MIGHT NOT BE A PRODUCTIVE
    PROCESS, IN WHICH CASE THIS ARC IS UNNECESSARY AND
    SHOULD BE REPLACED BY APPROPRIATE COMPOUND
    DICTIONARY ENTRIES.)

    (ADDL ADJS (BUILDQ (ADJ +)
                       N))
    (SETR N (BUILDQ (N *)))
    (SETR NU (GETF NUMBER))
    (TO NP/N)))
```

```
(NP/NP
   (WRD , (NULLR NPLIST)

          (* THIS ARC STARTS OFF A SERIES OF COMMA-CONJOINED
          NP'S, WHICH ARE ANALYZED BY PUSHING FOR NP'S FROM
          WITHIN THE FIRST NP OF THE SERIES.
          NPLIST IS ONLY EMPTY FOR THE TOP-LEVEL NP
          (THE FIRST ONE) OF THE SERIES, SO THAT ALL NP PUSHES
          ARE DONE FROM THE TOP LEVEL, THAT IS, THE SECOND NP
          CAN'T PUSH FOR THE THIRD, THE THIRD FOR THE FOURTH,
          ETC. AT THE TOP-LEVEL, THE SUBSEQUENT ITEMS IN THE
          SERIES ARE COLLECTED IN THE REGISTER BODY.)

       (SETR BODY (LIST (NPBUILD)))
       (TO NP/,))
(CAT LIST T

          (* THIS IS A TRICKY ARC--IT RECURSIVELY CALLS THE
          PARSER TO ANALYZE THE LIST OF NON-RESTRICTIVE
          MODIFIERS, BEGINNING AT STATE PAREN/ IN THE GRAMMAR.
          THE PARSE IS ADDED TO NR, AND IF IT IS NIL, WE
          ABORT.)

       (ADDL NR *

          (* THE ANALYSIS OF THE CURRENT NP HAS BEEN
          ESSENTIALLY COMPLETED. A COMMA AT THIS POINT CAN
          SIGNIFY THAT THIS IS THE BEGINNING OF A SERIES OF
          CONJOINED NP'S (ARC 1), WHILE A PARENTHETIC
          EXPRESSION (A LIST) IS INTERPRETED AS A
          NON-RESTRICTIVE MODIFIER ON THIS NP
          (E.G. 'FIBROUS MATERIALS (ASBESTOS, FIBERGLASS) ').
          A COLON CAN ALSO INDICATE THE BEGINNING OF A SERIES
          OF NON-RESTRICTIVE ITEMS (ARC 3). THE NORMAL CASE,
          HOWEVER, IS TO POP THE NP SO FAR ANALYZED.)

)
          (COND
            ((NULL (CAR (GETR NR)))
              (ABORT)))
          (TO NP/HEAD))
   (WRD : T
                                                  (* THIS ARC HANDLES
                                                  'INFORMATION ON THE
                                                  FOLLOWING: RADAR,
                                                  LASERS...')
          (TO NP/NP:))
```

```
   (POP (GETR POPVAL                                    (* SIMULTANEOUS
                                                        INTERPRETATION OF THE NP
                                                        WILL GO ON IF SIFLAG IS
                                                        T.))
        (AND (SETR POPVAL (NPBUILD))
             (OR (NOT SIFLAG)
                 (INTERP (GETR POPVAL))
                 T))))

(NP/NP:

        (* CURRENTLY, THE ONLY POSSIBILITY AFTER A COLON AT
        THE END OF A NP IS ANOTHER NP
        (PERHAPS A CONJUNCTION OF NP'S))

   (PUSH NP/ T
        (ADDL NR *)
        (TO NP/NP)))


(NP/ORD
   (PUSH QUANT/ (CAT (QUANT INTEGER ADV COMP)

        (* AN ORDINAL INDICATOR, IF PRESENT, HAS BEEN
        ANALYZED. AN OPTIONAL QUANTIFIER CAN FOLLOW: 'FIVE
        MEN', 'MANY PLANES'. THE QUANTIFIER IS ADDED TO THE
        POSTARTICLE STRUCTURE.)


                    )
        (SETR POSTART (BUILDQ (@ + *)
                              POSTART))
        (TO NP/QUANT))
   (JUMP NP/QUANT T))
```

```
(NP/QUANT
   (PUSH PP/ (AND (WRD OF)
                  (OR (GETR POSTART

            (* THE QUANTIFIER OR ORDINAL
            (AND CERTAIN DETERMINERS) CAN BE FOLLOWED BY A
            PARTITIVE CONSTRUCTION, AS IN 'THE LAST OF THE
            MOHICANS', 'FIVE OF THE BOYS', OR 'HOW MANY OF THE
            DOCUMENTS...'. THE PARTITIVE IS USUALLY INTRODUCED
            BY THE PREPOSITION 'OF' (ARC 2), BUT FOR SOME
            DETERMINERS (E.G. 'ALL', 'BOTH') THE PREPOSITION MAY
            BE MISSING (ARC 2); ARC 3 RETRIEVES A PARTITIVE THAT
            WAS ANALYZED AND PUT ON THE HOLD LIST AT STATE
            S/--'OF THE BOYS HOW MANY ...'.
            WHEN THERE IS A PARTITIVE, IT IS ADDED TO THE LIST
            OF NOUN MODIFIERS, AND THE HEAD OF THE NP BECOMES
               THE DUMMY ELEMENT 'ONES'.)

)
                           (WRD (WHICHQ HOWMANY HOWMUCH ALL SEVERAL MOST)
                                DET)))
        (SENDR PARTFLAG T)
        (ADDL NMODS *)
        (SETR DET '(DETBUILD))
        (SETRQ N (PRO ONES))
        (SETRQ NU SG/PL)
        (TO NP/HEAD))
   (PUSH PP/PREP (WRD (ALL BOTH)
                      DET)
        (SENDRQ PREP OF)
        (ADDL NMODS *)
        (SETR DET (DETBUILD))
        (SETRQ N (PRO ONES))
        (SETRQ NU SG/PL)
        (TO NP/HEAD))
   (VIR PP (AND (GETF PARTITIVE)
                (OR (GETR POSTART)
                    (WRD (WHICHQ HOWMANY HOWMUCH ALL SEVERAL)
                         DET)))
        (SETR DET (DETBUILD))
        (ADDL NMODS *)
        (TO NP/HELDPART))
   (JUMP NP/DET T
        (SETR DET (DETBUILD))))
```

```
(NP/R
   (PUSH R/ (WRD (WHO WHOM WHICH THAT))
        (SENDRQ TYPE REL)

           (* A RELATIVE CLAUSE HAS BEEN ANALYZED.
           THIS MAY BE FOLLOWED (OPTIONALLY) BY ANOTHER FULL
           RELATIVE (NOT REDUCED).)

        (SENDR WH (BUILDQ (NP (DET WHR)
                              +
                              (NU +))
                          N NU))
        (ADDL NMODS *)
        (TO NP/R))
   (JUMP NP/NP T))


(NP/SUPERLATIVE
   (CAT ADJ (OR (GETR MORE-MOST)
                (GETF SUPERLATIVE

           (* DEFINITELY DETERMINED INFLECTED ADJECTIVES PARSE
           INTO A HIGHER AND A LOWER NP NODE.
           THE HIGHER NODE CONTAINS THE INFLECTED ADJECTIVE AS
           ITS HEAD; THE LOWER NODE, THE SET OVER WHICH THE
           INFLECTED ADJECTIVE RANGES)

)
                (GETF COMPARATIVE))
        (SETR N (BUILDQ (@ (N)                          (*)
                           #)
                        (COND
                          ((WRD MORE-MORE-MOST)
                            (QUOTE (COMPARATIVE)))
                          (T (QUOTE (SUPERLATIVE))))))
        (SETRQ NU SG)
        (SETR DET (DETBUILD))
        (TO NP/SUPERSET))
(JUMP NP/ORD T))
```

```
(NP/SUPERSET
   (PUSH PP/ (WRD (OF AMONG))
        (SENDR V (GETR V)

           (* SEE NP/SUPERLATIVE ; VIR PP ARC ALLOWS
           SUPERLATIVES TO PICK UP FRONTED PP'S)


                )
        (ADDL NMODS *)
        (SETR HEAD (CADR (GETR N)))
        (COND
          ((OR (NULLR ANAPHORFLG)
               (EQ (GETR ANAPHORFLG)
                   T)
               (SEMNET (GETR ANAPHORFLG)
                       (HEAD (CADDR *)))))
          (T (ADDL NMODS (BUILDQ (PP (PREP OF)
                                     (NP (DET THE)
                                         (N #)
                                         (NU PL)))
                                 (GETR ANAPHORFLG)))))
        (TO NP/HEAD))
(PUSH PP/PREP (AND STRING (NOT (CAT PREP)))
     (! (COND
          ((GETR ANAPHORFLG)
            (SENDRQ DET THE))
          (T (SENDRQ DET NIL))))
     (SENDRQ NU! PL)
     (SENDRQ PREP OF)
     (SENDR V (GETR V))
     (ADDL NMODS *)
     (SETR HEAD (CADR (GETR N)))
     (TO NP/HEAD))
(VIR PP (GETF PARTITIVE)
     (ADDL NMODS *)
     (SETR HEAD (CADR (GETR N)))
     (TO NP/HEAD))
(JUMP NP/HEAD (AND (NOT (WRD OF))
                   (GETR ANAPHORFLG))
     (SETR HEAD (CADR (GETR N)))
     (ADDL NMODS (BUILDQ (PP (PREP OF)
                             (NP (DET THE)
                                 (N #)
                                 (NU PL)))
                         (GETR ANAPHORFLG))))
(JUMP NP/HEAD (OR (WRD OF)
                  (NULLR ANAPHORFLG))))
```

```
(NPR/
   (MEM (SAMPLE ROCK LINE LINE# APOLLO)
        T
        (SETR TITLE *)

           (* START OF THE PROPER NOUN NETWORK,
           EVENTUALLY, THIS WOULD INCLUDE A FULL GRAMMAR FOR
           THE SYNTAX OF PROPER NAMES--TITLES, ABBREVIATIONS,
           INITIALS, ETC. CURRENTLY, WE RECOGNIZE CERTAIN WORDS
           AS TITLES IF THEY ARE FOLLOWED BY A WORD IN THE NPR
           CATEGORY; OTHERWISE, THIS NETWORK WILL ONLY
           RECOGNIZE ISOLATED NPR WORDS AS PROPER NOUNS.)

        (TO NPR/TITLE))
   (CAT NPR T
        (SETR NPR (LIST *))
        (TO NPR/NPR)))


(NPR/NPR
   (POP (BUILDQ (@ (NPR)
                   +)
                NPR

           (* END OF THE PROPER NOUN NETWORK.
           ARC 1 POPS THE APPROPRIATE STRUCTURE;
           ARC 2 INSURES THAT THE SYSCONJ FACILITY WILL NOT BE
           INVOKED AT THIS LEVEL, THAT IS, THAT A CONJUNCTION
           OF NPR'S WILL BE ANALYZED AS A CONJUNCTION OF NP'S
           WITH NPR HEADS, NOT AS A SINGLE NP WITH A CONJOINED
           NPR HEAD.)

                )
        T)
   (CAT CONJ NIL))


(NPR/TITLE
   (CAT NPR T
        (SETR NPR (BUILDQ (+ *)
                          TITLE)

           (* HERE IF A TITLE WORD WAS FOUND.
           PICK UP THE FOLLOWING NPR AND BUILD THE CORRECT
           STRUCTURE.)

           )
        (TO NPR/NPR)))
```

```
(NPU/;
   (CAT CONJ (OR (NULLR CONJ

         (* HERE IF A ';' WAS FOUND IMMEDIATELY AFTER THE
         SUBJECT NP. THIS MARKS THE SENTENCE AS A NOUN-PHRASE
         UTTERANCE CONSISTING OF A SEQUENCE OF CONJOINED
         NP'S. (SEMI-COLON CONJOINING IS NOT ALLOWED WITHIN
         THE NP'S OF A REGULAR SENTENCE.) THE STRATEGY HERE
         IS SIMILAR TO THAT USED FOR SEMI-COLON CONJUNCTION
         AT THE END OF FULL SENTENCES
         (STATES S/; AND S/;S) AND SOMEWHAT RESEMBLES THE
         OPERATION OF COMMA-CONJOINING WITHIN NP'S
         (STATES NP/, AND NP/,NP))

)
                  (EQ * (GETR CONJ)))
         (COND
           ((NULLR CONJ)
             (SETR CONJ *)))
         (TO NPU/;))
   (PUSH NP/ T
         (ADDL NPU *)
         (TO NPU/;NP)))


(NPU/;NP
   (WRD ; T
         (TO NPU/;

            (* A SEQUENCE OF SEMICOLON-CONJOINED NP'S IN AN NPU
            CAN BE FOLLOWED BY A ';', INDICATING THAT ANOTHER
            ITEM IS TO FOLLOW, OR ELSE THE END OF THE STRING
            MUST HAVE BEEN REACHED, IN WHICH CASE THE FINAL NPU
            STRUCTURE IS BUILT.)

))
   (POP (BUILDQ (S NPU (@ (NP #)
                          #))
            (COND
              ((GETR CONJ))
              (T (QUOTE OR)))
            (REVERSE (GETR NPU)))
(NULL STRING)))
```

```
(PAREN/
   (PUSH NP/ T
        (SETR PAREN *

           (* THIS IS THE INITIAL STATE FOR THE GRAMMAR WHICH
           ANALYZES POST NOMINAL PARENTHETIC EXPRESSIONS.
           CURRENTLY, ONLY A NOUN-PHRASE CAN OCCUR AS SUCH A
           NON-RESTRICTIVE MODIFIER, BUT THE GRAMMAR SHOULD BE
           EXPANDED HERE TO INCLUDE SEQUENCES OF ADJECTIVAL
           PHRASES.)

)
        (TO PAREN/PAREN)))

(PAREN/PAREN
   (POP (GETR PAREN)
        T

           (* THE FINAL STATE OF THE PARENTHETIC-EXPRESSION
           GRAMMAR; JUST POP WHATEVER WAS IDENTIFIED.
           OF COURSE, WE MUST HAVE EXHAUSTED THE STRING WITHIN
           THE PARENTHESES.)

))

(PP/
   (CAT PREP T
        (SETR PREP *

           (* FIRST STATE OF THE PREPOSITIONAL PHRASE NETWORK.
           ALL PUSHES TO THIS STATE MAKE SURE THAT THE CURRENT
           WORD IS A PREPOSITION, SO WE CAN OMIT THE TEST
           HERE.)

)
        (TO PP/PREP)))

(PP/NP
   (SPOP (BUILDQ (PP (PREP +)
                     +)
                 PREP NP

           (* HERE AFTER THE PREP AND NP HAVE BEEN FOUND.
           THE PP STRUCTURE IS BUILT AND SPOPPED, THAT IS,
           POPPED TO THE LEVEL DETERMINED BY THE SELECTIVE
           MODIFIER PLACEMENT FACILITY.
           SINCE THE DICTIONARY DOES NOT YET CONTAIN SMP
           FEATURES, THE DEFAULT PLACES THE PP IN THE LOWEST
           CONSTITUENT IT CAN BELONG TO.)

)
        T))
```

```
(PP/PREP
   (WRD : T
        (TO PP/PREP                                    (* AFTER PICKING UP THE
                                                       PREP, FIND THE NP
                                                       PREPOSITIONAL OBJECT.))


            (* IF THE PREP IS FOLLOWED BY ':', SKIP PAST IT AND
            LOOK FOR THE OBJECT IN THE REGULAR WAY.
            E.G. 'INFORMATION ON: RADAR, LASERS...' IS PROPERLY
            ANALYZED IF THE COLON IS IGNORED.)

)
   (CAT QDET (NULLR TYPE

            (* THE CAT QDET AND CAT QWORD ARCS CATCH QUESTIONS
              IN FRONTED PP'S. E.G. "IN WHICH...")

)
        (SETR DET *)
        (TO PP/QDET))
   (CAT QWORD (NULLR TYPE)
        (SETR NP *)
        (LIFTR NP (GETR NP))
        (TO PP/NP))
   (PUSH NP/ T
        (! (COND
             ((GETR PARTFLAG

            (* WE MUST PASS INFORMATION FROM A HIGHER NP INTO
            THE NP WITHIN THE PP INDIRECTLY)

)
                (SENDR PARTFLAG T))))
        (SENDR RELVPFLG (GETR RELVPFLG))
        (SENDR V (GETR V))
        (SENDR DET (GETR DET))
        (SENDR NU' (GETR NU'))

(* NORMALLY, THE OBJECT OF THE PREP WILL BE FOUND AS
AN ORDINARY NP STARTING AT THIS STRING POSITION.)
```

```
        (SETR NP *)
        (TO PP/NP))
    (VIR NP T

            (* THE OBJECT MIGHT HAVE BEEN FRONTED, FOR EXAMPLE,
            BY RELATIVIZATION OR PASSIVIZATION, LEAVING A
            DANGLING PREPOSITION ('THE STORE I BOUGHT IT IN
            ...'); IF SO, THE OBJECT HAS BEEN PLACED ON THE HOLD
            LIST BY PREVIOUS STATES, AND THIS ARC RETRIEVES IT.
            THE RESUME ACTION IS NECESSARY TO DEAL WITH A
            RELATIVE CLAUSE EXTRAPOSED FROM THE FRONTED OBJECT
            AND LEFT IN THIS POSITION ('THE STORE I BOUGHT IT IN
            WHICH USUALLY HAS GOOD PRICES...'))

        (RESUME)
        (SETR NP *)
        (TO PP/NP))
    (VIR ADV (AND (WRD WHERE (CADR *))
                  (WRD (FROM TO AT)
                       PREP))

            (* IF THE DANGLING PREP IS A LOCATIVE ONE AND THE
            WORD 'WHERE' WAS FOUND AND HELD BY PREVIOUS STATES
            (E.G. S/) WE RETRIEVE IT HERE AND BUILD THE
            APPROPRIATE PP STRUCTURE.)

        (SETR NP (BUILDQ (NP (DET WHQ)
                             (N PLACE)
                             (NU SG))))
        (TO PP/NP))
    (VIR ADV (AND (WRD WHEN (CADR *))
                  (WRD AT PREP))

            (* IF WE HAVE A DANGLING TEMPORAL PREP AND WE
            PREVIOUSLY ENCOUNTERED AND HELD 'WHEN', WE BUILD A
            TIME PP.)

        (SETR NP (BUILDQ (NP (DET WHQ)
                             (N TIME)
                             (NU SG))))
        (TO PP/NP)))
```

```
(PP/QDET
   (PUSH NP/ART T
        (SENDR DET (GETR DET)

           (* PUSH FOR THE REST OF THE NP FOLLOWING THE QDET IN
           THE FRONTED PP. E.G.
           "IN WHICH SAMPLES DOES STRONTIUM OCCUR?")


                )
        (SETR NP *)
        (LIFTR NP (GETR NP))
        (TO PP/NP)))

(QUANT/
   (CAT COMP (NULLR ADV

           (* START OF QUANTIFIER NETWORK FOR NP DETERMINER
           STRUCTURE. QUANTIFIER CAN INCLUDE A COMPARATIVE
           ('MORE THAN') OR ELSE JUST BEGIN WITH AN INTEGER OR
           A WORD IN CATEGORY QUANT. CURRENTLY, MOST WORDS IN
           THIS CATEGORY ARE ALSO IN CATEGORY DET, SO THEY
           APPEAR AS DETERMINERS IN THE FIRST PARSES.)

)
        (SETR ADV *)
        (TO QUANT/))
   (CAT QUANT T
        (SETR NUMB *)
        (TO QUANT/QUANT))
   (CAT INTEGER T
        (SETR NUMB *)
        (TO QUANT/QUANT)))

(QUANT/QUANT
   (TST UNIT-TST (MARKER UNIT *)
        (SETR UNIT *

           (* AFTER THE QUANTIFIER HAS BEEN PICKED UP, A UNIT
           OF MEASURE CAN BE SPECIFIED
           ('FIVE GALLONS', 'MORE THAN 3 MM'). POTENTIAL UNITS
           HAVE A UNIT MARKER, WHICH IS TESTED ON THE ARCS FROM
           THIS STATE.)

)
        (SETRQ FLAG MUCH)
        (TO QUANT/UNIT))
   (JUMP QUANT/UNIT (NOT (MARKER UNIT *))
        (SETRQ FLAG MANY)))
```

```
(QUANT/UNIT
   (POP (COND
          ((GETR ADV)
            (BUILDQ ((COMP (ADV +)
                           (@ (NP (INTEGER +))
                              #))
                     +)
                    ADV NUMB (COND
                      ((GETR UNIT)
                        (BUILDQ ((UNIT +))
                                UNIT))
                      (T NIL))
                    FLAG)

                (* END OF QUANTIFIER NETWORK;
                BUILD THE CORRECT STRUCTURE.
                IF A UNIT IS PRESENT, THE STRUCTURE IS FLAGGED WITH
                THE WORD 'MUCH', OTHERWISE 'MANY'.
                ALSO, IF THERE WAS A COMPARATIVE, THE ROOT OF THE
                QUANTIFIER STRUCTURE IS THE NODE 'COMP'.)


                )
                (T (BUILDQ ((@ (NP (INTEGER +))
                               #)
                            +)
                           NUMB
                           (COND
                             ((GETR UNIT)
                               (BUILDQ ((UNIT +))
                                       UNIT)))
                           FLAG)))
          T))


(R/
   (MEM (WHICH THAT WHO)
        T
        (TO R/WH

                (* START OF RELATIVE CLAUSE NETWORK, GIVEN THAT WE
                ARE LOOKING AT A RELATIVE PRONOUN
                ('WHO', 'WHAT', 'WHICH') OR A PREP FOLLOWED BY A
                RELATIVE PRONOUN.)

))
   (WRD WHOM T

                (* FOR 'WHOM', WE KNOW THAT THE WH-NP IS NOT THE
                SUBJECT OF THE RELATIVE CLAUSE--WE HOLD IT TO BE
                PICKED UP LATER.)
```

```
        (HOLD (GETR WH))
        (SETR WH NIL)
        (TO R/WH))
    (PUSH NP/ (WRD WHOSE)

            (* 'WHOSE' MEANS THAT THE WH-NP IS A POSSESSIVE FOR
            AN NP AFTER THE 'WHOSE')

        (SENDR ADJS (BUILDQ ((POSS +))
                            WH))
        (SETR WH *)
        (TO R/WH))
    (CAT PREP T
        (SETR PREP *)
        (TO R/PREP)))

(R/NIL
   (CAT V T
        (COND
          ((AND (GETF PASTPART

            (* HERE TO LOOK FOR A REDUCED RELATIVE--WITHOUT A
            RELATIVE PRONOUN. DETERMINE THE TYPE OF SENTENCE,
            DISPOSE OF THE WH-NP PROPERLY, THEN TRANSFER INTO
            THE CORRECT PLACE IN THE S/ GRAMMAR TO ANALYZE THE
            REST OF THE CLAUSE.)

)
                (VPASSIVE *))
            (HOLD (GETR WH))
            (SETR SUBJ (BUILDQ (NP (PRO SOMETHING)))

            (* A PRESENT PARTICPLE MEANS THE RELATIVE CLAUSE IS
            A PROGRESSIVE SENTENCE WITH THE WH-NP AS SUBJECT;
            A PAST PARTICIPLE MEANS THE RELATIVE CLAUSE IS
            PASSIVIZED, AND THE WH-NP IS HELD AS THE SUBJECT OF
            A PASSIVE SENTENCE USUALLY IS AT STATE VP/V.)

                )
            (SETR AGFLAG T))
          ((GETF PRESPART)
            (SETR SUBJ (GETR WH))
            (SETRQ ASPECT (PROGRESSIVE)))
          (T (ABORT)))
       (SETR V *)
       (TO VP/V))
(PUSH NP/ T

            (* AN NP HERE IS THE SUBJECT OF THE RELATIVE CLAUSE;
            THE WH-NP IS EITHER THE OBJECT, INDIRECT OBJECT, OR
            PREP OBJECT--HOLD IT UNTIL WE CAN DETERMINE WHICH.)
```

```
        (HOLD (GETR WH))
        (SETR SUBJ *)
        (TO S/NP))
    (WRD THERE T
        (SETRQ THERE T)
        (SETR SUBJ (GETR WH))
        (TO S/NP))
    (CAT ADJ T

            (* POST-NOMINAL ADJECTIVES ARE PROCESSED AS REDUCED
            RELATIVE COPULAR SENTENCES,
            'INFORMATION AVAILABLE' IS ANALYZED AS 'INFORMATION
            WHICH IS AVAILABLE')

        (SETR SUBJ (GETR WH))
        (SETRQ V BE)
        (SETR OBJ (BUILDQ (@ (ADJ)
                             #                    (*)
                             )
                          FEATURES))
        (TO VP/NP))
    (CAT ADV T
        (ADDL VMODS (BUILDQ (ADV *)))
        (TO R/NIL)))


(R/PREP
   (MEM (WHICH WHOM)
        T
        (ADDL VMODS (BUILDQ (PP (PREP +)
                                +)
                            PREP WH)

            (* LOOKING AT RELATIVE PRONOUN AFTER THE PREP.
            IF THE RELATIVE PRONOUN IS 'WHICH' OR 'WHAT, THE
            WH-NP IS THE OBJECT OF THE PREP.
            FOR 'WHOSE', THE PREP OBJECT IS A NP BEGINNING WITH
            'WHOSE' AND HAVING THE WH-NP AS A POSSESSIVE
            MODIFIER. IN EITHER CASE, A PP IS BUILT AND ADDED TO
            THE VERB MODIFIERS OF THE RELATIVE CLAUSE.
            THIS STATE ALSO GETS THE STRING IN PHASE WITH PATHS
            THAT WENT DIRECTLY FROM R/ TO R/WH.)
```

```
                )
        (SETR WH NIL)
        (TO R/WH))
    (PUSH NP/ (WRD WHOSE)
        (SENDR ADJS (BUILDQ ((POSS +))
                            WH))
        (ADDL VMODS (BUILDQ (PP (PREP +)
                                *)
                            PREP))
        (SETR WH NIL)
        (TO R/WH)))


(R/WH
    (PUSH PP/ (CAT PREP)
        (ADDL VMODS *)
        (TO R/WH))
    (PUSH NP/ (NOR (CAT V)
                   (GETR RELVPFLG))
        (COND
          ((GETR WH)
            (HOLD (GETR WH))))
        (SETR SUBJ *)
        (TO S/NP))
    (WRD THERE T
        (SETR THERE T)
        (SETR SUBJ (GETR WH))
        (TO S/NP))
    (JUMP S/NP (AND (GETR WH)
                    (CAT V))
        (SETR SUBJ (GETR WH))))


(S/
    (JUMP S/Q (QSTART)
```

HI I HE INITIAL A E F HE H LE G AMMA
BASICALLY, THIS STATE TRIES TO DECIDE WHAT TYPE OF SENTENCE WE HAVE: QUESTION, INTERROGATIVE, OR IMPERATIVE. THE ARCS SET THE TYPE REGISTER AND TRANSFER TO STATES DESIGNED TO HANDLE THE DIFFERENT CONSTRUCTIONS. CERTAIN VERB MODIFIERS MAY OPTIONALLY PRECEDE THE MAIN BODY OF THE SENTENCE AND PARTITIVE CONSTRUCTIONS MAY HAVE BEEN FRONTED FROM A NOUN PHRASE; THESE CONSTITUENTS ARE ANALYZED BY LOOPING THROUGH S/.)

(* QSTART IS TRUE FOR THE SMALL SET OF WORDS THAT CAN START QUESTIONS.)

```
        (SETRQ TYPE Q))
    (WRD PLEASE (NULL STACK)
                                                (* AT THE TOP LEVEL,
                                                'PLEASE' USUALLY
                                                SIGNIFIES THE BEGINNING
                                                OF AN IMPERATIVE.)
        (ADDL VMODS (BUILDQ (ADV PLEASE)))
        (TO S/IMP))
    (JUMP S/IMP
            (* AN IMPERATIVE CAN OCCUR ONLY AT THE TOP LEVEL;
            IT USUALLY BEGINS WITH AN UNTENSED VERB: 'GIVE ME
            ...')

        (AND (CHECKF V UNTENSED)
             (NULL STACK)))
    (JUMP S/DCL (NOR (QSTART)
                     (CAT PREP)
                     (NULL STRING))

            (* THE BEST TEST FOR THE BEGINNING OF A DECLARATIVE
            IS THAT IT NOT BE THE BEGINNING OF A QUESTION.)

        (SETRQ TYPE DCL))
    (CAT ADV (RFEAT NEGADV)

            (* A NEGATIVE ADVERB ('HARDLY', 'BARELY') USUALLY
            INVOLVES SUBJECT-VERB INVERSION WHEN IT OCCURS AT
            THE BEGINNING OF A SENTENCE.
            THE TYPE IS STILL 'DCL', BUT WE GO TO STATE S/NP TO
            PICK UP THE VERB FOLLOWING THE ADVERB
            ('BARELY HAD HE LEFT ...'), SUBSEQUENT ANALYSIS
            RESEMBLES THE PROCESSING OF YES-NO QUESTIONS.)

        (ADDL VMODS (BUILDQ (ADV *)))
        (SETRQ TYPE DCL)
        (TO S/NP))
    (CAT ADV T
        (ADDL VMODS (BUILDQ (ADV *)))
        (TO S/))
    (PUSH PP/ (WRD OF)

            (* A PARTITIVE EXPRESSION MAY HAVE BEEN FRONTED
            ('OF THE MEN HOW MANY ...'). ANALYZE IT HERE, BUT
            HOLD IT (WITH THE FEATURE 'PARTITIVE') TO BE PICKED
            UP AT STATE NP/QUANT IN THE FIRST NP.)
```

```
        (HOLD * (QUOTE ((PARTITIVE))))
        (TO S/))
    (PUSH PP/ (CAT PREP)

            (* IF THERE IS A QWORD OR QDET IN THE OBJECT OF THE
            PP, IT BECOMES THE DS SUBJECT IN S/QP1)

        (HOLD * (QUOTE ((FRONTED))))
        (TO S/QP1)))

(S/;
    (CAT CONJ (OR (NULLR CONJ

            (* A ';' WAS FOUND AT THE END OF THE TOP-LEVEL S,
            INDICATING A SEQUENCE OF SEMICOLON CONJOINED
            SENTENCES, WITH REAL CONJUNCTIONS POSSIBLY FOLLOWING
            THE SEMICOLONS. WE PICK UP THE CONJS IF PRESENT, AND
            PUSH FOR THE FOLLOWING S. WE KEEP THE SEQUENCE OF
            S'S IN SBODY. NOTE: THE STRATEGY HERE IS ALSO USED
            FOR SEMICOLON CONJOINED NPU'S.)

)
                (EQ * (GETR CONJ)))

            (* IF WE FIND A CONJ, EITHER IT MUST BE THE FIRST
            ONE ENCOUNTERED CONJOINING THE S'S, OR IT MUST BE
            THE SAME AS PREVIOUSLY ENCOUNTERED ONES.
            THUS WE ACCEPT 'S; AND S; AND S' AND 'S;
            S; AND S', BUT NOT 'S; OR S;
            AND S'.)

        (SETR CONJ *)
        (TO S/;))
    (PUSH S/ T
        (ADDL SBODY *)
        (TO S/;S)))
```

```
(S/;S
   (WRD ; T
        (TO S/;

           (* HERE AFTER PROCESSING ONE S IN A SERIES OF
           SEMICOLON CONJOINED S'S. IF THE CURRENT WORD IS ';',
           THEN ANOTHER S OR CONJ FOLLOWS--GO TO STATE S/;.
           OTHERWISE, POP A COORDINATE LIST OF THE IDENTIFIED
           S'S. NOTE: IF THE END OF THE SERIES IS REACHED WITH
           NO CONJ, THE DEFAULT CONJ IS 'OR'.)

))
   (POP (BUILDQ (@ (S #)
                   #)
                (COND
                  ((GETR CONJ))
                  (T (QUOTE OR)))
                (REVERSE (GETR SBODY)))
        T))


(S/AUX
   (CAT NEG (NULLR NEG)

           (* HERE AFTER FINDING THE FIRST VERB, WHICH MIGHT
           HAVE BEEN THE MAIN VERB OR AN AUXILIARY.
           LOOP FOR AN OPTIONAL NEG ('NOT') AND UNDO
           'DO-SUPPORT' IF NECESSARY. THEN IF WE ALREADY HAVE
           THE SUBJECT, GO TO VP/V IF IT AGREES WITH THE VERB;
           IF WE HAVEN'T IDENTIFIED THE SUBJECT
           (BECAUSE OF SUBJECT-VERB INVERSION) GO TO S/NO-SUBJ
           TO FIND ONE.)


        (COND
          ((WRD DO MODAL)
           (SETR MODAL NIL)))
        (SETRQ NEG NEG)
        (TO S/AUX))
(JUMP VP/V (OR (AND (GETR SUBJ)
                    (PNCHECK (GETR SUBJ)
                             (GETR PNCODE)))
               (GETR THERE)))
(JUMP S/NO-SUBJ (NOR (GETR SUBJ)
                     (GETR THERE))))
```

```
(S/DCL
   (WRD THERE T
        (SETR THERE T)

           (* WE THINK THIS IS A DECLARATIVE SENTENCE.
           IT MUST BEGIN WITH A SUBJECT NP, A SUBJECT
           COMPLEMENT ('TO HAVE THE INFORMATION IS IMPORTANT'),
           OR 'THERE' IF THERE-INSERTION HAS OCCURRED
           (WE MUST LATER FIND 'BE' OR 'EXIST'))

        (TO S/NP))
   (PUSH NP/ T
        (SETR SUBJ *)
        (TO S/NP))
   (PUSH COMPL/ (OR (WRD (FOR TO THAT))
                    (AND (WRD TO (NEXTWRD))
                         (CAT NEG)))

           (* THERE ARE 4 TYPES OF SUBJECT COMPLEMENTS: 'FOR ME
           TO GO...', 'THAT I WENT ...', 'TO GO ...', 'NOT TO
           GO...'. THE PUSH IS PERMITTED ONLY IF WE HAVE AN
           APPROPRIATE COMPLEMENTIZER.)

        (SETR SUBJ *)
        (TO S/NP)))

(S/HOW
   (CAT ADJ T
        (HOLD (BUILDQ (ADJ *)))
        (HOLD (BUILDQ (PP (PREP TO)
                          (NP (DET WHQ)
                              (N DEGREE)
                              (NU SG)))))
        (TO S/NP

           (* RECOGNIZES "HOW<ADJ>IS..."
           AND "HOW <ADV><AUX>...". THE ADJECTIVE, ADVERB AND
           THE WORD "HOW" ARE ALL HELD FOR LATER.)

))
   (CAT ADV T
        (HOLD (BUILDQ (ADV *)))
        (HOLD (BUILDQ (PP (PREP TO)
                          (NP (DET WHQ)
                              (N DEGREE)
                              (NU SG)))))
        (TO S/NP)))
```

```
(S/IMP
   (CAT V (GETF UNTENSED

            (* WE RECOGNIZE AN IMPERATIVE SENTENCE, AND SET UP
            REGISTERS ACCORDINGLY. THE SUBJECT IS 'YOU' AND THE
            TENSE IS 'PRESENT'. WE SET THE V AND GO TO VP/HEAD
            TO PICK UP POST-VERB CONSTITUENTS.)

)
        (SETRQ TYPE IMP)
        (SETR SUBJ (BUILDQ (NP (PRO YOU))))
        (SETR V *)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (TO VP/HEAD)))

(S/NO-SUBJ
   (WRD THERE (OR (NULLR WHQ

            (* THERE WAS NO IDENTIFIABLE SUBJECT BEFORE THE
            FIRST VERB. THE SUBJECT MIGHT BE HERE IN THE STRING
            IF S-V INVERSION OCCURRED, OR IT MIGHT BE IN THE WHQ
            REGISTER.)

)
                  (PNCHECK (GETR WHQ)
                           (GETR PNCODE)))

            (* ANYTHING IN WHQ MUST AGREE WITH THE VERB AND
            BECOMES THE SUBJECT ('HOW MANY MEN WERE THERE...').
            IF WHQ IS EMPTY ('WERE THERE MANY MEN...') WE MOVE
            ON TO S/THERE.)
```

```
      (COND
        ((GETR WHQ)
          (SETR SUBJ (GETR WHQ)))
        ((GETR WH)
          (SETR SUBJ (GETR WH))))
      (SETR THERE T)
      (TO S/THERE))
   (JUMP VP/V (AND (GETR WH)
                   (WRD HAVE V
```

(* WE CHOOSE TO MAKE WH THE SUBJECT IF THE VERB IS "HAVE", RATHER THAN LOOKING FOR ANOTHER NP ON THE FOLLOWING PUSH NP/ ARC. IF HAVE TURNS OUT TO BE AN AUXILIARY FOLLOWED BY THE REAL SUBJECT , THIS ARC WILL FAIL (E.G. IF THE SENTENCE WERE "HOWMANY PEARS HAVE THE BOYS EATEN?"))

```
)
                   (PNCHECK (GETR WH)
                            (GETR PNCODE)))
      (SETR SUBJ (GETR WH)))
   (PUSH NP/ T
```

(* WE LOOK FOR AN NP IN THIS POSITION. IF NPFEATURES WAS SET (IN THE PUSH FROM STATE S/QDET) WE PRESERVE THE OLD VALUE BECAUSE THE REGISTER WILL BE RESET BY THIS PUSH (AT STATE NP/HEAD). IF THIS PUSH IS SUCCESSFUL, THE RESULTING NP MUST AGREE WITH THE VERB AND BECOMES THE SUBJECT. OUR INDECISION ABOUT THE WHQ IS RESOLVED--IT CANNOT BE THE SUBJECT SO IT IS HELD TO BE PICKED UP AS AN OBJECT OR PREP OBJECT. WE ALSO HOLD THE NPFEATURES ASSOCIATED WITH IT, FOR LATER RESUMPTION. FINALLY, THE DO-SUPPORT NECESSARY FOR S-V INVERSION IS UNDONE.)

```
        (! (SETR HOLDNPFEATURES (GETR NPFEATURES)))
        (SENDR ANAPHORFLG (GETR ANAPHORFLG))
        (COND
          ((NULL (PNCHECK * (GETR PNCODE)))
            (ABORT)))
        (SETR SUBJ *)
        (COND
          ((GETR WHQ)
            (HOLD (GETR WHQ)
                  (GETR HOLDNPFEATURES))))
        (COND
          ((GETR WH)
            (HOLD (GETR WH)
                  (GETR HOLDNPFEATURES))))
        (COND
          ((WRD DO MODAL)
            (SETR MODAL NIL)))
            (TO VP/V))
      (JUMP VP/V (AND (GETR WH)
                      (NOT (WRD HAVE V

                (* IF "THERE" AND NP DIDN'T WORK, BUT "WH" AGREES
                WITH THE VERB, MAKE THAT THE SUBJECT)

    ))
                      (PNCHECK (GETR WH)
                               (GETR PNCODE)))
            (SETR SUBJ (GETR WH)))
      (JUMP VP/V

                (* IF 'THERE' AND NP DIDN'T WORK BUT WE HAVE A WHQ
                WHICH AGREES WITH THE VERB, WE TRY THAT AS THE
                SUBJECT.)


            (AND (GETR WHQ)
                 (PNCHECK (GETR WHQ)
                          (GETR PNCODE)))
            (SETR SUBJ (GETR WHQ))))
```

```
(S/NP
   (POP (GETR POPVAL)
        (AND (NOR STRING HOLD STACK (GETR VMODS))
             (SETR POPVAL (COND
                    ((WRD Q TYPE)
                      (BUILDQ (S NPQ +)
                              WHQ))
                    (T (BUILDQ (S NPU +)
                               SUBJ))))
             (OR (NOT SIFLAG)
                 (INTERP (GETR POPVAL))))

          (* IF WE REACHED THE END OF THE STRING AT THE TOP
          LEVEL, WE BUILD A NOUN-PHRASE UTTERANCE AS THE PARSE
          OF THE SENTENCE ('INFORMATION ON ...' OR 'WHICH
          MAN').)

          (* HERE AFTER OUR FIRST ATTEMP AT FINDING A NP,
          EITHER AS A DECLARATIVE SUBJECT OR AS A QUESTION
          WORD (WHQ).)

(CAT ADV T                                    (* WE ATTEMPT
                                              SIMULTANEOUS
                                              INTERPRETATION IF SIFLAG
                                              IS T.))
     (ADDL VMODS (BUILDQ (ADV *))             (* AN ADVERB MAY PRECEDE
                                              AN AUXILIARY VERB)
                 )
     (TO S/NP))
(CAT V (GETF TNS)
          (* USUALLY WE FIND A TENSED VERB AT THIS STRING
          POSITION, EITHER AS THE FIRST WORD IN A QUESTION
          (IF WE JUMPED FROM S/Q), OR FOLLOWING AN NP OR
          'THERE'. IF IT IS A MODAL ('WOULD', 'COULD', ETC.)
          WE PUT IT IN THE MODAL REGISTER, OTHERWISE IN V.
          IF WE ARE IN A WH-QUESTION AND THE VERB WAS NOT AN
          AUXILIARY ('WHO HIT JOHN') THEN THE WHQ IS THE
          SUBJECT. IN ANY CASE, SAVE THE TENSE AND
          PERSON-NUMBER CODE.)
```

```
        (COND
          ((MODAL)
            (SETR MODAL *))
          (T (SETR V *)))
        (COND
          ((AND (GETR WHQ)
                (NOR (MODAL)
                     (WRD (HAVE BE))))
            (SETR SUBJ (GETR WHQ))
            (SETR WHQ NIL)))
        (COND
          ((AND (GETR WH)
                (NOR (MODAL)
                     (WRD (HAVE BE DO))))
            (SETR SUBJ (GETR WH)
                  (* IF WE ARE IN A QREL CLAUSE AND THE VERB WAS NOT
                  AN AUXILIARY, THE THE WH PASSED DOWN FROM THE MATRIX
                  SENTENCE IS THE SUBJECT.)

                  )
            (SETR PNCODE (GETF PNCODE))
            (TO S/AUX)))
        (SETR TNS (GETF TNS))
        (SETR PNCODE (GETF PNCODE))
        (TO S/AUX))
    (WRD ; (NULL STACK)

            (* AT THE TOP LEVEL, A SEMICOLON HERE INTRODUCES A
            SEQUENCE OF NP'S TO BE PARSED AS CONJOINED NPU'S.
            THE GRAMMAR FOR THIS BEGINS AT NPU/;

)

        (ADDL NPU (GETR SUBJ))
        (TO NPU/;))
    (JUMP S/DCL (WRD IDQ TYPE)

            (* WE ARE IN AN INDIRECT QUESTION, PUSHED TO FROM
            VP/HEAD ('I KNOW WHO YOU ARE') THE WHQ REGISTER IS
            SET, BUT INVERSION HASN'T OCCURRED, I.E., THE
            BEGINNING OF THE CLAUSE AFTER THE Q-WORD LOOKS LIKE
            A DECLARATIVE. ERGO, JUMP TO S/DCL.)

        (HOLD (GETR WHQ)
              (GETR NPFEATURES))
        (SETR WHQ NIL))
    (JUMP S/AUX (GETR V)))
```

```
(S/Q
   (WRD HOW T
        (TO S/HOW)

            (* THE SENTENCE BEGINS LIKE A QUESTION.
            FOUR POSSIBILITIES: (1) A QWORD--A WH WORD THAT
            FUNCTIONS AS A PRONOUN ('WHO','WHAT');
            (2) AN AUXILIARY VERB; (3) A Q-DETERMINER
            ('WHICH MAN'); (4) THE ADVERB "HOW")

)
   (CAT QWORD T

            (* THE DICTIONARY ENTRIES FOR QWORDS ARE COMPLETE NP
            OR ADVERBIAL STRUCTURES. WE COPY THEM SO WE DON'T
            DESTROY THE DICTIONARY ENTRIES BY FUTURE OPERATIONS.
            THE FEATURE 'SUBJ/OBJ' INDICATES THAT THE QWORD CAN
            REPRESENT EITHER THE SUBJECT OR THE OBJECT, SO WE
            STORE IT IN WHQ UNTIL FURTHER INFORMATION ENABLES US
            TO DECIDE. IF THE QWORD LACKS THIS FEATURE, THEN IT
            CANNOT BE THE SUBJECT ('WHOM'); WE HOLD IT FOR
            POST-VERBAL PROCESSING.)


(COND
  ((GETF SUBJ/OBJ)
    (SETR WHQ (COPY *)))
  (T (HOLD (COPY *))))
(COND
  ((GETF ANAPHORIC)
    (SETRQ ANAPHORFLG T

    (* THE QWORD "WHICH" IS ANAPHORIC: IT IMPLIES CHOICE
    FROM A PREVIOUSLY MENTIONED SET.
    THE QWORD "WHAT" IS NOT, ANAPHORFLG SIGNALS THIS
    DISTINCTION AND WILL BE USED IF THE SENTENCE
            CONTAINS A SUPERLATIVE ADJECTIVE TO DETERMINE THE
            SET THAT IT NEEDS.)

)))
        (TO S/NP))
   (JUMP S/NP (CAT V))
   (CAT QDET T

            (* CURRENTLY, THE NP/ LEVEL DOES NOT PROCESS
            QUESTION DETERMINERS. WE PICK THEM UP HERE AND PUSH
            INTO THE MIDDLE OF THE NP/ NETWORK FROM S/QDET,
            SENDING THE QDET DOWN.)


        (SETR DET *)
        (TO S/QDET)))
```

```
(S/QDET
   (PUSH NP/ART T
        (! (COND
             ((NEQ (CAR (GETR DET)

             (* DET CONTAINS THE Q-DETERMINER FOUND AT STATE S/Q.
             WE PUSH INTO THE MIDDLE OF THE NP/ NETWORK TO FIND
             THE WHQ NOUN-PHRASE. WE INITIALIZE THE REGISTER QDET
             TO PREVENT REDUCED RELATIVE CLAUSES WITHIN THE NP:
             'THE MAN I SAW...' AND 'HOW MANY MEN WHO I SAW...'
             ARE ALLOWED, BUT 'HOW MANY MEN I SAW...' IS OUT.)

                          )
                (QUOTE POSTART))
           (SENDRQ RELVPFLG T

             (* WE DO NOT RELATIVIZE THE REMAINDER OF THE
             SENTENCE IF THE QDET WAS "HOW MUCH")

))))
        (SENDR DET (GETR DET))
        (SENDRQ QDET T)
        (SETR WHQ *)
        (TO S/NP)))


(S/QP1
   (VIR PP (GETR NP)
        (SETR WHP (RELATIVIZE (COPY *)

             (* WE MAKE A QUESTIONED NP OR QWORD IN A FRONTED PP
             THE DEEP STRUCTURE SUBJECT)

                                  ))
        (SETRQ TYPE Q)
        (TO S/QP2))
   (JUMP S/ (NOT (GETR NP))))


(S/QP2
   (PUSH S/NP T
        (SENDR VMODS (LIST (GETR WHP)))

        (* WH-QUESTIONS ARE PARSED AS NOUN PHRASE UTTERANCES
        WITH THE QUESTION-NP AS SUBJECT AND THE REST OF THE
        QUESTION IN A RELATIVE CLAUSE ON THE SUBJECT.
        THE TYPE OF THE RELATIVE CLAUSE IS QREL TO
        DISTINGUISH IT FROM A SURFACE STRUCTURE RELATIVE.
        A WHQ REGISTER IS USED INSTEAD OF A SUBJ REGISTER TO
        HOLD THE SUBJECT.)
```

```
        (SENDRQ TYPE QREL)
        (SETR WHQ (APPEND (GETR NP)
                          (LIST *)))
        (TO S/VP)))

(S/S
   (POP (GETR POPVAL)
        T
                                        (* POPVAL MAY HAVE BEEN
                                        INTERPRETED IN S/VP)))

(S/SADV
   (PUSH NP/ T
        (ADDL VMODS (BUILDQ (ADVP (ADV +)
                                  *)
                            SADV))
        (TO S/VP)))

(S/THERE
   (TEST DO T
        (SETQ FEATURES (GETR NPFEATURES)

           (* THIS STATE IS PLACED BETWEEN S/NO-SUBJ AND VP/V
           TO ALLOW FOR EXTRAPOSED RELATIVE CLAUSES WITH THERE
           INSERTION IN QUESTIONS: 'HOW MANY MEN WERE THERE WHO
           BOUGHT ...'. THE RESUME ACTION WILL MOVE THE
           RELATIVE TO ITS PROPER LOCATION IN THE NP, WHICH
           THEN BECOMES THE SUBJECT. NOTICE THAT UNLESS
           NPFEATURES HAS BEEN SET, THIS ARC IS ESSENTIALLY A
           NO-OP. THUS, SINCE QWORDS DON'T SET NPFEATURES
           (QDETS DO), 'WHO WAS THERE WHO DID ...' IS
           (PERHAPS ERRONEOUSLY) NOT ALLOWED.)


        )
   (RESUME)
   (COND
     ((GETR NPFEATURES)
       (SETR SUBJ *)))
   (JUMP VP/V)))
```

```
(S/VP
   (WRD . T
        (TO S/VP

            (* HERE WHEN THE VERB-PHRASE OF THIS S HAS BEEN
            NEARLY COMPLETED. THERE MIGHT BE SOME ADVERBS OR
            PP'S STILL ON THE HOLD LIST, WHICH WE PICK UP HERE.
            ALSO, AT THE TOP LEVEL, THERE MIGHT BE SOME TERMINAL
            PUNCTUATION, OR A SEMICOLON, INDICATING THAT THIS IS
            THE FIRST ITEM IN A SERIES OF CONJOINED S'S.
            THE USUAL CASE, HOWEVER, IS TO POP THE ANALYZED S
            STRUCTURE.)

))
   (CAT ADV (RFEAT TRANSADV)
        (SETR SADV *)
        (TO S/SADV))
(VIR PP T
     (ADDL VMODS *)
     (TO S/VP))
(VIR ADJ (RFEAT COPULA V)
     (SETR V (BUILDQ (@ (ADJ)                    (*)
                        #)
                     (COND
                       ((WRD (APPEAR SEEM)
                             V)
                        (QUOTE (SEEMING)))))))

        (* THE ADJECTIVE PICKED UP IN S/HOW REPLACES THE
        COPULA AS DS VERB. IF THE COPULA WAS "APPEAR" OR
        "SEEM", THE FEATURE "SEEMING" IS ADDED TO THE NEW
        VERB.)


          )
     (TO S/VP))
(VIR ADV T
     (ADDL VMODS *)
     (TO S/VP))
(MEM (%. ? !)
     (NULL STACK)

        (* NOTE THAT A TERMINATING QUESTION MARK OVERRIDES
        THE SYNTACTIC TYPE OF THE SENTENCE: 'I NEED SOME
        INFORMATION?' IS A QUESTION, NOT A DECLARATIVE.)
```

```
      (COND
        ((AND (WRD ?)
               (NOT (WRD Q TYPE)))
          (SETRQ TYPE Q)))
      (TO S/VP))
   (WRD ; (NULL STACK)
      (ADDL SBODY (SBUILD))
      (TO S/;))
   (JUMP S/S T
      (SETR POPVAL (SBUILD)                    (* THIS ARC ALLOWS FOR
                                                SIMULTANEOUS
                                                INTERPRETATION IF SIFLAG
                                                IS T.)
            )
      (COND
        ((OR (NOT SIFLAG)
             (WRD REL TYPE))
          T)
        ((AND (OR STACK (NOR STRING HOLD))
              (NOT (INTERP (GETR POPVAL))))
          (SUSPEND 2)))))


    ADJ
   (WRD THAN (COMPARATIVE V                    (* VP/ADJ PROCESSES THE
                                                COMPLEMENTS OF PREDICATE
                                                ADJECTIVES, E.G. THE
                                                INFINITIVE ON
                                                "JOHN IS EASY TO PLEASE"
))
      (TO VP/ADJ-COMP))
   (PUSH COMPL/ (AND (COMPARATIVE V)
                     (WRD THAN))
      (SETR COMPL *)
      (TO S/VP))
   (PUSH COMPL/ (AND (WRD (FOR TO THAT))
                     (EQUAL (GETR SUBJ)
                            (QUOTE (NP (PRO IT)
                                       (NU SG)))))
      (SETR SUBJ *)
      (TO S/VP))
   (PUSH COMPL/ (AND (WRD FOR)
                     (RFEAT FORCOMP (CADR (GETR V))))
      (SETR COMPL *)
      (TO S/VP))
   (PUSH FOR/NP (AND (RFEAT TOCOMP (CADR (GETR V)))
                     (WRD TO))
      (! (COND
           ((RFEAT SUBJLOW (CADR (GETR V)))
             (SENDR SUBJ (GETR SUBJ)))
           (T (SENDR SUBJ (QUOTE (NP (PRO SOMETHING)
                                     (NU SG))))
              (SENDR OBJ (GETR SUBJ)))))
      (SETR COMPL *)
      (TO S/VP))
   (JUMP VP/VP T))
```

```
(VP/ADJ-COMP
   (PUSH NP/ T
        (SENDR V (GETR V))
        (SENDR ANAPHORFLG (GETR ANAPHORFLG))
        (SETR OBJ *

           (* WE LOOK FOR AN NP FOLLOWING "THAN" IN A
           COMPARATIVE COMPLEMENT. IF WE FIND ONE
           (RATHER THAN A SENTENCE), WE MAKE IT THE OBJECT OF
           THE VERB. E.G. "FRED IS TALLER THAN JIM."
           IS ANALYZED AS "FRED [TALL COMPARATIVE] JIM.")

)
        (TO VP/NP)))

(VP/AGT
   (PUSH NP/ T
        (SETR SUBJ *)

           (* HERE IF THE SENTENCE IS PASSIVE, WE HAVE NOT YET
           FOUND THE AGENT, BUT WE HAVE SEEN THE PREPOSITION
           BY, WHICH MIGHT INTRODUCE THE AGENT NP.
           USALLY, THE AGENT NP WOULD BE IDENTIFIED HERE 'BY' A
           PUSH, BUT IN A QUESTION OR RELATIVE CLAUSE, THE
           AGENT MIGHT HAVE BEEN FRONTED, LEAVING THE 'BY'
           DANGLING: 'WHO IS THE INFORMATION NEEDED BY' OR 'THE
           MAN THE INFORMATION IS NEEDED BY...'.
           IN THESE CASES, THE NP HAS BEEN HELD, AND ARC 2
           PICKS IT UP. THE RESUME ACTION ALLOWS FOR AN
           EXTRAPOSED RELATIVE CLAUSE OR PP.)

        (SETR AGFLAG NIL)
        (TO VP/VP))
   (VIR NP T
        (RESUME)
        (SETR SUBJ *)
        (SETR AGFLAG NIL)
        (TO VP/VP)))
```

```
(VP/COMP-ADJ
   (CAT ADJ T
        (SETR V (BUILDQ (@ (ADJ)                    (*)
                           # #)
                        (COND
                          ((WRD MORE MORE-MOST)
                            (QUOTE (COMPARATIVE)))
                          (T (QUOTE (SUPERLATIVE))))
                        (COND
                          ((WRD (APPEAR SEEM)
                                V)
                            (QUOTE (SEEMING)))))

            (* MAKES AN UNDETERMINED COMPARATIVE OR SUPERLATIVE
            ADJECTIVE IN PREDICATE ADJECTIVE POSITION THE DS
            VERB. IF THE COPULA WAS "APPEAR" OR "SEEM", THE
            FEATURE "SEEMING" IS ADDED TO THE NEW VERB.
            E.G. "FRED IS MOST INTERESTED IN SNAKES" THE DS VERB
            IS (ADJ INTERESTED SUPERLATIVE))


                )
        (TO VP/ADJ)))


(VP/HEAD
   (CAT PREP (SETQ TEMP (VPARTICLE V

            (* HERE WHEN WE HAVE MADE FIRM DECISIONS ABOUT THE
            MAIN VERB AND THE SUBECT. WE LOOK FOR POST-VERBAL
            MODIFIERS (OBJECTS, SENTENTIAL COMPLEMENTS,
            PARTICLES, PREDICATE ADJECTIVES), MANY OF WHICH ARE
            SPECIFIED BY ROOT FEATURES ON THE VERB.)

))


(* THIS ARC IDENTIFIES A PARTICLE IMMEDIATELY
FOLLOWING THE VERB ('LOOK UP', 'LOOK FOR'). THE
FUNCTION VPARTICLE EXAMINES THE PROPERTY 'PARTICLES'
IN THE VERB'S DICTIONARY ENTRY, WHICH INDICATES WHAT
PARTICLES ARE ALLOWED, AND WHAT THE NEW ROOT VERB
CORRESPONDING TO THE VERB+PARTICLE COMBINATION IS.
THUS, 'LOOK UP' MIGHT CAUSE THE MAIN VERB TO BE
CHANGED TO 'LOOK-UP', WHICH MIGHT HAVE DIFFERENT
ROOT-FEATURES THAN 'LOOK'.)
```

```
        (SETR V TEMP)
        (SETR HEAD TEMP)
        (TO VP/V))
        H N     AND     D BE
                      STRING)
        (SENDRQ V BE)
        (SENDRQ ANAPHORFLG (GETR ANAPHORFLG)      (* WE MAKE THE NP
                                                  FOLLOWING BE ITS OBJECT
                                                  RIGHT OFF.)
                )
        (SETR OBJ *)
        (TO VP/NP))
    (JUMP VP/NP (OR (RFEAT INTRANS V)
                    (AND (VTRANS V)
                         (GETR OBJ)))

                (* IF THE MAIN VERB IS MARKED INTRANSITIVE, THERE IS
                NO PREDICATE COMPLEMENT DIRECTLY TIED TO THE
                VERB--WE SKIP TO VP/NP.)

    )
        (PUSH S/Q (AND (VTRANS V)
                       (WRD (WHICH WHO WHAT WHOSE)))


                (* CERTAIN VERBS CAN TAKE INDIRECT QUESTIONS AS
                THEIR OBJECTS (E.G. 'KNOW'). THESE ARE MARKED AS
                ORDINARY TRANSITIVES IN THE DICTIONARY, SO IN ORDER
                TO RECOGNIZE THESE CONSTRUCTIONS, WE ALLOW THE
                POSSIBILITY THAT ALL TRANSITIVE VERBS CAN TAKE THESE
                OBJECTS. ('I KNOW WHO WANTED THE INFORMATION.'))


            (SENDRQ TYPE IDQ)
            (SETR OBJ (BUILDQ (NP *)))
            (TO VP/NP))
        (VIR NP (VTRANS V)

(* FOR RELATIVE CLAUSES, PASSIVES, AND
WHO-BE-QUESTIONS, THE DIRECT OBJECT HAS BEEN HELD;
WE PICK UP HERE, LOOKING FOR POSSIBLE EXTRAPOSED
RELATIVES.)
```

```
      (RESUME V)
      (SETR OBJ *)
      (TO VP/NP))
(PUSH NP/ (AND (VTRANS V)
               (NOT (WRD BE V)))

        (* HERE WE PICK UP THE REGULAR OBJECT OF TRANSITIVE
        VERBS. NOTE THAT FOR A WHQ-QUESTION WITH 'BE' AS THE
        MAIN VERB ('WHO IS THE LEADER?'), THE SUBJECT
        ('THE LEADER') WAS PICKED UP AT STATE S=NO-SUBJ, AT
        WHICH POINT THE WHQ WAS HELD.
        THUS WE DON'T LOOK FOR THE OBJECT ON THIS ARC, BUT
        RATHER ON THE SUBSEQUENT VIR ARC.)

      (SENDR V (GETR V))
      (SENDR ANAPHORFLG (GETR ANAPHORFLG))
      (SETR OBJ *)
      (TO VP/NP))
(WRD MORE (AND (RFEAT COPULA V                    (* WE RECOGNIZE TWO WORD
                                                  INFLECTED ADJECTIVES IN
                                                  PREDICATE ADJECTIVE
                                                  POSITION))
               (GETP (NEXTWRD)
                     (QUOTE ADJ)))
      (TO VP/MORE))
(PUSH COMPL/ (AND (WRD THAT)
                  (RFEAT THATCOMP V))

        (* VERBS MARKED 'THATCOMP' CAN TAKE A THAT-CLAUSE AS
        A COMPLEMENT ('I BELIEVE THAT THEY...').)

      (SETR COMPL *)
      (TO VP/NP))
(PUSH COMPL/ (AND (WRD (FOR TO))
                  (RFEAT FORCOMP V))

(* 'FORCOMP' VERBS TAKE A FOR- OR TO-COMPLEMENT: 'WE
HOPE FOR JOHN TO COME', 'WE WANT TO COME'.
FOR A TO-COMPLEMENT, THE SUBJECT OF THE COMPLEMENT
IS THE SAME AS THE SUBJECT OF THE SENTENCE.)
```

```
        (! (COND
             ((WRD TO)
               (SENDR SUBJ (GETR SUBJ)))))
        (SETR COMPL *)
        (TO VP/VP))
    (PUSH COMPL/NTYPE (SCOMP V)

        (* FINALLY, CERTAIN VERBS ALLOW THE 'THAT' PRECEDING
        A COMPLEMENT TO BE DELETED,
        THE PUSH HERE ALLOWS FOR THIS.)

        (SENDRQ NTYPE THAT)
        (SETR COMPL *)
        (TO VP/NP)))

(VP/MORE
   (CAT ADJ T
        (SETR V (BUILDQ (@ (ADJ)                    (*)
                           (COMPARATIVE)
                           #)
                        (COND
                          ((WRD (APPEAR SEEM)
                                V)
                           (QUOTE (SEEMING))))

            (* A TWO WORD COMPARATIVE ADJECTIVE REPLACES A
            COPULA AS DS VERB. IF THE COPULA WERE "APPEAR" OR
            "SEEM", THE NEW VERB GETS THE FEATURE "SEEMING".)


                        ))
        (TO VP/ADJ)))
```

```
(VP/NP
   (PUSH COMPL/ (OR (AND (WRD (FOR THAT))
                         (EQUAL (GETR SUBJ)
                                (QUOTE (NP (PRO IT)))
                                (* THE FIRST OBJECT OR
                                COMPLEMENT CAN SOMETIMES
                                BE FOLLOWED BY OTHERS.)
                         ))
                    (AND (WRD THAT)
                         (GETR AGFLAG)))
        (COND
          ((GETR AGFLAG)
            (SETRQ AGFLAG NIL)))
        (SETR SUBJ *)
        (TO VP/VP
              (* IF THE SUBJECT WAS 'IT', THE SUBJECT COMPLEMENT
              OF THE VERB MIGHT HAVE BEEN EXTRAPOSED TO THIS
              POSITION: 'IT IS CLEAR THAT ...' OR 'IT IS EASY FOR
              JOHN TO ...'. THIS ARC MOVES THESE COMPLEMENTS BACK
              TO SUBJECT POSITION, WHERE THEY BELONG.
              ALSO, A THAT-COMPLEMENT SUBJECT COULD HAVE BEEN
              MOVED TO THIS POSITION BY PASSIVIZATION: 'I WAS
              SURPRISED THAT...' FROM 'THAT ...
              SURPRISED ME'. THE OBJECT 'I-ME' IS PICKED UP ON THE
              VIR NP ARC FROM STATE VP/HEAD;
              THE AGENT CLAUSE IS PICKED UP HERE.)

))
   (PUSH FOR/NP (AND (RFEAT TOCOMP V)
                     (GETR OBJ))

              (* A TO-COMPLEMENT CAN OCCUR AFTER THE OBJECT: 'I
              PROMISED JOHN TO GO' OR 'I WANTED JOHN TO GO'.
              FOR MOST VERBS, THE SUBJECT OF THE COMPLEMENT IS THE
              OBJECT ('JOHN') OF THE MAIN SENTENCE, BUT VERBS
              MARKED 'SUBJLOW' HAVE THE SUBJECT OF THE MAIN
              SENTENCE PASSED DOWN (E.G. 'PROMISE'). FOR
              'TRANSCOMP' VERBS, THE OBJECT PASSED DOWN TO BE
              SUBJECT REMAINS AS THE TOP-LEVEL OBJECT
              ('PERSUADE'), BUT THIS IS NOT ALWAYS TRUE
              ('EXPECT'))
```

```
        (SENDR SUBJ (COND
                      ((RFEAT SUBJLOW V)
                        (GETR SUBJ))
                      (T (GETR OBJ))))
        (COND
          ((NOT (RFEAT TRANSCOMP V))
            (SETR OBJ NIL)))
        (SETR COMPL *)
        (TO VP/VP))
    (CAT PREP (SETQ TEMP (VPARTICLE V))
                                                  (* A PARTICLE CAN OCCUR
                                                  AFTER THE OBJECT: 'LOOK
                                                  THE INFORMATION UP')
        (SETR V TEMP)
        (SETR HEAD TEMP)
        (TO VP/VP))
    (CAT ADV T
        (ADDL VMODS (BUILDQ (ADV *)                (* AN ADVERB MAY FOLLOW
                                                   THE INDIRECT OBJECT)
                            ))
        (TO VP/NP))
    (PUSH NP/ (AND (RFEAT INDOBJ V)
                   (GETR OBJ))

            (* A NP CAN OCCUR IN THIS POSITION IF THE VERB CAN
            TAKE AN INDIRECT OBJECT. WHAT WE THOUGHT WAS THE
            OBJECT WAS REALLY THE INDIRECT OBJECT, AND THE NP
            HERE IS TO BE THE DIRECT OBJECT.
            ('I GAVE JOHN THE INFORMATION' --> 'I GAVE THE
            INFORMATION TO JOHN'))

        (ADDL VMODS (BUILDQ (PP (PREP TO)
                                +)
                            OBJ))
        (SETR OBJ *)
        (TO VP/VP))
    (PUSH COMPL/ (AND (WRD (FOR THAT))
                      (RFEAT INDOBJ V))

            (* THE DIRECT OBJECT CAN ALSO BE A COMPLEMENT, FOR
            CERTAIN VERBS THAT ALLOW INDIRECT OBJECTS: 'I TOLD
            MARY THAT...')

        (ADDL VMODS (BUILDQ (PP (PREP TO)
                                +)
                            OBJ))
        (SETR OBJ NIL)
        (SETR COMPL *)
        (TO VP/VP))
    (JUMP VP/VP T
                                                  (* FINALLY, JUMP TO
                                                  VP/VP)))
```

```
(VP/V
   (CAT V T

        (* THE FIRST VERB CAN BE FOLLOWED BY OTHER VERBS TO
        FILL OUT THE PERFECT-PROGRESSIVE-PASSIVE AUXILIARY
        STRUCTURE. ADVERBS AND THE SUBJECT OF A
        THERE-INSERTED SENTENCE CAN BE INTERSPERSED BETWEEN
        THE VERBS--WE LOOP FOR THEM HERE.)


        (* VERBS AFTER THE MAIN VERB MUST BE PARTICIPLES OR
        UNTENSED FORMS. IF A PAST PARTICIPLE, THE PREVIOUS
        VERB IN THE SEQENCE MUST BE EITHER 'HAVE'
        (ASPECT=PERFECT) OR 'BE' (SENTENCE IS PASSIVE, IF
        POSSIBLE). A PRESENT PARTICIPLE MUST BE PRECEDED BY
        'BE' (ASPECT=PROGRESSIVE). OTHERWISE, THE CURRENT
        WORD MUST BE AN UNTENSED VERB AND THE REGISTER V
        MUST BE EMPTY (BECAUSE THE FIRST VERB WAS A MODAL).
        IF ANY OF THESE CONDITIONS IS SATISFIED, WE REPLACE
        THE VERB BY THE ROOT FORM OF THE CURRENT WORD.)


   (COND
     ((GETF PASTPART)
       (COND
         ((AND (WRD BE V)
               (VPASSIVE *))
           (HOLD (GETR SUBJ)
                 (GETR NPFEATURES))
           (SETR SUBJ (BUILDQ (NP (PRO SOMETHING))))
           (SETR AGFLAG T))
         ((AND (NULLR ASPECT)
               (WRD HAVE V))
           (SETRQ ASPECT (PERFECT)))
         (T (ABORT))))
     ((GETF PRESPART)
       (COND
         ((WRD BE V)
           (ADDR ASPECT (QUOTE PROGRESSIVE)))
         ((WRD POSS-ING TYPE))
         (T (ABORT))))
     ((OR (NOT (GETF UNTENSED))
          (GETR V))
       (ABORT)))
   (SETR V *)
   (TO VP/V))
```

```
(CAT ADJ (RFEAT COPULA V)
     (SETR V (BUILDQ (@ (ADJ)                    (*)
                        # #)
                     (COND
                       ((GETF COMPARATIVE)
                         (QUOTE (COMPARATIVE)))
                       ((GETF SUPERLATIVE)
                         (QUOTE (SUPERLATIVE))))
                     (COND
                       ((WRD (APPEAR SEEM)
                             V)
                         (QUOTE (SEEMING)))))))
```

(* A PREDICATE ADJECTIVE (SIMPLE OR INFLECTED) REPLACES A COPULA AS DS VERB. IF THE COPULA WAS "APPEAR" OR "SEEM", THE FEATURE "SEEMING" IS ADDED TO THE NEW VERB.)

```
     (TO VP/ADJ))
(MEM (MORE MOST)
     (AND (RFEAT COPULA V)
          (GETP (NEXTWRD)
                (QUOTE ADJ)))
     (SETR MORE-MOST *)
     (TO VP/COMP-ADJ))
(PUSH NP/ (AND (GETR THERE)
               (NULLR SUBJ)
               (WRD (BE EXIST)
                    V))
                                        (* HERE WE PUSH FOR THE
                                        SUBJECT OF A
                                        THERE-INSERTED
                                        SENTENCE.)
     (COND
       ((NOT (PNCHECK * (GETR PNCODE)))
         (ABORT)))
     (SETR SUBJ *)
     (TO VP/V))
(JUMP VP/HEAD (GETR SUBJ)
```

(* IF WE HAVE THE SUBJECT, WE CAN ASSUME THAT WE ALSO HAVE THE MAIN VERB (USUALLY, THIS WILL BE TRUE BECAUSE WE WOULD HAVE LOOPED THROUGH THE FIRST ARC AS LONG AS POSSIBLE) AND JUMP TO VP/HEAD TO LOOK FOR POST-VERB CONSTITUENTS. IF THE V REGISTER IS EMPTY (THE FIRST AND ONLY VERB WAS A MODAL), WE ABORT UNLESS THE MODAL WAS 'DO'--WE ALLOW 'DO' TO BECOME THE MAIN VERB.)

```
      (COND
        ((NULLR V)
          (COND
            ((WRD DO MODAL)
              (SETRQ V DO)
              (SETR MODAL NIL))
            (T (ABORT))))
        (T (COND
             ((AND (GETR THERE)
                   (WRD BE V))
              (SETRQ V EXIST)))))
      (SETR HEAD (GETR V)))
   (CAT ADV T
      (ADDL VMODS (BUILDQ (ADV *)))
      (TO VP/V)))

(VP/VP
   (WRD BY (GETR AGFLAG)

           (* THE ELEMENTS OF THE VERB PHRASE WHICH ARE CLOSELY
           TIED TO THE MAIN VERB (E.G. COMPLEMENTS) HAVE BEEN
           PROCESSED. VARIOUS ADDITIONAL MODIFIERS ARE STILL
           PERMITTED (ADVERBS, PREP-PHRASES). ALSO, WE LOOK FOR
           THE AGENT OF PASSIVE SENTENCES AND THE OBJECT OR
           SUBJECT OF POSS-ING COMPLEMENTS, IF THESE HAVE NOT
           BEEN ALREADY IDENTIFIED.)



           (* AGFLAG IS SET IF WE HAVEN'T FOUND THE SUBJECT OF
           A PASSIVE SENTENCE. 'BY' CAN INTRODUCE IT.)


        (TO VP/AGT))
   (WRD BY (AND (WRD POSS-ING TYPE)
           (NULLR OBJ)
           (NULLR SUBFLAG))

(* IN A POSS-ING COMPLEMENT WHERE THE SUBJECT WAS
SENT DOWN FROM THE POSSESSIVE AT STATE NP/DET, THE
SENT SUBJECT MIGHT REALLY BE THE OBJECT IF NO OBJECT
WAS FOUND ('THE DUCK'S SHOOTING BY THE HUNTERS ...')
AND THE REAL SUBJECT CAN FOLLOW A 'BY' IN THIS
POSITION.)
```

```
        (SETR OBJ (GETR SUBJ))
        (TO ING/BY))
    (MEM (OF BY)
        (GETR SUBFLAG)

            (* IN A POSS-ING COMPLEMENT, IF THE SUBJECT WAS NOT
            SENT DOWN FROM THE POSSESSIVE, IT CAN APPEAR
            FOLLOWING EITHER 'OF' OR 'BY': 'THE SHOOTING OF THE
            HUNTERS ...' OR 'THE SHOOTING BY THE HUNTERS')

        (SETR SUBFLAG NIL)
        (TO ING/BY))
    (CAT ADV T
        (ADDL VMODS (BUILDQ (ADV *)))
        (TO VP/VP))
        (PUSH PP/ (CAT PREP)
            (ADDL VMODS *)
            (TO VP/VP))
        (JUMP S/VP T))
)
STOP
```

Appendix C

Semantic Rules

```
  (PROGN (LISPXPRIN1 (QUOTE "FILE CREATED ")
                     T)
         (LISPXPRIN1 (QUOTE "11-JUN-72 12:29:Ø8")
                     T)
         (LISPXTERPRI T))
  (LISPXPRINT (QUOTE RULESCOMS)
              T)
  (RPAQQ RULESCOMS ((V: GENRULES)
           (R: TRULES)
           (V: NEWRULES)
           (V: TREEFRAGS)
           (V: RULELISTS)))
(LISPXPRINT (QUOTE (V: GENRULES)) T)
(RPAQQ GENRULES (ADJ:MASS ADJ:SET ANY:TERM D:ALL D:ALL-PL D:ALL\ONES
D:ANAPHORA D:ATLEAST D:ATMOST D:AVERAGE D:CARDINAL D:EACH D:EVERY
D:EXACTLY D:HOWMANY D:LESSTHAN D:MASS D:MAXIMUM D:MINIMUM D:MORETHAN
D:NEG D:NIL D:NO D:NOT-SET D:NUMBER D:OLDEST D:ORDINAL D:SEMI-ANAPHOR
D:SET1 D:SETOF D:SOME D:SSET D:THE-PL D:THE-SG D:THE-SG2 D:WHQ-PL
D:WHQ-SG D:WHR NP:NPR PR1 PR2 PR3 PR4 PR5 PR6 R:ADJ R:PP R:QREL R:REL
S:BE-AROUND S:BE-EQUAL S:BE-GREATER-VAL S:BE-LESS=VAL S:BE-MEMBER
S:BE-MEMBER* S:DCL S:IMP S:NEG S:NPQ S:NPU S:QREL S:QREL-NEG S:WHQ
S:YES/NO SS3Ø SS32 SS33 SS34 SS35 SS36 SS41))
(DEFINEV
(ADJ:MASS ((T T)
          ->
          (PROGN

          (* THE INTERPRETATION PRODUCED FOR A MASS NOUN
          MODIFYING ANOTHER NOUN (E.G. "THE SILICA PHASE") IS
          (NPR* X / (QUOTE ---)), WHERE THE STANDARD FORM OF
          THE MASS NOUN IS INSERTED IN THE SPACE.
          THE FUNCTION NPR FINDS THE SPANDARD FORM)

                  (SETQ SEM (SUBST (QUOTE (NPR* X / W))
                                   (QUOTE DLT)
                                   QUANT))
                  (SETQQ QUANT DLT)
                  (SETQ SEM
                    (SUBST (LIST (QUOTE QUOTE)
                                 (EVAL (CONS (QUOTE NPR)
                                             (QUOTE (# Ø TERM)))))
                           (QUOTE W)
                           SEM)))))
(ADJ:SET ((ADJ.NP (MEM 1 SET))
          ->
          (PROG1 (SETQ SEM
                   (SUBLIS (QUOTE ((FOR . UNION)
                                   (DLT (# 1 1 NRULES))))
                           QUANT))
                 (SETQQ QUANT DLT
```

```
            (* ADJ:SET MATCHES A NOUN-NOUN MODIFIER WHEN IT
            REFERS TO A SET, E.G. "RARE-EARTH ANALYSES".
            RARE-EARTH REFERS TO THE SET OF ELEMENTS WITH ATOMIC
            NUMBERS 58 THROUGH 71.0 THE INTERPRETATION PRODUCED
            IS SIMILAR TO THAT FOR "ANALYSES OF RARE-EARTHS":
            (FOR EVERY X6 / (SEQ RARE-EARTHS): T ;
            DLT).)

))))
(ANY:TERM ((T T)
           ->
           (LIST (QUOTE QUOTE)
                 (EVAL (CONS (QUOTE NPR)
                             (QUOTE (# 0 TERM))

            (* ANY:TERM IS USED TO INTERPRET VERBS, ADVERBS,
            INTEGERS, PROPER NOUNS AND ADJECTIVES.
            THE INTERPRETATION IS (QUOTE ---), WHERE THE SPACES
            ARE FILLED BY THE STANDARD FORM OF THE WORD,
            COMPUTED BY NPR.)


                                    )))))
(D:ALL ((NP.DET T)
        ->
        (QUANT (FOR EVERY X / (# 0 NRULES)
                    :
                    (# 0 RRULES)
                    ; DLT))))
(D:ALL-PL ((NP.DET (AND (EQU 1 ALL)
                        (EQU 2 PL)))
           ->
           (PROGN (DSUBST (QUOTE EVERY)
                          (QUOTE GEN)
                          QUANT)
                  (QUANT (FOR EVERY X / (# 0 NRULES)
                              :
                              (# 0 RRULES)
                              ; DLT)))))
(D:ALL\ONES ((NP.DET (EQU 1 ALL))
             (NP.PRO (EQU 1 ONES))
             (NP.PP (EQU 1 OF))
             ->
             (QUOTE (# 3 2 ALL)

            (* D:ALL\ONES INVOKES THE RULE D:ALL ON THE NODE'S
            DEPENDENT PREPOSITIONAL PHRASE VIA THE TYPEFLAG "ALL".
            E.G. ON THE PP OF "ALL OF THE TYPE/A SAMPLES", THE
            INTERPRETATION IS SUCH THAT THE QUANTIFIER COMES
            FORM "ALL" AND THE CLASS AND RESTRICTIONS FROM THE
            DEPENDENT NP.)
```

```
                    )))
(D:ANAPHORA ((OR (NP.PRO (NOT (OR (EQU 1 I)
                                  (EQU 1 YOU)
                                  (EQU 1 ONES))))
                 (NP.DET (OR (EQU 1 THIS)
                             (EQU 1 THAT)
                             (EQU 1 THESE)
                             (EQU 1 THOSE))))
             (NOT (NP.PP T))
             (NOT (NP.REL T))
             ->
             (PROGN (SETQ QVAR (ANTECEDANT (QUOTE (# Ø IDENTITY)))

            (* D:ANAPHORA MATCHES ANAPHORIC NP'S NOT MODIFIED BY
            PREPOSITIONAL PHRASES OR RELATIVE CLAUSES, E.G. "IT",
            "THOSE BARIUM ANALYSES". THE INTERPRETATION DEPENDS
            ON THE ANTECEDANT. SEE THE FUNCTION DESCRIPTIONS OF
            ANTECEDANT, ANTEQUANT AND SCOPEVARS FOR FURTHER
            EXPLANATION.)


                       )
                      (MAPC (SCOPEVARS QVAR)
                            (FUNCTION ANTEQUANT))
                      (ANTEQUANT QVAR))))
(D:ATLEAST ((NP.DET.COMP (OR (EQU 1 ATLEAST)
                             (EQU 1 ASMANYAS)))
            ->
            (QUANT (FOR (EQ N (# 1 2))
                        X / (# Ø NRULES)
                        :
                        (# Ø RRULES)
                        ; DLT))))
(D:ATMOST ((NP.DET.COMP (EQU 1 ATMOST))
           ->
           (QUANT (NOT (FOR (GREATER N (# 1 2))
                            X / (# Ø NRULES)
                            :
                            (# Ø RRULES)
                            ; DLT)))))
(D:AVERAGE ((T T)
            ->
            (QUOTE (SEQL (AVERAGE X / (# Ø NRULES)
                                  :
                                  (# Ø RRULES))))))
(D:CARDINAL ((NP.DET.INTEGER T)
             ->
             (QUANT (FOR (EQ N (# 1 1 INTEGER))
                         X / (# Ø NRULES)
                         :
                         (# Ø RRULES)
                         ; DLT))))
```

```
(D:EACH ((NP.DET (AND (EQU 1 EACH)
                      (EQU 2 SG)))
         ->
         (QUANT (FOR EVERY X / (# Ø NRULES)
                     :
                     (# Ø RRULES)
                     ; DLT))))
(D:EVERY ((NP.DET (AND (EQU 1 EVERY)
                       (EQU 2 SG)))
          ->
          (QUANT (FOR EVERY X / (# Ø NRULES)
                      :
                      (# Ø RRULES)
                      ; DLT))))
(D:EXACTLY ((NP.DET.COMP (EQU 1 EXACTLY))
            ->
            (QUANT (FOR (EQ N (# 1 2))
                        X / (# Ø NRULES)
                        :
                        (# Ø RRULES)
                        ; DLT))))
(D:HOWMANY ((NP.DET (AND (EQU 1 HOWMANY)
                         (OR (EQU 2 PL)
                             (EQU 2 SG/PL))))
            ->
            (QUANT (FOR THE X / (# Ø NUMBER)
                        : T ; (PRINTOUT X)))))
(D:LESSTHAN ((NP.DET.COMP (OR (EQU 1 FEWERTHAN)
                              (EQU 1 LESSTHAN)))
             ->
             (QUANT (NOT (FOR (EQ N (# 1 2))
                              X / (# Ø NRULES)
                              :
                              (# Ø RRULES)
                              ; DLT)))))
(D:MASS ((NP.N (OR (MEM 1 (MASS)

          (* D:MASS MATCHES A MASS NOUN AND PRODUCES AS ITS
          INTERPRETATION, A VARIABLE ASSOCIATED WITH THE
          STANDARD FORM OF THE NOUN. E.G. "ALUMINUM" IS
          INTERPRETED AS (NPR* X / (QUOTE AL203)))


                 )
              (EQ (GETP (CAR (TERM (CONSTITUENTS (# 1))))
                        (QUOTE N))
                  (QUOTE MASS))))
->
(PROGN (SETQ SEM (SUBST (QUOTE (NPR* X / W))
                        (QUOTE DLT)
                        QUANT))
       (SETQQ QUANT DLT)
       (SETQ SEM (SUBST (LIST (QUOTE QUOTE)
                              (TABFORM (# Ø HEAD)))
                        (QUOTE W)
                        SEM)))))
```

```
(D:MAXIMUM ((T T)
            ->
            (QUOTE (SEQL (MAXIMUM X / (# Ø NRULES)
                                  :
                                  (# Ø RRULES))))))
(D:MINIMUM ((T T)
            ->
            (QUOTE (SEQL (MINIMUM X / (# Ø NRULES)
                                  :
                                  (# Ø RRULES))))))
(D:MORETHAN ((NP.DET.COMP (EQU 1 MORETHAN))
             ->
             (QUANT (FOR (GREATER N (# 1 2))
                         X / (# Ø NRULES)
                         :
                         (# Ø RRULES)
                         ; DLT))))
(D:NEG ((NP.NEG T)
        ->
        (QUANT (NOT (# 1 1)))))
(D:NIL ((NP.DET (EQU 1 NIL))
        ->
        (QUANT (FOR GEN X / (# Ø NRULES)
                    :
                    (# Ø RRULES)
                    ; DLT))))
(D:NO ((NP.DET (EQU 1 NO))
       ->
       (QUANT (NOT (FOR SOME X / (# Ø NRULES)
                        :
                        (# Ø RRULES)
                        ; DLT)))))
(D:NOT-SET ((T T)
            ->
            (QUOTE (# Ø)

        (* D:NOT-SET IS A DEFAULT RULE USED IF AN NP NODE
        CANNOT BE INTERPRETED AS REFERRING TO A SET.
        AS A RESULT, THE NORMAL DRULES FOR THAT NODE ARE
        MATCHED.)


                    )))
(D:NUMBER ((T T)
           ->
           (SSUNIONF (SEQL (NUMBER X / (# Ø NRULES)
                                   :
                                   (# Ø RRULES))))))
```

```
(D:OLDEST ((T T)
           ->
           (QUOTE (SEQL (OLDEST X / (# Ø NRULES)
                                :
                                (# Ø RRULES))))))
(D:ORDINAL ((NP.DET.ART (EQU 1 THE))
            (NP.DET.POSTART (NUMBERP (EVAL (CADADR (# 1 TERM)))))
            ->
            (QUANT (FOR (ORDINAL (# 2 1 TERM))
                        X / (# Ø NRULES)
                        :
                        (# Ø RRULES)
                        ; DLT)

          (* D:ORDINAL MATCHES DETERMINERS WHICH ARE ORDINAL
          NUMBERS, E.G.
          "THE THIRD POTASSIUM ANALYSIS FOR SAMPLE 1ØØØ3" THE
          INTERPRETATION IS (FOR (ORDINAL 3) X /
          (DATALINE (WHQFILE S1ØØØ3) S1ØØØ3 OVERALL K2O): T ;
          DLT))


                    )))
(D:SEMI-ANAPHOR ((NP.PRO (NOT (OR (EQU 1 I)
                                  (EQU 1 YOU)
                                  (EQU 1 ONES))))
                 (NP.PP T)
                 ->
                 (PROG1 (SEMIANAPHOR (QUOTE (# Ø IDENTITY))

          (* D:SEMI-ANAPHOR MATCHES ON TYPE OF PARTIAL
          ANAPHORA, PRONOUNS MODIFIED BY PREPOSITIONAL
          PHRASES. E.G. "GIVE ME THOSE FOR S1ØØØ3" THE
          INTERPREATION DEPENDS ON THE ANTECEDANT.
          SEE THE FUNCTION DESCRIPTION OF SEMIANAPHOR FOR
          FURTHER DETAIL.)


                                                  ))))
(D:SET1 ((NP.N (MEM 1 SET))
         (NP.DET (OR (EQU   THE)
                     (EQU   NIL)
                     (EQU   A)))
         ->
         (PROG1 (SETQ SEM (SUBLIS (QUOTE ((FOR . UNION)
                                          (DLT SETLIST X /
                                               (# Ø NRULES)
                                               :
                                               (# Ø RRULES)
                                               ; T)))
                                  QUANT)
```

```
            (* D:SET1 MATCHES AN NP INTERPRETABLE AS A SET, E.G.
            "THE TYPE/B ROCKS WHICH CONTAIN SILICA" THE
            INTERPRETION IS THE LIST OF OBJECTS IN THECLASS
            MEETING THE GIVEN RESTRICTIONS E.G.
            (SETLIST X / (SEQ TYPEBS):
            (CONTAIN X SILICA); T))


                        )
                    (SETQQ QUANT DLT))))
(D:SETOF ((NP.DET (AND (OR (EQU 1 THE)
                           (EQU 1 ALL)
                           (EQU 1 NIL))
                       (EQU 2 PL)))
          ->
          (PROG1 (SETQ SEM (SUBLIS (QUOTE ((FOR . UNION)
                                           (DLT SETOF X / (# Ø NRULES)
                                                :
                                                (# Ø RRULES)
                                                ; T)))
                                   QUANT)

            (* D:SETOF INVOKES THE INTERPRETATION OF THE NODE AS
            A RESTRICTED SET, PRODUCING A SINGLE SUCCESSOR
            FUNCTION FOR THE CLASS OF THE NODE AND ITS
            RESTRICTIONS. E.G., IT WOULD PRODUCE FOR
            "THE BRECCIAS WHICH CONTAIN KRYPTON" THE
            INTERPRETATION (SETOF X / (SEQ TYPECS):
            (CONTAIN X (QUOTE KR)); T))


                        )
                    (SETQQ QUANT DLT))))
(D:SOME ((NP.DET (OR (EQU  : SOME)
                     (EQU  : A)
                     (EQU    AN)
                     (EQU  : ANY)))
          ->
          (QUANT (FOR SOME X / (# Ø NRULES)
                      :
                      (# Ø RRULES)
                      ; DLT))))
(D:SSET ((NP.DET (OR (EQU    EVERY)
                     (EQU 2 SG/PL)
                     (EQU 2 PL)))
          ->
          (QUANT (SSUNION X / (# Ø NRULES)
                          :
                          (# Ø RRULES)
                          ; DLT))))
```

```
(D:THE-PL ((NP.DET (AND (EQU 1 THE)
                        (OR (EQU 2 PL)
                            (EQU 2 SG/PL))))
           ->
           (QUANT (FOR EVERY X / (# Ø NRULES)
                       :
                       (# Ø RRULES)
                       ; DLT))))
(D:THE-SG ((NP.DET (AND (OR (EQU 1 THE)
                            (EQU 1 THIS)
                            (EQU 1 THAT))
                        (EQU 2 SG)))
           ->
           (QUANT (FOR THE      (#    NRULE )
                       :
                       (# Ø RRULES)
                       ; DLT))))
(D:THE-SG2 ((NP.DET (AND (EQU 1 THE)
                         (EQU 2 SG)))
            (NP.N (MEM 1 (NONSPECIFIC)))
            ->
            (QUANT (FOR EVERY X / (# Ø NRULES)
                        :
                        (# Ø RRULES)
                        ; DLT)

           (* D:THE-SG2 MATCHES DEFINITELY DETERMINED SINGULAR
           NOUN PHRASES WHICH DO NOT HAVE SINGLE REFERENTS.
           FOR EXAMPLE, "THE AGE OF S1ØØØ7" DOES NOT HAVE A
           SINGLE REFERENT, BUT RATHER SEVERAL, ONE FOR EACH
           MEASURING TECHNIQUE EMPLOYED.)


                       )))
(D:WHQ-PL ((NP.DET (AND (OR (EQU 1 WHICH)
                            (EQU 1 WHAT)
                            (EQU 1 WHQ)
                            (EQU 1 WHICHQ))
                        (EQU 2 PL)))
           ->
           (PROGN (LSUBST (QUOTE EVERY)
                          (QUOTE GEN)
                          QUANT)
                  (QUANT (FOR EVERY X / (# Ø NRULES)
                              :
                              (# Ø RRULES)
                              ;
                              (PRINTOUT X))))))
```

```
(D:WHQ-SG ((NP.DET (AND (OR (EQU 1 WHQ)
                            (EQU 1 WHICHQ)
                            (EQU 1 WHICH)
                            (EQU 1 WHAT))
                        (EQU 2 SG)))
           ->
           (QUANT (FOR THE X / (# Ø NRULES)
                       :
                       (# Ø RRULES)
                       ;
                       (PRINTOUT X)))))
(D:WHR ((NP.DET (EQU 1 WHR))
        ->
        (QUOTE (# Ø)

          (* THE INTERPREATION OF THE RELATIVE NP IN A
          RELATIVE CLAUSE IS THE VARIABLE ATTACHED TO IT IN
          THE MATRIX SENTENCE.)


                )))
(NP:NPR ((NP.NPR T)
         ->
         (PROGN (SETQ SEM (SUBST (QUOTE (NPR* X / W))
                                 (QUOTE DLT)
                                 QUANT)

          (* THE INTERPRETION OF A PROPER NOUN IS A VARIABLE
          ASSOCIATED WITH THE NOUN IN ITS STANDARD FORM.
          E.G. "NASA" IS INTERPRETED AS
          (NPR* X / (QUOTE NASA)))


                )
                (SETQQ QUANT DLT)
                (SETQ SEM
                  (SUBST (LIST (QUOTE QUOTE)
                               (EVAL (CONS (QUOTE NPR)
                                           (QUOTE (# 1 1 TERM)))))
                         (QUOTE W)
                         SEM)))))
(PR1 ((S.Q-NEG (NOT (LEAFMEMB P
                              (QUOTE (WHQ WHICHQ WHEN WHERE WHY HOW
                                          HOWMANY)))))
       ->
       (PRED (TEST (# Ø SRULES))                    (* PR1 MATCHES YES-NO
                                                    QUESTIONS PHRASED IN THE
                                                    NEGATIVE)

             )))
```

```
(PR2 ((S.Q (NOT (LEAFMEMB P
                          (QUOTE (WHQ WHICHQ WHEN WHERE WHY HOW HOWMANY)
                                 ))))
       (NOT (S.Q-MODAL T))
       -->
       (PRED (TEST (# Ø SRULES))                    (* PR2 MATCHES YES-NO
                                                    QUESTIONS. THE
                                                    INTERPRETATION IS

"TEST THE VALIDITY OF THE PROPOSITION EXPRESSED BY THE SENTENCE.")

(PR3 ((S.NEG T)
       -->
       (PRED (NOT (# Ø SRULES))

            (* PR3 MATCHES REQUESTS VOICED IN THE NEGATIVE THAT
            ARE NOT YES-NO QUESTIONS, E.G.
            "WHICH SAMPLES DO NOT CONTAIN SILICA?")


              )))
(PR4 ((OR ((S (DCL))
           T)
          ((S (REL))
           T)
          ((S (POSS-ING))
           T))
       -->
       (PRED (# Ø SRULES)

            (* PR4 MATCHES DECLARATIVE SENTENCES AND RELATIVE
            AND POSSESSIVE-PARTICIPLE CLAUSES E.G.
            "S10003'S CONTAINING SILICA". THE INTERPRETATION IS
            THE PROPOSITION EXPRESSED BY THE SENTENCE OR
            CLAUSE.)


              )))
(PR5 ((S.IMP T)
       -->
       (PRED (DO (# Ø SRULES))                      (* PR5 MATCHES
                                                    IMPERRTIVE SENTENCES.)

              )))
(PR6 ((S.NP T)
       (S.VP T)
       -->
       (PRED (# Ø SRULES)                           (* PR6 MATCHES ALL
                                                    SENTENCES EXCEPT FOR
                                                    NOUN PHRASE UTTERANCES.)


              )))
```

```
(R:ADJ (AND (NP.ADJ (NOT (USED? (# 1))))
            (NP.N T)
            ->
            (PROG (ANS)
                  (COND
                    ((USED? (SETQ ANS (CDR (ASSOC 1 (CAR RVECTOR)))

          (* R:ADJ MATCHES ALL ADJECTIVES IN THE REQUEST WHICH
          HAVEN'T CONTRIBUTED TO ITS INTERPRETATION.
          IT CAUSES A MESSAGE TO BE PRINTED OUT TO THE USER TO
          THAT EFFECT, AND THAN ASKS HIM WHETHER IT IS SAFE TO
          IGNORE THOSE ADJECTIVES. THE USER IS GIVEN THE
          OPTIONS OF AGREEING, TERMINATING THE REQUEST, OR
          BREAKING IN ORDER TO INVESTIGATE THE MATTER
          FURTHER.)

                     ))
                     (RETURN T)))
                  (PRIN1 (QUOTE "I DO NOT UNDERSTAND ")
                         T)
                  (PRINT ANS T)
                  (PRIN1 (QUOTE "AS A MODIFIER OF ")
                         T)
                  (PRINT (CDR (ASSOC 1 (CADR RVECTOR))))
                  (PRIN1 (QUOTE "DO YOU WANT ME TO IGNORE IT?")
                         T)
                  (TERPRI T)
                  (COND
                    ((MEMB (SETQ ANS (READ))
                           (QUOTE (YES T TRUE)))
                      (PRINT (QUOTE OK)
                             T))
                    ((EQ ANS (QUOTE BREAK))
                      (BREAK1 T T SEMSUB))
                    (T (QUIT)))
T    )))
```

```
(R:PP !AND (NP.PP (NOT (MEANING? (# 2)                    (* R:PP ACTS LIKE AS
                                                          R:ADJ ON PREPOSITIONAL
                                                          PHRASES.)
                                  )))
           ->
           (PROG (ANS)
                 (PRIN1 (QUOTE "I DO NOT UNDERSTAND ")
                        T)
                 (PRINT (CDR (ASSOC 3 (CAR RVECTOR)))
                        T)
                 (PRIN1 (QUOTE "AS A MODIFIER OF ")
                        T)
                 (PRINT (CDR (ASSOC 4 (CAR RVECTOR)))
                        T)
                 (PRIN1 (QUOTE "DO YOU WANT ME TO IGNORE IT?")
                        T)
                 (TERPRI T)
                 (COND
                   ((MEMB (SETQ ANS (READ))
                          (QUOTE (YES T TRUE)))
                     (PRINT (QUOTE OK)
                            T))
                   ((EQ ANS (QUOTE BREAK))
                     (BREAK  T T SEMSUB))
                   (T (QUIT)))
                T    )))
(R:QREL ((NP.QREL (AND (RELTAG (# 1)

           (* R:QREL ACTS LIKE AS R:ADJ ON RELATIVE CLAUSES
           DERIVING FROM THE SURFACE STRUCTURE VERB PHRASES OF
           WH-QUESTIONS.)


                             )
                      (OR (INTERP (# 1))
                          (PROG (ANS)
                                (PRIN1 (QUOTE "I DO NOT UNDERSTAND ")
                                       T)
                                (PRINT (# 1)
                                       T)
                                (PRIN1 (QUOTE
                                      "DO YOU WANT ME TO IGNORE IT?")
                                       T)
                                (TERPRI T)
                                (COND
                                  ((MEMB (SETQ ANS (READ))
                                         (QUOTE (YES T TRUE)))
                                    (PRINT (QUOTE OK)
                                           T))
                                  ((EQ ANS (QUOTE BREAK))
                                    (BREAK1 T T SEMSUB))
                                  (T (QUIT)))
                               T    ))))
           ->
           (QUOTE (# 1 1))))
```

```
(R:REL (AND (NP.REL (AND (RELTAG (# 1)                    (* R:REL ACTS LIKE AS
                                                          R:ADJ ON RELATIVE
                                                          CLAUSES.)
                         )
                    (OR (INTERP (# 1))
                        (PROG (ANS)
                              (PRIN1 (QUOTE "I DO NOT UNDERSTAND ")
                                     T)
                              (PRINT (# 1)
                                     T)
                              (PRIN1 (QUOTE
                                  "DO YOU WANT ME TO IGNORE IT?")
                                     T)
                              (TERPRI T)
                              (COND
                                ((MEMB (SETQ ANS (READ))
                                       (QUOTE (YES T TRUE)))
                                 (PRINT (QUOTE OK)
                                        T))

                                (BREAK1 T T SEMSUB))
                              (T (QUIT)))
                        T    ))))
             ->
             (PRED (# 1 1))))
(S:BE-AROUND ((S.NP-V (AND (MEM 1 (ANALYSIS CONCENTRATION)
                                                          (* S:AROUND MATCHES
                                                          REQUESTS LIKE WHICH
                                                          ALUMINUM ANALYSES ARE
                                                          AROUND 7 PERCENT?)
                           )
                      (EQU 2 BE)))
              (S.COMP-N (AND (OR (EQU 1 AROUND)
                                 (EQU 1 APPROXIMATELY))
                             (MEM 3 UNIT)))
              ->
              (BUILDQ (AROUNDVAL (# 1 1)
                                 #)
                      (LIST (QUOTE QUOTE)
                            (CONS (# 2 2 INTEGER)
                                  (# 2 3 UNIT))))))
(S:BE-EQUAL ((S.NP-V (AND (EQU 2 BE)
                          (NOT (OR (EQU 1 WHQ THING SG)
                                   (EQU 1 WHQ THING PL)
                                   (EQU 1 WHQ THING SG/PL)))))
             (S.OBJ (AND (NOT (OR (EQU 1 WHQ THING SG)
                                  (EQU 1 WHQ THING PL)
                                  (EQU 1 WHQ THING SG/PL)))
                         (EQU 2 SG)))
             ->
             (PRED (EQUAL (# 1 1)
                          (# 2 1))
```

```
            (* S:BE-EQUAL MATCHES QUESTIONS OF THE FORM IS X Y?
            , E.G. IS S1ØØ72 THE OLDEST ROCK?)


                        )))
(S:BE-GREATER-VAL ((S.NP-V (AND (MEM 1 (ANALYSIS CONCENTRATION)
                                                (* S:BE-GREATER-VAL
                                                MATCHES REQUESTS LIKE
                                                ARE ALL ALUMINUM
                                                CONCENTRATIONS GREATER
                                                THAN 5 PERCENT?)
                                )
                           (EQU 2 BE)))
                   (S.COMP-N (AND (OR (EQU 1 MORETHAN)
                                      (EQU 1 GREATERTHAN))
                                  (MEM 3 UNIT)))
                   ->
                   (BUILDQ (GREATER (# 1 1)
                                    #)
                           (LIST (QUOTE QUOTE)
                                 (CONS (# 2 2 INTEGER)
                                       (# 2 3 UNIT)))))))
(S:BE-LESS-VAL ((S.NP-V (AND (MEM 1 (ANALYSIS CONCENTRATION)

            (* S:BE-LESS-VAL MATCHES REQUESTS LIKE IS THE
            AVERAGE CONCENTRATION OF ALUMINUM IN BRECCIAS LESS
            THAN 9 PERCENT?)


                                )
                           (EQU 2 BE)))
                   (S.COMP-N (AND (OR (EQU 1 LESSTHAN)
                                      (EQU 1 FEWERTHAN))
                                  (MEM 3 UNIT)))
                   ->
                   (BUILDQ (LESSVAL (# 1 1)
                                    #)
                           (LIST (QUOTE QUOTE)
                                 (CONS (# 2 2 INTEGER)
                                       (# 2 3 UNIT)))))))
(S:BE-MEMBER ((S.NP-V (AND (EQU 2 BE)
                           (NOT (OR (EQU 1 WHQ THING SG)
                                    (EQU 1 WHQ THING PL)
                                    (EQU 1 WHQ THING SG/PL)))))
              (S.OBJ (AND (NOT (OR (EQU 1 WHQ THING SG)
                                   (EQU 1 WHQ THING PL)
                                   (EQU 1 WHQ THING SG/PL)))
                          (OR (MEM 1 SET)
                              (EQU 2 PL)
                              (EQU 2 SG/PL))))
              ->
              (PRED (MEMBER (# 1 1)
                            (# 2 1 SET?))
```

(* S:BE-MEMBER MATCHES QUESTIONS OF THE FORM IS X A Y? AND ARE X'S Y'S? E.G. IS S10003 A BRECCIA?)

```
                    )))
(S:BE-MEMBER* ((S.NP-V (EQU 2 BE))
               (S.OBJ (MEM 1 TYPE))
               ->
               (QUOTE (MEMBER* (# 1 1)
                               (# 2 1))
```

(* S:BE-MEMBER* MATCHES QUESTIONS OF THE FORM WHAT KIND OF X IS Y? AND IS Y A TYPE OF X? E.G. WHAT KIND OF ROCK IS S10003?)

```
                    )))
(S:DCL ((S.DCL-S T)
        ->
        (PRED (# 1 1)
```

(* S:DCL, S:IMP, S:NEG, S:WHQ, AND S:YES-NO MATCH S NODES OF THE FORM (S (DCL (S ...))), (S (IMP (S ...))), (S (NEG (S ...))), AND (S (Q (S ...))), OTHER POSSIBLE DEEP STRUCTURES FOR SENTENCES THAT THE PARSER MIGHT BE REQUIRED TO PRODUCE.)

```
                )))
(S:IMP ((S.IMP-S T)
        ->
        (PRED (DO (# 1 1)))))
(S:NEG ((S.NEG-S T)
        ->
        (PRED (NOT (1 . 1)))))
(S:NPQ ((S.NPQ T)
        ->
        (PRED (# 1 1 REFS?)                    (* S:NPQ MATCHES NOUN
                                                PHRASE QUESTIONS, E.G.
                                                WHICH BARIUM ANALYSES?)
                )))
(S:NPU ((S.NPU T)
        ->
        (PRED (PRINTOUT (# 1 REFS?)))))
(S:QREL (((S (QREL))
          T)
         ->
         (QUOTE (# 0 SRULES)
```

(* S:QREL MATCHES RELATIVE CLAUSES DERIVING FROM THE SURFAC STRUCTURE VERB PHRASES OF WH-QUESTIONS.)

```
                    )))
(S:QREL-NEG (((S (QREL NEG))
              T)
             ->
             (QUOTE (NOT (# 0 SRULES))          (* S:QREL-NEG DOES THE
                                                SAME FOR WH-QUESTIONS
                                                PHRASED IN THE
                                                NEGATIVE.)

                    )))
(S:WHQ ((S.Q-S (LEAFMEMB P (QUOTE (WHQ WHICHQ WHEN WHERE WHY HOW
                                   HOWMANY))))
       ->
       (PRED (# 1 1))))
(S:YES/NO ((S.Q-S (NOT (LEAFMEMB P
                                 (QUOTE (WHQ WHICHQ WHEN WHERE WHY HOW
                                         HOWMANY)))))
           ->
           (PRED (TEST (#  1)))))
(SS30 ((S.NP-V (EQU 2 BE))
       (S.OBJ (OR (EQU 1 WHQ THING SG)
                  (EQU 1 WHQ THING PL)
                  (EQU 1 WHQ THING SG/PL)
                  (EQU 1 WHAT)))
       ->
       (PRED (PRINTOUT (1 . 1))

          (* SS30 MATCHES QUESTIONS OF THE FORM WHAT IS ...
          AND WHAT ARE ...?)


                )))
(SS32 ((S.NP-V (AND (MEM 1 INTEGER)
                    (EQU 2 BE)))
       (S.COMP (AND (OR (EQU 1 AT LEAST)
                        (EQU 1 AS MANY AS)
                        (EQU 1 ATLEAST)
                        (EQU 1 ASMANYAS))
                    (MEM 2 INTEGER)))
       ->
       (PRED (NOT (GREATER (2 . 2)
                           (1 . 1))))))
(SS33 ((S.NP-V (AND (MEM 1 INTEGER)
                    (EQU 2 BE)))
       (S.COMP (AND (OR (EQU 1 MORE THAN)
                        (EQU 1 MORETHAN))
                    (MEM 2 INTEGER)))
       ->
       (PRED (GREATER (1 .  )
                      (2 . 2))))))
```

```
(SS34 ((S.NP-V (AND (MEM 1 INTEGER)
                    (EQU 2 BE)))
       (S.COMP (AND (EQU 1 EXACTLY)
                    (MEM 2 INTEGER)))
       ->
       (PRED (EQUAL (1 . 1)
                    (2 . 2)))))
(SS35 ((S.NP-V (AND (MEM 1 INTEGER)
                    (EQU 2 BE)))
       (S.COMP (AND (OR (EQU 1 FEWER THAN)
                        (EQU 1 LESS THAN)
                        (EQU 1 FEWERTHAN)
                        (EQU 1 LESSTHAN))
                    (MEM 2 INTEGER)))
       ->
       (PRED (GREATER (2 . 2)
                      (1 .  )))))
(SS36 ((S.NP-V (AND (MEM 1 INTEGER)
                    (EQU 2 BE)))
       (S.COMP (AND (OR (EQU 1 AT MOST)
                        (EQU 1 ATMOST))
                    (MEM 2 INTEGER)))
       ->
       (PRED (NOT (GREATER (1 . 1)
                           (2 . 2))))))
(SS41 ((S.NP-V (OR (EQU 2 BE)
                   (EQU 2 EXIST)))
       ->
       (PRED (EXIST (1 . 1))

          (* SS41 MAICHES REQUESTS OF THE FORM IS THERE AN X?
          AND ARE THERE Y'S?)


             )))
)
(LISPXPRINT (QUOTE (R: TRULES)) T)
(RPAQQ TRULES (TOPIC\ADJ TOPIC\ADJ-N TOPIC\ADJ-NPR TOPIC\ADJ.NP
TOPIC\AND-NP TOPIC\AND-S TOPIC\ESP TOPIC\N TOPIC\NOM TOPIC\NR.S
TOPIC\NR.NP TOPIC\NOT-NP TOPIC\NOT-S TOPIC\NPR TOPIC\OR-NP TOPIC\OR-S
TOPIC\PP TOPIC\REL TOPIC\S.COMPL TOPIC\S.NP TOPIC\S.OBJ TOPIC\S.PP
TOPIC\S.V TOPIC\TERM TOPIC\TERM2 TOPIC\AUTHOR TOPIC\AUTHOR2
TOPIC\NP.COMPL TOPIC\PP.COMPL TOPIC\V-INTRANS TOPIC\V-TRANS
TOPIC\V-TRANS2 TOPIC\EMPHASIS TOPIC\ADJ.SUPER TOPIC\ADJ.COMP))
(DEFINEG

[TOPIC\ADJ
    AND
    (NP.ADJ (AND (NOT (MEM  DOCUMENT))
                 (NOT (EQ (CAR (# 1))
                          (QUOTE NP)))
                 (NOT (MEM  PADDING))))
        ->  (QUOTE (# 1 1 TERM))  ]
```

```
[TOPIC\ADJ-N
     OR
     (NP.ADJ-N (NOT (AND (OR (MEM 1 DOCUMENT)
                             (MEM 1 PADDING))
                         (MEM 2 DOCUMENT))))
     (OR (NP.PP T)
         (NP.REL T)
         (NP.COMPL T))
         -> (ADJPHRSE (QUOTE (# 0 IDENTITY)))  ]

[TOPIC\ADJ-NPR
     OR
     (NP.NPR T)
     (NP.ADJ T)
     (OR (NP.PP T)
         (NP.REL T)
         (NP.COMPL T))
         -> (ADJPHRSE (QUOTE (# 0 IDENTITY)))  ]

[TOPIC\ADJ.NP
     AND
     (NP.ADJ.NP T)
         -> (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\AND-NP
     AND
     (NP.AND T)
         -> (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\AND-S
     AND
     (S.AND T)
         -> (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\ESP
     (NP.ADVP (MEM 1 TRANSADV))
         -> (*FLAG (QUOTE (# 1 2 TOPIC)))  ]

[TOPIC\N
     (NP.N (AND (NOT (MEM 1 PADDING))
                (NOT (MEM 1 DOCUMENT))))
          >

[TOPIC\NOM
     (NP.NOM T)
         -> (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\NR.S
     (NP.NR.S T)
         -> (QUOTE (# 1 1 TOPIC))  ]
```

```
[TOPIC\NR.NP
    (NP.NR.NP T)
        ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\NOT-NP
    (NP.NEG T)
        ->  (QUOTE (NOT (# 1 1 TOPIC)))  ]

[TOPIC\NOT-S
    (S.NEG T)
        ->  (QUOTE (NOT (# 1 1 TOPIC)))  ]

[TOPIC\NPR
    (NP.NPR T)
        ->  (QUOTE (# 1 1 TERM))  ]

[TOPIC\OR-NP
    OR
    (NP.OR T)
        ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\OR-S
    OR
    (S.OR T)
        ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\PP
    AND
    (OR (NP.PP (NOT (EQ (CAADDR (# 2))
                        (QUOTE COMPL))))
        (NP.PP.AND.PP (NOT (EQ (CAADDR (# 2))
                               (QUOTE COMPL)))))
        ->  (QUOTE (# 1 2 TOPIC))  ]

[TOPIC\REL
    AND
    (NP.REL T)
        ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\S.COMPL
    (S.COMPL T)
        ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\S.NP
    (S.NP (AND (NOT (MEM 1 PADDING))
               (NOT (MEM 1 DOCUMENT))))
    (NOT (S.PRO T))
        ->  (QUOTE (# 1 1 TOPIC))  ]
```

```
[TOPIC\S.OBJ
    (OR (S.OBJ (AND (NOT (MEM 1 PADDING))
                    (NOT (MEM 1 DOCUMENT))))
        (S.OBJ.REL T))
        -> (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\S.PP
    AND
    (OR (S.PP T)
        (S.PP.AND.PP T))
        -> (QUOTE (# 1 2 TOPIC))  ]

[TOPIC\S.V
    (S.V NIL)
        -> (LIST (PACK (NCONC (QUOTE (# 1 1 TERM)) (QUOTE (ING)))))  ]
```

```
[TOPIC\TERM
    (OR (NP.N (AND (LESSP (LENGTH (KEYPHRASE (LIST P)))
                          5)
                   (GREATERP (LENGTH (KEYPHRASE (LIST P)))
                             1)
                   (NOT (MEM 1 PADDING))
                   (NOT (MEM 1 DOCUMENT))))
        (NP.NPR (AND (LESSP (LENGTH (KEYPHRASE (LIST P)))
                            5)
                     (GREATERP (LENGTH (KEYPHRASE (LIST P)))
                               1))))
    (NOT (NP.REL T))
        -> (KEYPHRASE (LIST (QUOTE (# 0 IDENTITY))))  ]

[TOPIC\TERM2
    (NP.N (AND (LESSP (LENGTH (KEYPHRASE (LIST P)))
                      5)
               (GREATERP (LENGTH (KEYPHRASE (LIST P)))
                         1)
               (OR (MEM 1 PADDING)
                   (MEM 1 DOCUMENT))))
    (NP.ADJ (AND (NOT (MEM 1 PADDING))
                 (NOT (MEM 1 DOCUMENT))))
    (NOT (NP.REL T))
        -> (KEYPHRASE (LIST (QUOTE (# 0 IDENTITY))))  ]

[TOPIC\AUTHOR
    AND
    (OR (S.AND.NPR T)
        (S.NPR T))
    (S.V (MEM 1 WRITE))
        -> (AUTHOR: (QUOTE (# 1 1 TERM)))  ]
```

```
[TOPIC\AUTHOR2
     AND
     (NP.N (MEM 1 DOCUMENT))
     (NP.PP.NPR (EQU 1 BY))
          ->  (AUTHOR: (QUOTE (# 2 2 TERM)))  ]

[TOPIC\NP.COMPL
     (NP.COMPL T)
          ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\PP.COMPL
     (NP.PP.COMPL T)
          ->  (QUOTE (# 1 1 TOPIC))  ]

[TOPIC\V-INTRANS
     (NOT (S.OBJ T))
     (S.V T)
          ->  (APPEND (LIST (QUOTE V:)) (QUOTE (# 2 1 TERM)))  ]

[TOPIC\V-TRANS
     (S.OBJ (NOR (MEMB (QUOTE AND)
                       (# 1))
                 (MEMB (QUOTE OR)
                       (# 1))))
     (S.V T)
          ->  (APPEND (APPEND (LIST (QUOTE VP:)) (LIST (NFLCT-ING (CAR
(QUOTE (# 2 1 TERM)))))) (ADJPHRSE (QUOTE (# 1 1 IDENTITY))))  ]

[TOPIC\V-TRANS2
     OR
     (OR (S.OBJ.AND T)
          (S.OBJ.OR T))
     (S.V T)
          ->  (APPEND (APPEND (LIST (QUOTE VP:)) (LIST (NFLCT-ING (CAR
(QUOTE (# 2 1 TERM)))))) (ADJPHRSE (QUOTE (# 1 1 IDENTITY))))  ]

[TOPIC\EMPHASIS
     (NP.PP (MEM 2 EMPHASIS))
     (NP.PP.PP T)
          ->  (*FLAG (QUOTE (# 2 2 TOPIC)))  ]

[TOPIC\ADJ.SUPER
     AND
     (NP.DET.POSTART (MEMB (QUOTE SUPERLATIVE)
                           (CADR (# 1))))
     (NP.N T)
          ->  (APPEND (LIST (NFLCT-ADJ (CADR (QUOTE (# 1 1 TERM))) (QUOTE
SUPERLATIVE))) (QUOTE (# 2 1 TERM)))  ]
```

```
[TOPIC\ADJ.COMP
     AND
     (NP.ADJ.COMP T)
     (NP.N T)
          ->  (APPEND (LIST (NFLCT-ADJ (CAR (QUOTE (# 1 1 TERM))) (QUOTE
COMPARATIVE))) (QUOTE (# 2   TERM)))  ]
)
(LISPXPRINT (QUOTE (V: NEWRULES)) T)
(RPAQQ NEWRULES (N:AGE N:AGE' N:ANALYSIS N:AVERAGE N:AVG-CONC? N:BASALT
N:CONCENTRATION N:CORETUBE N:DOCUMENT N:DUST N:ELT N:GABBRO N:HALOGEN
N:LINE# N:MAJOR-ELT N:MAXIMUM N:MINERAL N:MINIMUM N:MODAL-ANALYSIS
N:MODAL-CONC N:NUMBER N:OLDEST N:ONE N:ONEOF N:ONES N:ONES-OF-PRO
N:PHASE N:RARE/EARTH N:RATIO N:ROCK N:ROCKTYPE N:SAMPLE N:SAMPLETYPE
N:SPEC-ACT N:TYPEA N:TYPEB N:TYPEC N:TYPED R:ANALYSIS-REF R:ANALYSIS-TAG
R:AROUND R:BIBLIOGRAPHY R:DOC-ON R:ELT#1 R:ELT#2 R:GLASS R:GREATERVAL
R:LESSVAL R:N-DOC R:ONE R:ONEOF R:ONES R:PHASE R:PHASE#2 R:ROCKTYPE
R:SAMPLE-WITH R:SAMPLE-WITH-COMP R:SAMPLETYPE REFRULE REFRULE? S:ADD
S:ADDLINE S:ANALYZE S:AND S:BE-ABOUT S:BE-IN S:BE=IN2 S:BE-INTERESTED
S:CHANGE S:COMMON S:CONCERN S:DELETE S:DELETE# S:DISCOVER S:EDIT S:GIVE
S:GREAT S:I-NEED S:LIKE S:OLD S:OR S:PAPER-HAVE S:PERTAIN S:POSSESS
S:PRINTFILE S:REFER S:SAMPLE-BE-COMPOSED S:SAMPLE=CONTAIN
S:SAMPLE-HAVE#1 S:SAMPLE-HAVE#2 S:SEARCH S:SORT))
(DEFINEV
(N:AGE ((NP.N (EQU 1 AGE))
        (NP.PP (MEM 2 SAMPLE))
        ->
        (SSUNIONF (AGE (# 2 2 SSET))                    (* E.G. THE AGE OF EACH
                                                        TYPE/B ROCK)
                  )))
(N:AGE' ((NP.N (EQU 1 AGE))
         (NP.ADJ (MEM 1 CLOCK))
         (NP.PP (OR (MEM 2 SAMPLE)
                    (SAMPLEP (HEAD (# 2)))))
         ->
         (SSUNIONF (AGE (# 3 2 SSET)
                        (# 2 1))                        (* E.G. THE K-AR AGE OF
                                                        EACH TYPE/B ROCK)
))))
```

```
(N:ANALYSIS ((NP.N (MEM 1 ANALYSIS))
             (NOT (NP.ADJ (EQU 1 MODAL)))
             (OR (NP.PP (MEM 2 (SAMPLE ROCK)))
                 (NP.PP.PP (MEM 2 (SAMPLE ROCK)))
                 (NP.PP.PP.PP (MEM 2 (SAMPLE ROCK)))
                 (DEFAULT (2 NP (DET ALL)
                                (N SAMPLE)
                                (NU PL))))
             (OR (NP.PP (MEM 2 (PHASE MINERAL)))
                 (NP.PP.PP (MEM 2 (PHASE MINERAL)))
                 (NP.PP.PP.PP (MEM 2 (PHASE MINERAL)))
                 (NP.ADJ#2 (MEM 2 (PHASE MINERAL)))
                 (NP.PP.ADJ-N (AND (OR (EQU 2 FINE)
                                       (EQU 2 COARSE))
                                   (MEM 1 DUST)))
                 (DEFAULT (2 NP (NPR OVERALL))))
             (OR (NP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                 (NP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                 (NP.PP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                 (NP.ADJ#2 (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                 (DEFAULT (2 NP (DET EVERY)
                                (ADJ MAJOR)
                                (N ELEMENT)
                                (NU SG))))
             (OR (NP.ADJ (EQU 1 CHEMICAL))
                 (DEFAULT ( NPR NIL)))
             ->
             (SSUNIONF (DATALINE (WHQFILE (# 3 2 SSET))
                                 (# 3 2 SSET)
                                 (# 4 2)
                                 (# 5 2 SSET))
                                          (* E.G. ANALYSES OF
                                          KRYPTON IN TYPE/B ROCKS)
                       )))
(N:AVERAGE ((NP.N (MEM 1 (MEAN AVERAGE)))
            (NP.PP (MEM 2 (QUANTITY)))
            ->
              (QUOTE (# 2 2 AVERAGE)

            (* N:AVERAGE INVOKES THE RULE D:AVERAGE ON THE NP
            HANGING OFF THE NODE HEADED BY AVERAGE)


                       )))
(N:AVG-CONC? ((NP.N (MEM 1 (CONCENTRATION)))
              (NP.DET (AND (EQU 1 THE)
                           (EQU 2 SG)
                           (AVERAGE?)))
              ->
              (QUOTE (# 0 AVERAGE)
```

```
            (* N:AVG-CONC? CAUSES A MESSAGE TO BE PRINTED OUT TO
            THE USER, ASKING IF HE MEANT BY 'THE CONCENTRATION',
            'THE AVERAGE CONCENTRA- TION'' THE RULE MATCHES IF
            THE USER ANSWERS 'YES' AND INVOKES THE RULE
            D:AVERAGE.)


                        )))
(N:BASALT ((NP.N (MEM 1 (BASALT)))
            ->
            (QUOTE (SEQ TYPEAS))))
(N:CONCENTRATION ((NP.N (MEM 1 (CONCENTRATION)))
                  (NOT (NP.ADJ (EQU 1 MODAL)))
                  (OR (NP.PP (MEM 2 (SAMPLE ROCK)))
                      (NP.PP.PP (MEM 2 (SAMPLE ROCK)))
                      (NP.PP.PP.PP (MEM 2 (SAMPLE ROCK)))
                      (DEFAULT (2 NP (DET ALL)
                                     (N SAMPLE)
                                     (NU PL))))
                  (OR (NP.PP (MEM 2 (PHASE MINERAL)))
                      (NP.PP.PP (MEM 2 (PHASE MINERAL)))
                      (NP.PP.PP.PP (MEM 2 (PHASE MINERAL)))
                      (NP.ADJ#2 (MEM 2 (PHASE MINERAL)))
                      (NP.PP.ADJ-N (AND (OR (EQU 2 FINE)
                                            (EQU 2 COARSE))
                                        (MEM 1 DUST)))
                      (DEFAULT (2 NP (NPR OVERALL))))
                  (OR (NP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                      (NP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                      (NP.PP.PP.PP (MEM 2 (ELEMENT OXIDE ISOTOPE)))
                      (NP.ADJ#2 (MEM 2 (ELEMENT OXIDE ISOTOPE))))
                  ->
                  (SSUNIONF (DATALINE (WHQFILE (# 3 2 SSET))
                                      (# 3 2 SSET)
                                      (# 4 2)
                                      (# 5 2 SSET))
                                            (* E.G. THE
                                            CONCENTRATION OF KRYPTON
                                            IN TYPE/B SAMPLES)
                            )))
(N:CORETUBE ((OR (NP.N (MEM 1 (CORETUBE)))
                 (NP.ADJ-N (EQU 1 CORE)
                           L      BE
                 ->
                 (QUOTE (SEQ CORETUBES))))
(N:DOCUMENT ((NP.N (MEM 1 DOCUMENT))
             (NOT (NP.ADJ (MEM 1 LUNAR)))
             ->
             (QUOTE DOCUMENT)))
```

```
(N:DUST ((NP.N (OR (MEM 1 (SOIL))
                   (EQU 1 DUST)
                   (EQU 1 FINE)))
         ->
         (QUOTE (SEQ DUSTS))))
(N:ELT ((NP.N (EQU 1 ELEMENT))
        (NOT (NP.ADJ (EQU 1 MAJOR)))
        ->
        (QUOTE (SEQ ELEMENTS))))
(N:GABBRO ((NP.N (OR (MEM 1 (GABBRO))
                     (EQU 1 MICROGABBRO)))
           ->
           (QUOTE (SEQ TYPEBS))))
(N:HALOGEN ((NP.N (MEM 1 (HALOGEN)))
            ->
            (QUOTE (SEQ HALOGENS))))
(N:LINE# ((NP.N (EQU 1 NUMBER))
          (NP.ADJ (EQU 1 LINE))
          (NP.PP (AND (EQU 1 OF)
                      (MEM 2 ANALYSIS)))
          ->
          (QUOTE (SEQL (# 3 2)))))
(N:MAJOR-ELT ((NP.N (EQU 1 ELEMENT))
              (NP.ADJ (EQU 1 MAJOR))
              ->
              (QUOTE (SEQ MAJORELTS))))
(N:MAXIMUM ((NP.N (OR (EQU 1 MAXIMUM)
                      (AND (MEM 1 (BIG))
                           (SUPERLATIVE 1))))
            (NP.PP (MEM 2 (QUANTITY)))
            ->
            (QUOTE (# 2 2 MAXIMUM)

            (* N:MAXIMUM INVOKES THE RULE D:MAXIMUM ON THE NP
            HANGING OFF THE NODE HEADED BY 'MAXIMUM'.
            'MAXIMUM' CAN THEN ACT LIKE A FUNCTION, RATHER THAN
            A SET.)
                    )))
(N:MINERAL ((NP.N (EQU 1 MINERAL))
            ->
            (QUOTE (SEQ MINERALS))))
(N:MINIMUM ((NP.N (OR (EQU 1 MINIMUM)
                      (AND (MEM 1 (LITTLE))
                           (SUPERLATIVE 1))))
            (NP.PP (MEM 2 (QUANTITY)))
            ->
            (QUOTE (# 2 2 MINIMUM)                    (* SEE N:MAXIMUM)
                    )))
```

```
(N:MODAL-ANALYSIS ((NP.N (OR (MEM 1 ANALYSIS)
                             (EQU 1 MODE)))
                   (OR (NP.ADJ (EQU 1 MODAL))
                       (NP.N (EQU 1 MODE)))
                   (OR (NP.PP (MEM 2 (SAMPLE)))
                       (DEFAULT (2 NP (DET EVERY)
                                      (N SAMPLE)
                                      (NU SG))))
                   (OR (NP.PP (MEM 2 (PHASE MINERAL ELEMENT OXIDE
                                            ISOTOPE)))
                       (NP.ADJ#2 (MEM 2 (PHASE MINERAL ELEMENT OXIDE
                                               ISOTOPE))))
                   ->
                   (SSUNIONF (DATALINE (WHQFILE (# 3 2 SSET))
                                       (# 3 2      )
                                       OVERALL
                                       (# 4 2))
                                       (* E.G. MODAL ANALYSES
                                       OF OLIVINE IN TYPE/C
                                       ROCKS)
                             )))
(N:MODAL-CONC ((NP.N (MEM    (CONCENTRATION)))
               (OR (NP.PP (MEM 2 (SAMPLE)))
                   (NP.PP.PP (MEM 2 (SAMPLE)))
                   (DEFAULT (2 NP (DET EVERY)
                                  (N SAMPLE)
                                  (NU SG))))
               (OR (NP.PP (MEM 2 (PHASE MINERAL ELEMENT OXIDE ISOTOPE)))
                   (NP.ADJ#2 (MEM 2 (PHASE MINERAL ELEMENT OXIDE
                                           ISOTOPE))))
               ->
               (SSUNIONF (DATALINE (WHQFILE (# 2 2 SSET))
                                   (# 2 2 SSET)
                                   OVERALL
                                   (# 3 2)))))
(N:NUMBER ((NP.N (EQU 1 NUMBER))
           (NP.PP (EQU 1 OF))
           ->
           (QUOTE (# 2 2 NUMBER)

           (* N:NUMBER INVOKES D:NUMBER ON THE NP DEPENDENT ON
                 'NUMBER'. E.G. 'THE NUMBER OF TYPE/A SAMPLES' IS
           INTERPRETED AS          (SEQL (NUMBER X /
           (SEQ TYPEAS): T)))


                  )))
(N:OLDEST ((NP.N (AND (MEM    (OLD))
                      (SUPERLATIVE 1)))
           (NP.PP (MEM 2 (SAMPLE)))
           ->
           (QUOTE (# 2 2 OLDEST)
```

```
            (* N:OLDEST INVOKES THE RULE D:OLDEST ON THE NP
            DEPENDENT ON "OLDEST". LIKE N:MAXIMUM AND N:NUMBER,
            IT ALLOWS "OLDEST" TO ACT LIKE A FUNCTION, RATHER
            THAN A SET)


                    )))
(N:ONE ((NP.PRO (EQU 1 ONE))
        ->
        (PROGN (SETQ ANTEVAR (ANTECEDANT (QUOTE (# Ø IDENTITY)))

            (* N:ONE MATCHES THE ANAPHORIC PRONOUNS 'ONE' AND
            'ONES'.        THE INTERPRETATION DEPENDS ON THE CLASS OF
            THE ANTECENDANT.)

                  )
                 (MAPC (SCOPEVARS ANTEVAR)
                       (FUNCTION ANTEQUANT))
                 (NEWCLASS ANTEVAR))))
(N:ONEOF ((NP.PRO (EQU 1 ONE))
          (NP.PP (EQU 1 OF))
          ->
          (QUOTE (# 2 2 NRULES)

          (* N:ONEOF MATCHES THE PRONOUNS 'ONE' AND 'ONES'
          WHEN FOLLOWED BY      A PARTITIVE CONSTRUCTIN
          EXPRESSING THE SET. E.G. 'ONE OF THE TYPE/B ROCKS.'
          THE INTERPRETATION IS THE CLASS OF THE DEPENDENT
          NP.)


                    )))
(N:ONES ((NP.PRO (EQU 1 ONES))
         (NP.PP (AND (EQU   OF)
                     (NEQ (CAADR (# 2))
                          (QUOTE PRO))))
         ->
         (QUOTE (# 2 2 NRULES)

          (* N:ONES IS LIKE N:ONEOF, BUT IS USED TO MATCH
          SPECIAL PARTITIVES CONSTRUCTED BY THE PARSER.)


                   )))
(N:ONES-OF-PRO ((NP.PRO (EQU 1 ONES))
                (NP.PP (AND (EQU 1 OF)
                            (EQ (CAADR (# 2))
                                (QUOTE PRO))))
                ->
                (QUOTE (# 2 2)
```

```
            (* N:ONES-OF-PRO MATCHES NODES WITH PARTITIVE
            CONSTRUCTIONS, WHERE THE HEAD OF THE PARTITIVE IS
            ITSELF ANAPHORIC.)


                              )))
(N:PHASE ((NP.N (EQU 1 PHASE))
            ->
            (QUOTE (SEQ PHASES))))
(N:RARE/EARTH ((NP.N (EQU 1 RARE/EARTH))

                  (QUOTE (SEQ RARE/EARTHS))))
(N:RATIO ((NP.N (EQU 1 RATIO))
            (OR (NP.ADJ.N/N (AND (MEM 1 (ELEMENT ISOTOPE OXIDE))
                                 (MEM 2 (ELEMENT ISOTOPE OXIDE))))
                (NP.ADJ-ADJ (AND (MEM 1 (ELEMENT ISOTOPE OXIDE))
                                 (MEM 2 (ELEMENT ISOTOPE OXIDE)))))
           (OR (NP.PP (AND (OR (EQU 1 IN)
                               (EQU 1 FOR))
                           (MEM 2 (PHASE MINERAL))))
               (NP.PP.PP (AND (OR (EQU 1 IN)
                                  (EQU 1 FOR))
                              (MEM 2 (PHASE MINERAL))))
               (DEFAULT (2 NP (NPR OVERALL))))
           (OR (NP.PP (AND (OR (EQU 1 IN)
                               (EQU 1 OF)
                               (EQU 1 FOR))
                           (MEM 2 (SAMPLE ROCK))))
               (NP.PP.PP (AND (OR (EQU 1 IN)
                                  (EQU 1 OF)
                                  (EQU 1 FOR))
                              (MEM 2 (SAMPLE ROCK))))
               (DEFAULT (2 NP (DET EVERY)
                              (N SAMPLE)
                              (NU SG))))
           ->
           (APPLY (FUNCTION SSUNIONF)
                  (LIST (BUILDQ (RATIO (QUOTE #)
                                       (QUOTE #)
                                       (# 4 2 SSET)
                                       (# 3 2 SSET))
                                (TABFORM (# 2 1 HEAD))
                                (TABFORM (# 2 2 HEAD))))))

(* E.G. POTASSIUM / RUBIDIUM RATIOS INTERPRETS AS
(SSUNION X1 / (SEQ SAMPLES): T ;
(RATIO (QUOTE K2O) (QUOTE RB) X1
(QUOTE OVERALL))))
```

```
                    )))
(N:ROCK ((NP.N (OR (EQU 1 ROCK)
                   (EQU 1 VOLCANIC)))
         ->
         (QUOTE (SEQ VOLCANICS))))
(N:ROCKTYPE ((NP.N (MEM 1 (TYPE)))
             (OR (NP.PP (AND (EQU 1 OF)
                             (EQ (HEAD (# 2))
                                 (QUOTE ROCK))
                             (TAG (# 2)
                                  (QUOTE USED)
                                  T)))
                 (NP.ADJ (EQU 1 ROCK)))
             ->
             (QUOTE (SEQ ROCKTYPES))))
(N:SAMPLE ((OR (NP.N (EQU 1 SAMPLE))
               (NP.ADJ-N (AND (EQU 1 LUNAR)
                              (EQU 2 MATERIAL))))
           (OR (NP.ADJ (EQU 1 LUNAR))
               (DEFAULT (1 ADJ NIL)))
           ->
           (QUOTE (SEQ SAMPLES))))
(N:SAMPLETYPE ((NP.N (MEM 1 (TYPE)))
               (OR (NP.PP (AND (EQU 1 OF)
                               (EQ (HEAD (# 2))
                                   (QUOTE SAMPLE))
                               (TAG (# 2)
                                    (QUOTE USED)
                                    T)))
                   (NP.ADJ (EQU 1 SAMPLE)))
               ->
               (QUOTE (SEQ SAMPLETYPES))))
(N:SPEC-ACT ((NP.N (EQU 1 ACTIVITY))
             (NP.ADJ (EQU 1 SPECIFIC))
             (NP.PP (MEM 2 (ISOTOPE)))
             (OR (NP.PP (MEM 2 (SAMPLE)))
                 (NP.PP.PP (MEM 2 (SAMPLE)))
                 (DEFAULT (2 NP (DET EVERY)
                                (N SAMPLE)
                                (NU SG))))
             ->
             (QUOTE (DATALINE (WHQFILE (# 4 2))
                              (# 4 2)
                              OVERALL
                              (# 3 2)))          (* E.G. THE SPECIFIC
                                                 ACTIVITY OF CO56 IN
                                                 S10003)
         )))
```

```
(N:TYPEA ((NP.N (OR (EQU 1 PARTICLE)
                    (EQU 1 ROCK)
                    (EQU 1 SAMPLE)))
          (OR (NP.ADJ (EQU 1 TYPE/A))
              (NP.ADJ-ADJ (AND (EQU 1 HIGH)
                               (OR (EQU 2 ALKALI)
                                   (EQU 2 ALKALINE)
                                   (MEM 2 (RUBIDIUM)))))
              (NP.ADJ-ADJ-ADJ (AND (EQU 1 FINE)
                                   (EQU 2 GRAINED)
                                   (OR (EQU 3 IGNEOUS)
                                       (EQU 3 CRYSTALLINE)))))
          ->
          (QUOTE (SEQ TYPEAS))))
(N:TYPEB ((NP.N (OR (EQU 1 PARTICLE)
                    (EQU 1 SAMPLE)
                    (EQU 1 ROCK)))
          (OR (NP.ADJ (EQU 1 TYPE/B))
              (NP.ADJ-ADJ (AND (EQU 1 LOW)
                               (OR (EQU 2 ALKALI)
                                   (EQU 2 ALKALINE)
                                   (MEM 2 (RUBIDIUM)))))
              (NP.ADJ-ADJ-ADJ (AND (EQU 1 COARSE)
                                   (EQU 2 GRAINED)
                                   (OR (EQU 3 CRYSTALLINE)
                                       (EQU 3 IGNEOUS)))))
          ->
          (QUOTE (SEQ TYPEBS))))
(N:TYPEC ((OR (NP.N (OR (EQU 1 BRECCIA)
                        (EQU 1 MICROBRECCIA)))
              (NP.ADJ-N (AND (OR (EQU 2 PARTICLE)
                                 (EQU 2 SAMPLE)
                                 (EQU 2 ROCK))
                             (EQU 1 TYPE/C))))
          ->
          (QUOTE (SEQ TYPECS))))
(N:TYPED ((OR (NP.N (OR (EQU 1 SOIL)
                        (EQU 1 DUST)
                        (EQU 1 FINE)))
              (NP.ADJ-N (AND (OR (EQU 1 PARTICLE)
                                 (EQU 1 SAMPLE)
                                 (EQU 1 ROCK))
                             (EQU 1 TYPE/D))))
          ->
          (QUOTE (SEQ DUSTS))))
```

```
(R:ANALYSIS-REF ((NP.N (MEM 1 (ANALYSIS)))
                 (OR
                   (NP.PP
                     (DOCP (CADR (ASSOC (QUOTE NPR)
                                        (CDR (CONSTITUENTS
                                               (# 2))))))))
                   (NP.PP.PP
                     (DOCP (CADR (ASSOC (QUOTE NPR)
                                        (CDR (CONSTITUENTS
                                               (# 2))))))))
                   (NP.ADJ#2 (DOCP (CADR (# 2)))))
                 ->
                 (QUOTE (REF* X (# 2 2)))))
(R:ANALYSIS-TAG ((NP.N (MEM 1 (ANALYSIS)))
                 (OR (NP.PP.ADJ-NPR (AND (EQU 1 TAG)
                                         (NUMBERP (CADR (# 2))))))
                     (NP.PP.PP.ADJ-NPR
                       (AND (EQU 1 TAG)
                            (NUMBERP (CADR (# 2))))))
                 ->
                 (QUOTE (TAG* X (# 2 2)))))
(R:AROUND ((NP.N (MEM 1 ANALYSIS))
           (NP.PP.COMP (AND (OR (EQU 3 AROUND)
                                (EQU 3 APPROXIMATELY))
                            (MEM 2 (ELEMENT ISOTOPE OXIDE))))
           ->
           (BUILDQ (AROUNDVAL X #)
                   (LIST (QUOTE QUOTE)
                         (CONS (# 2 4 INTEGER)
                               (# 2 5 UNIT)))

           (* R:AROUND MATCHES NP'S LIKE 'ANALYSES WITH AROUND
           7 PPM          TITANIUM'. THE INTERPRETATION OF THE
           RESTRICTION THAT R:AROUND PRODUCES    IS
           (AROUND X (7 . PPM)). WHERE 'X' REFERS TO THE
           ANALYSES.)
                      )))
(R:BIBLIOGRAPHY ((NP.N (EQU 1 BIBLIOGRAPHY))
                 (NP.PP (OR (EQU 1 ON)
                            (EQU 1 ABOUT)))
                 ->
                 (QUOTE (ABOUT X (# 2 2 TOPIC)))))
(R:DOC-ON ((NP.N (MEM 1 DOCUMENT))
           (NP.PP T)
           ->
           (QUOTE (ABOUT X (# 2 2 TOPIC)))))
```

```
(R:ELT#1 ((NP.N (EQU 1 ELEMENT))
          (NP.ADJ (AND (MEM 1 (ELEMENT OXIDE ISOTOPE))
                       (NOT (MEM 1 (SET)))))
          ->
          (QUOTE (EQUAL X (# 2 1))                    (* E.G. THE URANIUM
                                                      ELEMENT)
                 )))
(R:ELT#2 ((NP.N (EQU 1 ELEMENT))
          (NP.ADJ.NP (AND (MEM 1 (ELEMENT OXIDE ISOTOPE))
                          (MEM 1 (SET))))
          ->
          (QUOTE (MEMB X (# 2 1 NRULES))              (* E.G. A RARE-EARTH
                                                      ELEMENT)
                 )))
(R:GLASS ((NP.N (MEM 1 (SAMPLE FRAGMENT PARTICLE)))
          (NP.ADJ (EQU 1 GLASS))
          ->
          (QUOTE (CONTAIN X (# 2 1)))))
(R:GREATERVAL ((NP.N (MEM 1 ANALYSIS))
               (NP.PP.COMP (AND (OR (EQU 3 GREATERTHAN)
                                    (EQU 3 MORETHAN))
                                (MEM 2 (ELEMENT ISOTOPE OXIDE))))
               ->
               (BUILDQ (GREATER X #)
                       (LIST (QUOTE QUOTE)
                             (CONS (# 2 4 INTEGER)
                                   (# 2 5 UNIT)))

          (* R:GREATERVAL IS THE ANALOGUE OF R:AROUND FOR
          PHRASES WITH 'GREATER THAN' OR 'MORETHAN'.)


                       )))
(R:LESSVAL ((NP.N (MEM 1 ANALYSIS))
            (NP.PP.COMP (AND (OR (EQU 3 LESSTHAN)
                                 (EQU 3 FEWERTHAN))
                             (MEM 2 (ELEMENT OXIDE ISOTOPE))))
            ->
            (BUILDQ (LESSVAL X #)
                    (LIST (QUOTE QUOTE)
                          (CONS (# 2 4 INTEGER)
                                (# 2 5 UNIT)))
                                                    (* R:LESSVAL IS THE
                                                    ANALOGUE OF R:AROUND FOR
                                                    PHRASES WITH 'LESS
                                                    THAN'.)
                    )))
(R:N-DOC ((NP.N (MEM 1 DOCUMENT))
          (NP.ADJ.NP T)
          ->
          (QUOTE (ABOUT X (# 2 1 TOPIC)))))
```

```
(R:ONE ((NP.PRO (EQU 1 ONE))
        ->
        (PROG1 (NEWPX ANTEVAR

          (* R:ONE MATCHES THE ANAPHORIC PRONOUN 'ONE'.
          ITS  INTERPRETATION IS THE SET OF RESTRICTIONS ON
          THE ANTEAEDANT OF 'ONE' THAT WERE NOT PRODUCED FROM
          THE VERB PHRASE. SEE THE FUNCTION DESCRIPTION          OF
          NEWPX FOR FURTHER DETAIL.)

))))
(R:ONEOF ((NP.PRO (EQU 1 ONE))
          (NP.PP (AND (EQU 1 OF)
                      (NOT (EQ (CAADR (# 2))
                               (QUOTE PRO
          (* R:ONEOF MATCHES THE PRONOUN 'ONE' WHEN IT
          DOMINATES A PARTITIVE CONSTRUCTION WHOSE HEAD IS NOT
          A PRONOUN. E.G. 'WHICH ONE OF THE TYPE/A ROCKS'.
          R:ONEOF CALLS FOR THE CLASS RESTRICTIONS ON THE HEAD
          OF THE PARTITIVE NP.)

)))))
          ->
          (QUOTE (# 2 2 RRULES))))
(R:ONES ((NP.PRO (EQU 1 ONES))
         (NP.PP (AND (EQU  OF)
                     (NEQ (CAADR (# 2))
                          (QUOTE PRO

          (* R:ONES IS THE ANALOGUE OF R:ONEOF USED FOR THE
                PRONOUN 'ONES' INSERTED BY THE PARSER FOR DEEP
          STRUCTURE PARTITIVES.          E.G.
          'ALL THE BOYS' IS PARSED AS 'ALL ONES OF THE BOYS'.)

))))
          ->
          (QUOTE (# 2 2 RRULES))))
(R:PHASE ((NP.N (EQU 1 PHASE))
          (NP.ADJ (MEM 1 (PHASE MINERAL)))
          ->
          (QUOTE (EQUAL X (# 2 1)))))
(R:PHASE#2 ((NP.N (EQU 1 PHASE))
            (NP.PP (OR (MEM 2 (SAMPLE))
                       (SAMPLEP (HEAD (# 2)))))
            ->
            (QUOTE (CONTAIN (# 2 2 SSET)
                            X))))
```

```
(R:ROCKTYPE ((NP.ADJ-N (AND (EQU 1 ROCK)
                            (MEM 2 (TYPE))))
             (NP.PP (MEM 2 (SAMPLE)))
             ->
             (QUOTE (MEMBER* (# 2 2)
                             X)                    (* E.G. THE ROCK TYPE OF
                                                   SAMPLE 10003)
                    )))
(R:SAMPLE-WITH ((NP.N (MEM 1 SAMPLE))
                (NOT (NP.PP.COMP T))
                (NP.PP (AND (EQU 1 WITH)
                            (MEM 2 (ELEMENT ISOTOPE OXIDE))))
                (OR (NP.PP (AND (EQU 1 IN)
                                (MEM 2 (MINERAL PHASE))))
                    (NP.PP.PP (AND (EQU 1 IN)
                                   (MEM 2 (MINERAL PHASE))))
                    (DEFAULT (2 NPR NIL)))
               ->
               (QUOTE (CONTAIN X (# 3 2)
                                 (# 4 2))          (* E.G. SAMPLES WITH
                                                   BERYLLIUM)
                      )))
(R:SAMPLE-WITH-COMP ((NP.N (MEM 1 (SAMPLE)))
                     (NP.PP.COMP (AND (EQU 1 WITH)
                                      (MEM 2 (ELEMENT ISOTOPE OXIDE))))
                     (OR (NP.PP (MEM 2 (PHASE MINERAL)))
                         (NP.PP.PP (MEM 2 (PHASE MINERAL)))
                         (DEFAULT (2 NPR NIL)))
                     ->
                     (QUOTE (CONTAIN' X (# 2 2)
                                        (# 3 2)
                                        (# 2 6 TERM))
                                                   (* E.G. SAMPLES WITH
                                                   MORE THAN 6 PPM
                                                   BERYLLIUM IN PLAG)
                            )))
(R:SAMPLETYPE ((NP.ADJ-N (AND (EQU 1 SAMPLE)
                              (MEM 2 (TYPE))))
               (NP.PP (MEM 2 (SAMPLE)))
               ->
               (QUOTE (MEMBER* (# 2 2)
                               X))))
(REFRULE ((T T)
          ->
          (QUANT (FOR EVERY X / DOCUMENT : (ABOUT X (# Ø TOPIC))
                      ;
                      (PRINTOUT X)))))
(REFRULE? ((T (INTERP P))
           ->
           (QUOTE (# Ø))))
```

```
(S:ADD ((S.V (MEM 1 (ADD)))
        (S.OBJ.NPR7 (AND (SAMPLEP (CADR (# 1)))
                         (MEM 2 (PHASE MINERAL))
                         (MEM 3 (ELEMENT OXIDE ISOTOPE))
                         (NUMBERP (CADR (# 4)))
                         (MEM 5 (UNIT))
                         (DOCP (CADR (# 6)))
                         (NUMBERP (CADR (# 7)))))
        (S.PP (MEM 2 (FILE)))
        ->
        (PRED (APPLY (FUNCTION PRENEWLINE)
                     (LIST (# 3 2)
                           (# 2 1)
                           (# 2 2)
                           (# 2 3)
                           (# 2 4)
                           (# 2 5)
                           (# 2 6)
                           (# 2 7)))

            (* E.G. ADD THE LINE (S10003, OVERALL, BE, 6, PPM,
            D70-154, 0) TO APOLLO11)


                  )))
(S:ADDLINE ((S.IMP T)
            (S.V (OR (EQU  ADDLINE)
                     (EQU  ADD)))
            (S.OBJ.NPR T)
            (OR (S.PP (MEM 2 (FILE)))
                (S.OBJ.PP (MEM 2 (FILE))))
            ->
            (BUILDQ (APPLY (FUNCTION PRENEWLINE)
                           (QUOTE #))
                    (CONS (QUOTE (# 4 2))
                          (QUOTE (# 3 1 TERM)))
                    (E.G. ADD
(↑ S10003 OVERALL BE 6 PPM D70-154 0)
TO APOLLO11 THE INTERPRETATION IS
(APPLY (FUNCTION PRENEWLINE)
       (QUOTE
              (APOLLO11 S10003 OVERALL BE 6
                        PPM D70-154 0))))))))
```

```
(S:ANALYZE ((S.NP-V (AND (MEM 2 (ANALYZE))
                         (COND
                           ((NOT (MEMB (CADR (# 1))
                                       (QUOTE (SOMEONE SOMETHING ANYONE)
                                              )))
                            (PRIN1 (QUOTE I PRESENTLY IGNORE SUBJECTS
                                          OF THE VERB)
                                   T)
                            (PRINT HEAD T))
                           (T T))))
                    (S.OBJ (MEM 1 (SAMPLE)))
                    (S.PP (EQU 1 FOR))
                    ->
                    (SSUNIONF (DATALINE (WHQFILE (# 2 1 SSET))
                                        (# 2 1 SSET)
                                        OVERALL
                                        (# 3 2 SSET))

                    (* E.G. HAS S10003 BEEN ANALYZED FOR KRYPTON? THE
                    INTERPRETATION IS (TEST (DATALINE
                    (WHQFILE S10003) S10003 OVERALL        KR)))


                                  )))
 (S:AND (AND (S.AND T)
             ->
             (PRED (# 1 1))))
 (S:BE-ABOUT ((S.NP (MEM 1 DOCUMENT))
              (S.V (EQU 1 BE))
              (S.PP (OR (EQU 1 ON)
                        (EQU 1 ABOUT)
                        (EQU 1 TO)
                        (EQU 1 FOR)))
              ->
              (PRED (PRINTOUT (# 3 2 REFS)))))
 (S:BE-IN ((S.NP-V (AND (MEM 2 (BE EXIST OCCUR))
                        (MEM 1 (ELEMENT OXIDE ISOTOPE PHASE MINERAL))))
           (OR (S.PP (AND (EQU 1 IN)
                          (MEM 2 (SAMPLE))))
               (S.PP.PP (AND (EQU 1 IN)
                             (MEM 2 (SAMPLE)))))
           (OR (S.PP (AND (EQU 1 IN)
                          (MEM 2 (PHASE MINERAL))))
               (DEFAULT (2 NPR NIL)))
           ->
           (PRED (CONTAIN (# 2 2 SET?)
                          (# 1 1)
                          (# 3 2))))))
```

```
(S:BE-IN2 ((S.NP.COMP (MEM 1 (ELEMENT OXIDE ISOTOPE PHASE MINERAL)))
           (S.V (MEM 1 (BE EXIST OCCUR)))
           (OR (S.PP (AND (EQU 1 IN)
                          (OR (MEM 2 (SAMPLE))
                              (SAMPLEP (HEAD (# 2))))))
               (S.PP.PP (OR (MEM 2 (SAMPLE))
                            (SAMPLEP (HEAD (# 2))))))
           (OR (S.PP (AND (EQU 1 IN)
                          (MEM 2 (PHASE MINERAL))))
               (DEFAULT (2 NPR NIL)))
           ->
           (PRED (CONTAIN' (# 3 2)
                           (# 1 1)
                           (# 4 2)
                           (# 1 2 TERM))
                 (* E.G. IS THERE MORE THAN 7 PPM KRYPTON IN S10003?
                 THE INTERPRETATION IS (TEST
                 (CONTAIN' (QUOTE S10003) (QUOTE KR)
                 (MORETHAN 7 PPM))))


                      )))
(S:BE-INTERESTED ((S.V (EQU 1 BE))
                  (S.ADJ (EQU 1 INTERESTED))
                  (S.PP (EQU 1 IN))
                  ->
                  (PRED (PRINTOUT (# 3 2 REFS)))))
(S:CHANGE ((S.V (EQU 1 CHANGE))
           (S.IMP T)
           (S.OBJ (MEM 1 FIELDNAME))
           (S.OBJ.PP (AND (EQU 1 OF)
                          (MEM 2 ANALYSIS)))
           (OR (S.OBJ.PP (EQU 1 TO))
               (S.OBJ.PP.PP (EQU 1 TO)))
           ->
           (PRED (CHANGE1LINE (# 4 2)
                              (QUOTE (# 3 1 HEAD))
                              (QUOTE (# 5 2 HEAD))))))
(S:COMMON ((S.NP-V (AND (MEM 1 (PHASE MINERAL ELEMENT ISOTOPE OXIDE))
                        (EQU 2 COMMON)))
           (S.PP (EQU 1 TO))
           ->
           (PRED (CONTAIN (# 2 2)
                          (# 1 1))
                 (E.G. WHAT PHASES ARE COMMON TO ALL SAMPLES?))))
(S:CONCERN ((S.NP (MEM 1 DOCUMENT))
            (S.V (MEM 1 CONCERN))
            (S.OBJ T)
            ->
            (PRED (PRINTOUT (# 3 1 REFS?)))))
```

```
(S:DELETE ((S.IMP T)
           (S.V (EQU 1 DELETE))
           (S.OBJ (MEM 1 ANALYSIS))
           ->
           (PRED (DELETE1LINE (# 3 1)))))
(S:DELETE# ((S.IMP T)
            (S.V (EQU 1 DELETE))
            (S.OBJ.NPR (LINEP (# 1)))
            (OR (S.PP (MEM 2 (FILE)))
                (S.OBJ.PP (MEM 2 (FILE))))
            ->
            (QUOTE (DELETE# (# 4 2)
                            (# 3 1)))))
(S:DISCOVER ((NOT (S.IMP T))
             (S.V (MEM 1 (DISCOVER CONTAIN)))
             (S.OBJ (MEM 1 (ELEMENT OXIDE ISOTOPE MINERAL PHASE)))
             (OR (S.PP (MEM 2 (MINERAL PHASE)))
                 (DEFAULT (2 NPR NIL)))
             (OR (S.PP (MEM 2 (SAMPLE)))
                 (S.PP.PP (MEM 2 (SAMPLE)))
                 (DEFAULT (2 NP (DET ANY)
                                (N SAMPLE)
                                (NU PL))))
             (S.NP (COND
                     ((NOT (EQU 1 SOMETHING))
                       (PRIN1 (QUOTE I PRESENTLY IGNORE SUBJECTS OF THE
                                     VERB)
                              T)
                       (PRINT HEAD T))
                     (T T)))
             ->
             (PRED (CONTAIN (# 5 2 SET?)
                            (# 3 1)
                            (# 4 2)))))
(S:EDIT ((S.V (EQU 1 EDIT))
         (S.OBJ.NPR (LINEP (# 1)))
         (OR (S.PP (MEM 2 (FILE)))
             (S.OBJ.PP (MEM 2 (FILE))))
         ->
         (PRED (EDITLINE (# 3 2)
                         (# 2 1)))))
(S:GIVE ((S.V (MEM 1 GIVE))
         (S.OBJ T)
         (OR (S.IMP T)
             (S.O-MODAL T))
         ->
         (PRED (PRINTOUT (# 2 1 REFS?)))))
(S:GREAT ((S.NP-V (MEM 2 GREAT))
          ->
          (PRED (# 1 1))))
```

```
(S:I-NEED ((S.NP (EQU 1 I SG))
           (S.V (OR (EQU 1 NEED)
                    (EQU 1 WANT)))
           (S.OBJ T)
           ->
           (PRED (PRINTOUT (# 3 1 REFS?)))))
(S:LIKE ((S.V (MEM 1 LIKE))
         (S.OBJ T)
         ->
         (PRED (PRINTOUT (# 2 1 REFS?)))))
(S:OLD ((S.NP-V (MEM 2 OLD))
        ->
        (PRED (FOR EVERY X0 / (AGE (# 1 1))
                   : T ; (PRINTOUT X0)))))
(S:OR (OR (S.OR T)
          ->
          (PRED (# 1 1))))
(S:PAPER-HAVE ((S.NP (MEM   DOCUMENT))
               (S.V (EQU 1 HAVE))
               (S.OBJ T)
               ->
               (PRED (PRINTOUT (# 3 1 REFS?)))))
(S:PERTAIN ((S.NP (MEM 1 DOCUMENT))
            (S.V (MEM 1 PERTAIN))
            (S.PP (OR (EQU   WITH)
                      (EQU   TO)))
            ->
            (PRED (ABOUT (# 1 1)
                         (# 3 2 TOPIC)))))
(S:POSSESS ((S.NP-V (AND (MEM 1 (WE YOU THEY))
                         (MEM 2 (POSSESS))))
            (S.OBJ T)
            ->
            (PRED (EXIST (# 2 1)))))
(S:PRINTFILE ((OR (S.IMP T)
                  (S.Q-MODAL T))
              (S.V (MEM 1 (GIVE)))
              (S.OBJ (MEM   (FILE)))
              ->
              (PRED (PRINTFILE (# 3 1)))))
(S:REFER ((S.NP (MEM 1 DOCUMENT))
          (S.V (EQU 1 REFER))
          (S.PP (EQU 1 TO))
          ->
          (PRED (ABOUT (#   1)
                       (# 3 2 TOPIC)))))
(S:SAMPLE-BE-COMPOSED ((S.NP-V (AND (MEM 1 (SAMPLE))
                                    (EQU 2 BE)))
                       (S.OBJ.COMP (MEM 1 (PHASE MINERAL)))
                       ->
                       (PRED (CONTAIN' (# 1 1)
                                       (# 2 1)
                                       (# 2 2 TERM)))))
```

```
(S:SAMPLE-CONTAIN (AND (S.NP (MEM 1 (SAMPLE)))
                       (S.V (OR (EQU 1 HAVE)
                                (EQU 1 CONTAIN)))
                       (OR (S.OBJ (MEM 1 (ELEMENT OXIDE ISOTOPE MINERAL)
                                  ))
                           (S.OBJ.AND (MEM 1 (ELEMENT OXIDE ISOTOPE
                                                      MINERAL))))
                       (OR (S.OBJ.PP (MEM 2 (MINERAL PHASE)))
                           (S.PP (MEM 2 (MINERAL PHASE)))
                           (DEFAULT (2 NPR NIL)))
                       ->
                       (PRED (CONTAIN (# 1 1 SET?)
                                      (# 3 1)
                                      (# 4 2)))))
(S:SAMPLE-HAVE#1 ((S.NP-V (AND (MEM 1 (SAMPLE))
                               (MEM 2 (HAVE CONTAIN))))
                  (S.OBJ.COMP (MEM 1 (ELEMENT OXIDE ISOTOPE)))
                  (S.PP (MEM 2 (PHASE MINERAL)))
                  ->
                  (PRED (CONTAIN' (# 1 1)
                                  (# 2 1)
                                  (# 3 2)
                                  (# 2 2 TERM)))))
(S:SAMPLE-HAVE#2 ((S.NP-V (AND (MEM 1 (SAMPLE))
                               (MEM 2 (HAVE CONTAIN))))
                  (S.OBJ.COMP (MEM 1 (ELEMENT OXIDE ISOTOPE PHASE
                                              MINERAL)))
                  ->
                  (PRED (CONTAIN' (# 1 1)
                                  (# 2 1)
                                  (# 2 2 TERM)))))
(S:SEARCH ((S.V (MEM 1 SEARCH))
           (S.PP (OR (EQU  FOR)
                     (EQU  IN)
                     (EQU  ON)))
           ->
           (PRED (PRINTOUT (# 2 2 REFS?)))))
(S:SORT ((S.V (EQU 1 SORT))
         (S.IMP T)
         ->
         (PRED (SORTNEW))))
)
(LISPXPRINT (QUOTE (V: TREEFRAGS)) T)
(RPAQQ TREEFRAGS (ADJ.NP NP.ADJ NP.ADJ#2 NP.ADJ-ADJ NP.ADJ-ADJ-ADJ
NP.ADJ-N NP.ADJ.COMP NP.ADJ.NP NP.ADJ.NPR NP.ADJ.N/N NP.ADVP NP.AND
NP.AND2 NP.COMPL NP.DET NP.DET.ART NP.DET.COMP NP.DET.COMP-UNIT
NP.DET.INTEGER NP.DET.MANY NP.DET.POSTART NP.N NP.NEG NP.NOM NP.NPR
NP.NR.NP NP.NR.S NP.OR NP.OR2 NP.PP NP.PP.ADJ-N NP.PP.ADJ-NPR NP.PP.AND
NP.PP.AND.PP NP.PP.COMP NP.PP.COMPL NP.PP.NPR NP.PP.PP NP.PP.PP#4
NP.PP.PP.ADJ-NPR NP.PP.PP.COMP-UNIT NP.PP.PP.PP NP.PP.PP.PP.ADJ-NPR
```

```
NP.PRO NP.OREL NP.REL S.ADJ S.AND S.AND.NPR S.COMP S.COMP-N S.COMPL
S.COMPL-THAN S.COMPL.OBJ S.DCL S.DCL-S S.IMP S.IMP-S S.NEG S.NEG-S
S.NP S.NP.COMP S.NPR S.NPQ S.NPU S.NP-V S.OBJ S.OBJ.ADJ#2 S.OBJ.AND
S.OBJ.COMP S.OBJ.NPR S.OBJ.NPR7 S.OBJ.OR S.OBJ.PP S.OBJ.PP.PP
S.OBJ.PP.PP.PP S.OBJ.PP.PP.PP.PP S.OBJ.REL S.OR S.PP S.PP#1 S.PP.AND
S.PP.AND.PP S.PP.PP S.Q S.Q-MODAL S.Q-NEG S.Q-S S.V S.VP))
(DEFINEV
(ADJ.NP (ADJ ((NP NIL 1))))
(NP.ADJ (NP ((ADJ NIL 1))))
(NP.ADJ#2 (NP ((ADJ NIL 2))))
(NP.ADJ-ADJ (NP ((ADJ NIL )
                 (ADJ NIL 2))))
(NP.ADJ-ADJ-ADJ (NP ((ADJ NIL 1)
                     (ADJ NIL 2)
                     (ADJ NIL 3))))
(NP.ADJ-N (NP ((ADJ NIL 1)
               (N NIL 2))))
(NP.ADJ.COMP (NP ((ADJ ((COMPARATIVE))
                       1))))
(NP.ADJ.NP (NP ((ADJ ((NP NIL 1))))))
(NP.ADJ.NPR (NP ((ADJ ((NPR NIL 1))))))
(NP.ADJ.N/N (NP ((ADJ ((NP ((N ((N NIL 1)
                                /
                                (N NIL 2)))))))))))
(NP.ADVP (NP ((ADVP ((ADV NIL 1)
                     (NP NIL 2))))))
(NP.AND (NP (AND (NP NIL 1))))
(NP.AND2 (NP (AND (NP NIL )
                  (NP NIL 2))))
    C            C            ±
(NP.DET (NP ((DET NIL 1)
             (NU NIL 2))))
(NP.DET.ART (NP ((DET ((ART NIL 1))))))
(NP.DET.COMP (NP ((DET ((POSTART ((COMP ((ADV NIL 1)
                                         (NP ((INTEGER NIL 2)))))
                                  MANY))))
                 (NU NIL 3))))
(NP.DET.COMP-UNIT (NP ((DET ((POSTART ((COMP ((ADV NIL 1)
                                              (NP ((INTEGER NIL 2)
                                                   (UNIT NIL 3)))))
                                       MUCH))))))
(NP.DET.INTEGER (NP ((DET ((POSTART ((NP ((INTEGER NIL 1)))
                                     MANY))))))
(NP.DET.MANY (NP ((DET ((NP ((INTEGER NIL 1)))
                        MANY))
                 (NU NIL 2))))
(NP.DET.POSTART (NP ((DET ((POSTART NIL 1))))))
(NP.N (NP ((N NIL 1))))
(NP.NEG (NP (NEG (NP ((DET NIL 2))
                      1))))
```

```
(NP.NOM (NP (NOM (S NIL 1))))
(NP.NPR (NP ((NPR NIL 1))))
(NP.NR.NP (NP ((NR ((NP NIL 1))))))
(NP.NR.S (NP ((NR ((S NIL  ))))))
(NP.OR (NP (OR (NP NIL 1))))
(NP.OR2 (NP (OR (NP NIL 1)
                (NP NIL 2))))
(NP.PP (NP ((N NIL 4)
            (PP ((PREP NIL  )
                 (NP NIL 2))
                3))))
(NP.PP.ADJ-N (NP ((PP ((ADJ NIL 2)
                       (N NIL 1))))))
(NP.PP.ADJ-NPR (NP ((PP ((NP ((ADJ NIL 1)
                              (NPR NIL 2)))))))))
(NP.PP.AND (NP ((PP ((PREP NIL 1)
                     (NP (AND (NP NIL 2)
                              (NP NIL 3)))))))))
(NP.PP.AND.PP (NP ((PP (AND (PP ((PREP NIL 1)
                                 (NP NIL 2)))))))))
(NP.PP.COMP (NP
              ((PP
                 ((PREP NIL 1)
                  (NP ((DET ((POSTART ((COMP ((ADV NIL 3)
                                              (NP ((INTEGER NIL 4)
                                                   (UNIT NIL 5))))
                                             6)
                                        MUCH))))))
                   2))))))
(NP.PP.COMPL (NP ((PP ((NP ((COMPL ((S NIL 1)))))))))))
(NP.PP.NPR (NP ((PP ((PREP NIL 1)
                     (NP ((NPR NIL 2)))))))))
(NP.PP.PP (NP ((PP ((NP ((PP ((PREP NIL 1)
                              (NP NIL 2)))))))))))
(NP.PP.PP#4 (NP ((PP ((PREP NIL 3)
                      (NP ((PP ((PREP NIL 1)
                                (NP NIL 2))))
                           4))))))
(NP.PP.PP.ADJ-NPR (NP ((PP ((NP ((PP ((NP ((ADJ NIL 1)
                                            (NPR NIL 2)))))))))))))
(NP.PP.PP.COMP-UNIT (NP
                      ((PP
                         ((NP
                            ((PP
                               ((NP
                                  ((DET
                                     ((POSTART
                                        ((COMP ((ADV NIL 1)
                                                (NP ((INTEGER NIL 2)
                                                     (UNIT NIL 3)))))
                                         MUCH)))))))))))))))
```

```
(NP.PP.PP.PP (NP
                ((PP ((NP ((PP ((NP ((PP ((PREP NIL 1)
                                          (NP NIL 2))))))))))))))
(NP.PP.PP.PP.ADJ-NPR (NP
                       ((PP
                          ((NP
                             ((PP
                                ((NP ((PP ((NP ((ADJ NIL 1)
                                                (NPR NIL 2))))))))))))))
                       ))
(NP.PRO (NP ((PRO NIL 1))))
(NP.QREL (NP ((S (QREL)
                 1))))
(NP.REL (NP ((S (REL)
                1))))
(S.ADJ (S ((VP ((ADJ NIL 1))))))
(S.AND (S (AND (S NIL 1))))
(S.AND.NPR (S ((NP (AND (NP ((NPR NIL 1))))))))
(S.COMP (S
          ((VP ((NP ((DET ((POSTART ((COMP ((ADV NIL 1)
                                            (NP ((INTEGER NIL 2)))))
                                     MANY)))))))))))
(S.COMP-N (S
            ((VP
               ((NP ((DET ((POSTART ((COMP ((ADV NIL 1)
                                            (NP ((INTEGER NIL 2)))))
                                     MANY))))
                    (N NIL 3)))))))
(S.COMPL (S ((VP ((COMPL ((NP (NOM (S NIL 1)))))))))))
(S.COMPL-THAN (S ((VP ((COMPL ((NP (THAN (S NIL 1)))))))))))
(S.COMPL.OBJ (S
               ((VP
                  ((COMPL ((NP (NOM (S ((VP ((NP NIL 1)))))))))))))))
(S.DCL (S (DCL)))
(S.DCL-S (S (DCL (S NIL 1))))
(S.IMP (S (IMP)))
(S.IMP-S (S (IMP (S NIL 1))))
(S.NEG (S (NEG)))
(S.NEG-S (S (NEG (S NIL 1))))
(S.NP (S ((NP NIL 1))))
(S.NP.COMP (S ((NP ((DET ((POSTART ((COMP ((ADV NIL))
                                          2)
                                   MUCH)))))
                    1))))
(S.NPR (S ((NP ((NPR NIL 1))))))
(S.NPQ (S (NPQ (NP NIL 1))))
(S.NPU (S (NPU (NP NIL 1))))
(S.NP-V (S ((NP NIL 1)
            (VP ((V NIL 2))))))
(S.OBJ (S ((VP ((NP ((NU NIL 2))
                    1))))))
```

```
(S.OBJ.ADJ#2 (S ((VP ((NP ((ADJ NIL 2)))))))
(S.OBJ.AND (S ((VP ((NP (AND (NP NIL 1)))))))
(S.OBJ.COMP (S ((VP ((NP ((DET ((POSTART ((COMP ((ADV NIL))
                                                 2)
                                               MUCH)))))
                                 1))))))
(S.OBJ.NPR (S ((VP ((NP ((NPR NIL 1)))))))
(S.OBJ.NPR7 (S
              ((VP
                 ((NP
                    ((NR ((S ((NP (OR (NP ((NPR NIL 1)))
                                      (NP NIL 2)
                                      (NP NIL 3)
                                      (NP ((NPR NIL 4)))
                                      (NP NIL 5)
                                      (NP ((NPR NIL 6)))
                                      (NP ((NPR NIL 7))))))))))))))
(S.OBJ.OR (S ((VP ((NP (OR (NP NIL 1))))))))
(S.OBJ.PP (S ((VP ((NP ((PP ((PREP NIL 1)
                             (NP NIL 2)))))))))
(S.OBJ.PP.PP (S ((VP ((NP ((PP ((NP ((PP ((PREP NIL 1)
                                          (NP NIL 2))))))))))))))
(S.OBJ.PP.PP.PP (S
                  ((VP
                     ((NP
                        ((PP
                           ((NP ((PP ((NP ((PP ((PREP NIL 1)
                                               (NP NIL 2)))))))))))))))
                   )))
(S.OBJ.PP.PP.PP.PP (S
                     ((VP
                        ((NP
                           ((PP
                              ((NP
                                 ((PP
                                    ((NP
                                       ((PP
                                          ((NP ((PP ((PREP NIL 1)
                                                     (NP NIL 2)))))))))))))))))))))
(S.OBJ.REL (S ((VP ((NP ((S NIL 1)))))))
(S.OR (S (OR (S NIL 1))))
(S.PP (S ((VP ((PP ((PREP NIL 1)
                    (NP NIL 2)))))))
(S.PP#1 (S ((VP ((PP ((NP NIL 1)))))))
(S.PP.AND (S ((VP ((PP ((PREP NIL 1)
                        (NP (AND (NP NIL 2)
                                 (NP NIL 3)))))))))
(S.PP.AND.PP (S ((VP ((PP (AND (PP ((PREP NIL 1)
                                    (NP NIL 2)))))))))
```

```
(S.PP.PP (S ((VP ((PP ((NP ((PP ((PREP NIL 1)
                                 (NP NIL 2)))))))))))))
(S.Q (S (Q (NP NIL)
           (VP NIL))))
(S.Q-MODAL (S (Q (AUX ((MODAL NIL))))))
(S.Q-NEG (S (Q NEG (NP NIL)
                   (VP NIL))))
(S.Q-S (S (Q (S NIL 1))))
(S.V (S ((VP ((V NIL 1))))))
(S.VP (S ((VP NIL))))
)
(LISPXPRINT (QUOTE (V: RULELISTS)) T)
(RPAQQ RULELISTS (TOPICRULES PRERULES DRULES SETRUL SETRUL? NPRRULE
TERMRULE HEADRULES DALL SSETRUL QWORDS SEM-NOUNS SEM-VERBS AVERAGEFLAG))
(DEFINEV
(TOPICRULES (TOPIC\NOT-NP TOPIC\NOT-S TOPIC\AUTHOR NIL
                          (OR TOPIC\TERM TOPIC\TERM2 TOPIC\ESP
                              TOPIC\EMPHASIS TOPIC\NR.S TOPIC\NR.NP
                              (AND TOPIC\OR-S TOPIC\AND-S TOPIC\OR-NP
                                   TOPIC\AND-NP TOPIC\ADJ.NP
                                   TOPIC\NP.COMPL TOPIC\S.COMPL
                                   TOPIC\PP TOPIC\AUTHOR2
                                   TOPIC\PP.COMPL TOPIC\REL
                                   (OR TOPIC\ADJ.COMP
                                       (AND TOPIC\ADJ (OR
                                                       TOPIC\ADJ.SUPER
                                                           TOPIC\N)
                                                   TOPIC\NPR)
                                            TOPIC\ADJ-N TOPIC\ADJ-NPR)
                                   TOPIC\NOM TOPIC\NR.S TOPIC\NR.NP
                                   TOPIC\S.NP TOPIC\V-INTRANS
                                   TOPIC\V-TRANS TOPIC\V-TRANS2
                                   TOPIC\S.OBJ TOPIC\S.PP))))
(PRERULES (S:AND S:OR S:DCL S:IMP S:WHQ S:QREL-NEG NIL S:QREL S:YES/NO
                 NIL PR1 NIL PR2 NIL PR3 NIL PR4 NIL PR5 NIL S:NPU
                 S:NPQ NIL PR6))
(DRULES (N:NUMBER NIL D:MASS NIL NP:NPR D:SOME D:NIL D:HOWMANY
                  D:ATLEAST D:EXACTLY D:MORETHAN D:LESSTHAN D:ATMOST
                  D:EVERY D:ALL-PL D:NEG D:THE-SG D:WHQ-SG D:WHQ-PL
                  D:EACH D:ORDINAL D:CARDINAL D:ANAPHORA D:SEMI-ANAPHOR
                  D:ALL\ONES D:THE-SG2))
(SETRUL (D:SETOF NIL D:NOT-SET))
(SETRUL? (D:SET1 NIL D:NOT-SET))
(NPRRULE (NP:NPR))
(TERMRULE (ANY:TERM))
```

```
(HEADRULES (((S (NPU))
             (QUOTE NPU))
            ((S (NPQ))
             (QUOTE NPQ))
            ((S (AND))
             (QUOTE AND))
            ((S (OR))
             (QUOTE OR))
            ((S (Q (S NIL)))
             (QUOTE Q))
            ((S (Q (VP NIL 1)))
             (HEAD 1))
            ((S (DCL (S NIL)))
             (QUOTE DCL))
            ((S (DCL (VP NIL 1)))
             (HEAD 1))
            ((S (IMP (S NIL)))
             (QUOTE IMP))
            ((S (NEG (S NIL)))
             (QUOTE NEG))
            ((S (NEG (VP NIL 1)))
             (HEAD 1))
            ((S ((VP NIL 1)))
             (HEAD 1))
            ((VP ((VP NIL )))
             (HEAD 1))
            ((VP ((V NIL 1)))
             (TERM 1))
            ((NP ((NPR NIL 1)))
             (TERM 1))
            ((NPR NIL 1)
             (TERM 1))
            ((NP ((N NIL 1)))
             (TERM 1))
            ((N NIL 1)
             (TERM 1))
            ((NP (OR))
             (QUOTE OR))
            ((NP (AND))
             (QUOTE AND))
            ((ADJ NIL 1)
             (TERM 1))
            ((PREP NIL 1)
             (TERM 1))
            ((NP (NOM (S NIL 1)))
             (HEAD 1))
            ((V NIL 1)
             (TERM 1))
            ((NP ((PRO NIL )))
             (TERM 1))
```

```
                ((INTEGER NIL  )
                 (TERM 1))
                ((UNIT NIL 1)
                 (TERM 1))
                ((PRO NIL 1)
                 (TERM 1))
                ((NP (NEG))
                 (QUOTE NEG))))
(DALL (D:ALL))
(SSETRUL (D:SSET NIL D:NOT-SET))
(QWORDS (HOWMANY WHICHQ WHEN WHERE WHY HOW WHQ))
(SEM-NOUNS (AGE ANALYSIS AVERAGE BIG CONCENTRATION DOCUMENT DUST LITTLE
                OLD))
(SEM-VERBS (CONCERN CONTAIN DISCOVER GIVE LIKE PERTAIN REFER))
(AVERAGEFLAG NIL)
)
STOP
```

APPENDIX D: DOCUMENTATION OF FUNCTIONS

This appendix is designed to give a detailed description of the complete set of functions involved in the parser, interpreter, and grammar of the system. It describes each function, places it the overall framework of the system, explains how it interacts with other functions, and describes the functions of various arguments and temporary storage locations. Together with the listings of the programs, grammar, and semantic rules, it provides a thorough documentation of the LSNLIS system.

Each function is given with the names of its arguments in the form of a typical call to the function (e.g. (GETR REG WHERE) is a call to tne function GETR with arguments REG and WHERE, Those functions which take an indefinite number of arguments bound as a list to a single atom are listed as a dotted pair of the function and the argument list (e.g. (BUILD . ARGS) represents a call to the function BUILD with the CDR of the calling form bound to ARGS).

I. EXECUTIVE AND PARSING FUNCTIONS

(ABORT)

ABORT is a function for terminating fruitless paths in a parsing. It is used as an action on arcs of the grammar (usually embedded in a COND) to abort the arc if some condition is not satisfied.

(ACT ACTIONS NONTERMFLAG)

ACT is the function called by STEP to execute the actions on the arcs of the grammar. ACTIONS is the list of actions to be performed, and normally it is terminated by a terminating action (JUMP, TO, or ABORT). However, there are cases (e.g. on a JUMP arc) when ACT is called with a list of actions which is not so terminated. In this case, the argument NONTERMFLAG is set to permit a normal return from ACT. If NONTERMFLAG is not set and a nonterminated list of actions is encountered, then there is a bug in the grammar and ACT prints an error message.

Terminating actions are indicated by returning a value *L0, *L1, *L2, *END, *HELP, indicating a location in the STEP routine to which control is to resume. Terminal acts other than ABORT also have the responsiblity of setting up the configuration on which STEP is to operate.

(ADDL REG EXPRESSION)

ADDL is a function for use on arcs of the grammar for adding to the contents of a register. It adds the results of evaluating EXPRESSION to the left of the contents of the register REG (which must be a list).

(ADDLEX STRINGPOS LEXPAIR)

ADDLEX is a function to add lexical information to a table LEXTABLE which keeps a record of the results of calls to LEXIC. STRINGPOS is a pointer into the input string where the current word begins, and LEXPAIR is a pair consisting of the word found and a pointer into the input string immediately following the word. In the current system these string pointers are merely the LISP pointers into the list of words which make up the sentence and the dotted pair consisting of the word and the rest of the sentence is the same pointer as the pointer to the beginning of the word. However, the function ADDLEX was written to permit more flexible use of STRINGPOS.

(ADDR REG EXPRESSION)

ADDR is like ADDL except that it adds information to the right end of the list in register REG.

(ADJVERB)

ADJVERB tests whether the verb was produced from a predicate adjective replacing a copula verb, In this case, the verb has the form (V ADJ ---), and ADJVERB tests for the presence of ADJ.

(ALT.STACK ALT)

This function extracts the stack from an alternative ALT. It is used in several places by other functions. It is one of a large class of such functions which have been named to aid the readibility of the programs. All functions containing a period in their names in this fashion are functions for extracting information from a list that is functioning as a special "data type". The portion of the name before the period names the data type to which the function applies, and the portion of the name which follows the period names the "field" of the data type whose value is being extracted. Thus an expression of the form (ALT.STACK X) embedded in a piece of code tells the reader both that X is an ALT (the data type for alternatives in the grammar) and that the value of the expression will be the stack which was saved in that alternative.

(ALT.STATE ALT)

This function is similar to ALT.STACK, but extracts the state of the alternative.

(ALT.STRING ALT)

This function extracts the string from an alternative.

(ALT.WEIGHT ALT)

Extracts the weight associated with an alternative. (The weight of an alternative is a measure of how "likely" the alternative is.)

(ALTARC.ACONFIG ALT)

ALTARC.ACONFIG is a function for extracting the ACONFIG associated with an ALTARC alternative from the entry for that alternative.

(ALTARC.ARCS ALT)

This function extracts the list of arcs from an ALTARC alternative.

(ALTARC.TRAIL ALT)

This function extracts a pointer to the TRAIL entry from an ALTARC alternative ALT.

(ALTARCGEN)

This function is called in a number of places in the STEP function to store ALTARC alternatives. If there are arcs which are as yet untried (and if LEXMODE is not set) then an alternative is stored which will enable those arcs to be ried later if the current path is not successful. (LEXMODE is set during the processing of certain parts of reduced conjunctions, when the parser is constrained to follow the trail left by a previous parsing of the same string, and ALTARC alternatives are not generated.)

(ALTCONJGEN PPATH)

ALTCONJGEN generates ALTCONJ alternatives for restarting a previous configuration of the parse on the string which follows a conjunction (it is called by SYSCONJ as part of the system facility for handling reduced conjunctions). PPATH is the partial path entry from some previous configuration which saves the necessary information for restarting.

(ALTLOC ALTS STATE STACK)

ALTLOC is a function which locates alternatives for the selective modifier placement facility. It looks for the alternative which is in a specified state and has a specified stack pointer.

(ARCPICK LOCX LOCY)

ARCPICK is a function used by the selective modifier placement facility to select a particular arc out of a list of arcs and replace it by an equivalent arc which can only be followed if a flag *SPOP is not on. This enables the selective modifier placement facility to take a particular arc out of a list of arcs and replace it with an equivalent arc that can only be taken if in the context of another SPOP.

(ASSIST)

ASSIST is a function called by PARSER when no parsings have been found and no more alternatives remain to be tried. When the flag ASSISTFLAG is not set, it has no effect. Otherwise, it locates the blocked configuration which got the farthest into the input string before blocking, and executes a call to HELPER to allow the user to investigate. (It identifies itself with the typed message "ASSISTANCE:"). This function is used only for system debugging, and ASSISTFLAG would normally be off for users. If the call to HELPER is terminated with RPT instead of OK, ASSIST will attempt to resume the parsing from that point.

(BACKUP)

This function is designed for use in an ASSIST break (i.e. a call to HELPER from ASSIST). It allows the user to back up the configuration along tne path leading to it. Thus the user can back up along the computation of the blocked configuration (and by terminating the call to HELPER with RPT he can restart the computation from the configuration to which he has backed up). Each call to BACKUP backs up the configuration one step and prints out the state of the configuration after the backup.

(BUILD . ARGS)

BUILD is a function which can be used on the arcs of the grammar to build sentence structure. It takes an indefinite number of arguments, the first of which is the name of a structure fragment. The remaining arguments of BUILD (if any) are items to be substituted for specially marked leaves in the structure fragment. The structure fragment is processed from left to right, and when a node + is encountered the next item in tne remaining argument list is taken as the name of a register whose value is to be inserted for the symbol +. When a node # is encountered, the next item is taken as a form to be evaluated, and the resulting value is substituted for the symbol #. In addition, when the symbol * is encountered, the current value being scanned by the pointer * is copied into the structure, and where subexpressions of the form (@ X1 X2..Xn) are encountered, a single list is generated which is the result of appending X1, X2, ... Xn (which must be lists) into a single list. BUILD1 and BUILD2 are the functions that do most of the work.

(BUILD1 X)

See BUILD.

(BUILD2 X)

See BUILD.

(BUILDQ . ARGS)

BUILDQ is like BUILD except that its first argument is taken literally as the structure fragment while BUILD's first argument is evaluated.

(CAT CATEGORY)

CAT is a function for use in conditions on arcs in the grammar for testing the syntactic categories of the current word being scanned. CATEGORY can be either a single syntactic category name or a list of category names. CAT is true if the current word can be in the indicated category or one of the indicated categories.

(CATCHECK CATEGORY FLAG)

CATCHECK is the function used by CAT and by STEP for accessing the dictionary to determine whether the current word can be a member of a particular category. It returns a list (called a form-features list) whose first member is the standard (uninflected or "root") form of the word as used in this category and whose remaining elements are inflectional features associated with this word used as this category. In addition, if there are other such lists (corresponding to different senses of the word) for the same syntactic category and the flag FLAG is set, then an ALTCAT alternative is generated to enable that alternative to be pursued later. This flag is set when STEP calls CATCHECK for a CAT arc, but is not set for other uses.

(CCHECK TEMPLATE MLIST)

CCHECK is the function which checks semantic conditions in templates during the matching of semantic rules by the interpreter. TEMPLATE is the template in question, and MLIST is a LIST of possible matches for the template which are to be screened by CCHECK. Each element of MLIST is an ALIST whose elements are dotted pairs of node numbers in the template and their corresponding matches in the tree.

(CHANGEWORD . ARGS)

CHANGEWORD is a function for use in a REQUESTDEF break when the system encounters an unknowwn word. It takes an indefinite number of arguments ARGS, and it patches this list into the input string in place of the current word. If ARGS is NIL, then the current word is deleted. It performs the necessary side effects to enable the parsing to continue as if the resulting string

were the one originally typed.

(CHECKF CATEGORY FEAT)

CHECKF is a function which checks the current word (*) to see if it has feature FEAT under syntactic category CATEGORY in the dictionary. FEAT may be a list of features instead of a single feature, in which case CHECKF is true if * has any of the indicated features. The dictionary checking is actually performed by CHECKF1.

(CHECKF1 CATEGORY FEAT)

See CHECKF.

(CHOOSEALT N)

CHOOSEALT is a function for debugging grammars which enables the user to specify the alternative which will be tried next when the user types GO(PARSE) to TALKER. The number N refers to the number which is printed out with the alternative when running with TRACEFLAG set to T.

(COMPARATIVE ARGS)

COMPARATIVE tests whether the verb was produced from a comparative predicate adjective (e.g. JOHN WAS BIGGER...) replacing a copula verb. In this case, the verb has the form (V ADJ ... COMPARATIVE), and COMPARATIVE tests for the presence of the word COMPARATIVE.

(COMPFORM ENDING)

COMPFORM is a function to check whether the given ending is the indicated ending form the comparative form of the adjective *. It is used in MORPHTABLE for the morphological analysis of comparative adjectives.

(CONJOIN)

This function is one of three primary functions in the SYSCONJ facility. (The functions are CONJOIN, POPCONJ, and SYSCONJ.) It is never called explicitly from any part of the code, but a call to CONJOIN is placed on the stack by SYSCONJ along with the current status of the configuration which is being suspended in order to restart an earlier configuration on the string which follows the conjunction. When the restarted configuration has completed the construction which it was building, control will pop to this special stack entry, and the function CONJOIN will be executed to resume the suspended configuration on some tail of the string

consumed by the restarted configuration, CONJOIN enumerates all possible such tails, and generates alternatives for each of them. It also gathers up multiple conjuncts into a single level conjunction (i.e. (AND A B C) instead of (AND A (AND B C))--this is done by the first COND in the program) and it uses a heuristic strategy for selecting the "preferred" tail to try first. The strategy is that if the word which immediately preceded the conjunction word (i.e. (CAR (PATH.STRING (CAAR TEMP)))) is repeated somewhere in the string consumed by the restarted configuration, then the preferred place to resume the suspended alternative is immediately after this word. The location of such an alternative is performed by the loop at L1.

(CONJSCOPE SCOPEWORD CONJ)

This function is a predicate which is true of a conjunction and its left-hand scope indicator. That is, CONJSCOPE is true when SCOPEWORD is "both" and CONJ is "and" or when SCOPEWORD is "either" and CONJ is "or". It is used in CONJSTARTS for selecting preferred restart configurations.

(CONJSTARTS PPATH STATES)

This function computes restart configurations for SYSCONJ. It returns a list of the possible restart configurations ordered with the most likely one last, so that when they are placed on the ALTS list in the order given, the most likely one will be the most recent ALT. CONJSTARTS operates by backing up the partial path (PPATH) leading to the configuration where the conjunction was encountered and picking up possible restart configurations. It is forced to back up across at least one word in the string, and the flag FIRSTFLAG is used to remember whether this conditon has been met. It backs up along the partial path at each level until it is exhausted and then goes up the stack one level and starts backing across that level. It is forbidden, however, from backing up the stack beyond a previous SYSCONJ entry. When such a condition is encounttered, or if the stack is emptied, then QUITFLAG is set to indicate that all possible configurations have been found. It will not generate a restart configuration which was reached as a result of a JUMP arc, since that would duplicate a configuration which can be reached by restarting the configuration at the beginning of that JUMP arc. Also, if CONJSTARTS is given a list of states as its second argument STATES, then only restart configurations in those states will be considered; otherwise, any state is possible.

CONJSTARTS uses several heuristics to locate preferred restart configurations. In the course of operation it selects up to three preferred alternatives (PREFERRED0,PREFERRED1 and PREFERRED2, favored in that order). PREFERRED0 is that alternative, if any, which is indicated by a scopeword for the current conjunction ("both" for "and" or "either" for "or"). PREFERRED1 is the alternative that is indicated by a repeated word--i.e. when the word which immediately follows the conjunction occurs in the preceding string. PREFERRED2 is the alternativee which is the beginning of the current constituent being built.

(CONSTITUENTS NODE)

CONSTITUENTS is the function which when applied to a node of a parse tree yields a list of the immediate constituents (daughters) of that node. For the tree notation currently in use, this is simply the CDR of the node.

(CONTRACTP WORD)

CONTRACTP is a function for morphological analysis of words which appear to be contract numbers. It is only a crude approximation to an actual recognizer of contract numbers. It is only called for words which are not already in the dictionary, and hence will not cause any conflicts with words which are entered in the dictionary that might meet its conditions but not be contract numbers. The conditions are that the word contains at least one hyphen, one numerical digit, one alphabetic letter, and no other punctuation marks.

(CTYPE ENDING)

CTYPE is a predicate used in MORPHTABLE entries for determining whether the conjugation type of the current word is the same as that given as the argument ENDING.

(DDEF . ARGS)

DDEF is a function for adding entries to the dictionary. It is not called explicitly by any functions, but is intended for use by a user or systems programmer at the executive level or in a break. ARGS should be a list whose first element is the word to be defined, and whose remaining elements in pairs are property names and property values to be added to the dictionary entry for the word. It also adds the word to the global list DICTIONARY.

(DETBUILD)

DETBUILD is a function which is called by arcs of the grammar to build the determiner structure of a noun phrase. It combines the contents of the registers POSTART and DET with the appropriate structure.

(DETOUR)

DETOUR is the function which chooses the next alternative to be tried when the parser encounters a dead end or is instructed to find another parsing. It searches for the most recent alternative with the lowest weight.

(DICT? WORD)

DICT? is a function for examing the dictionary entry for a word WORD. It is not called by any function, but is intended for use by a user or systems programmer.

(DICTCHECK LEX CATEGORY)

This function checks the dictionary entry for the word LEX to see if it can be analyzed as a member of the syntactic category CATEGORY. It returns a list each member of which is a list consisting of a standard (root) form of the word and the inflectional features associated with that word as an inflected form of the indicated root. This is the function which decodes the various types of abbreviated dictionary formats into the standard list of form-features lists.

(DICTFETCH WORD)

DICTFETCH is a function which retrieves the dictionary entry for a word from the external dictionary file. The name of the external file is assumed to be on a global variable DICTFILE. DICTFETCH does a binary search on the file for the indicated word after first checking to verify that it has not already tried to find this word on the file before. In addition to calling in the dictionary entry for the word, it also calls itself recursively to obtain the entries for certain related words (see RELATEDWORDS).

(EQU N . ARGS)

EQU is a predicate used in the conditions in semantic rules. It verifies that the terminal string dominated by the node numbered N is equal to the string ARGS.

(EVALLOC FORM)

EVALLOC is a function which is used by several of the functions which are used by the grammar. FORM is an argument list for some function, and EVALLOC decides whether the argument is the name of a register (in which case it applies GETR to get the contents of a register) or a form for EVAL in which case it calls EVAL. It also handles the evaluation of the special current constituent pointer *, and fills the current constituent in as a default for certain cases where an argument is missing.

(F.NODE FRAGMENT)

F.NODE is a function used in the tree match facility to obtain the node name of a node in a partial tree fragment of a template.

(F.REF FRAGMENT)

F.REF is a function to access the reference number if any associated with a node in a partial tree fragment of a template.

(F.SONS FRAGMENT)

F.SONS is a function for accessing the list of constituents of a node in a partial tree fragment.

(FILEMATCH WORD FILE POS)

FILEMATCH is a function used by DICTFETCH to compare a given word with the word at the specified position POS on the external FILE. It returns one of the atoms GREATER, LESS, or EQUAL.

(FRONTED? SONS)

FRONTED? is used when the parser is doing simultaneous interpretation. It runs tnrough the SONS of a node, looking for prepositional phrases which were preposed in surface structure, and returns a list of those that were. This list is used by SORTREFS in deciding quantifier ordering, since quantifiers in fronted prepositional phrases retain their surface structure scope.

(GETF FEATURE)

GETF is a function which obtains the value of a feature FEATURE for the current word on a CAT arc. As a side effect of the call to CATCHECK on a CAT arc, the atom FEATURES is bound to the list of inflectional

features associated with the current inflected word. GETF accesses features from this list.

(GETLEX STRINGPOS)

GETLEX is a function for accessing the LEXTABLE constructed by LEXIC.

(GETR REG WHERE)

GETR is a function for getting the contents of a register REG. During the parsing, the contents of the registers are kept on a list REGS of alternating register names and register values. GETR searches this list for the register REG. However, there are times when one wants to get the contents of a register at some higher level on the stack. The argument WHERE allows for the specification of the stack location. If WHERE is T, then the top level is used. If it is "NEAREST" then GETR searches the current level and then successively looks up the stack until it finds an instance of REG. If WHERE is a number, then GETR looks at that numerical stack position. Otherwise WHERE can be a condition on the STATE, REGS, and ACTIONS of a stack entry which determines the level of the stack to use.

(GETROOT WORD CATEGORY)

GETROOT is a function for obtaining the root form of the word WORD viewed as a member of category CATEGORY. Note: if there are several possible roots, only the first one in the dictionary is found. e.g. (GETROOT (QUOTE SAW) V) will return SAW or SEE depending on which is first in the dictionary entry under the category V.

(HOLD FORM FEATURES)

HOLD is a function for use on arcs of the grammar which adds items which have been found in the sentence in some position other than their legitimate deep structure position to a special HOLD list. Entries on the HOLD list may later be recognized by VIR arcs in the grammar as if they had been found at the point in the sentence where the VIR arc is applied. The values of FORM and FEATURES are saved on the list so that when the VIR arc is applied, * will be bound to FORM and FEATURES will be bound to the saved value of FEATURES.

(HOLDSCAN HLIST CATEGORY TST)

This function scans the list HLIST (which will be the HOLD list) for elements of the type CATEGORY which meet the condition TST. It is used for processing VIR

arcs in the function STEP.

(HYPHENADJ WORD)

HYPENADJ is a predicate which is true of words which look like hyphenated adjectives. It is a crude approximation of a function to recognize hyphenated strings of English words. HYPHENADJ is true if WORD contains at least one hyphen, at least one alphabetic character, no numeric digits, and no other punctuation marks. It is used in the morphological analysis of adjectives.

(JUMP S)

JUMP is a function which can be used on arcs of the grammar to indicate transition to a new state without advancing the input string. It does this by setting up the new configuration CONFIG and returning *L1 to indicate that the function STEP is to continue at location L1.

(LEXALIZE STRING)

LEXALIZE is a function which obtains the list of (lex . stringpos) pairs which can be obtained from a given string by the compression of compound expressions into single lex's and by lexical substitutions.

(LEXIC ALTS)

LEXIC is the function whose job is to determine the next word in the input string. It is called by PARSER whenever the input string is to be moved. LEXIC provides for the expansion of contractions, the substitution of some synonyms, the compaction of compound phrases which are to be treated as single words, and the requesting from the user of definitions for unknown words.

The next word in the sentence is not always uniquely determined, and for this reason, LEXIC is designed to enumerate the alternative possible "next words". It does this as follows: When LEXIC is called, it chooses one of the possible next words and sets up the value LEX to hold it (and adjusts STRING accordingly so that LEX is (CAR STRING). If there are other possible "next words", then LEXIC generates alternatives (ALTCOMP's or ALTSUB'S) for these and returns a list of them as its value. If there is only one possible choice for LEX, then LEXIC returns NIL. The portion of PARSER which calls LEXIC takes any alternatives returned by LEXIC and generates an ALTLEX alternative on the parser's ALTS list. In restarting one of these ALTLEX's PARSER will call LEXIC with a list of

alternatives ALTS, and LEXIC will generate the appropriate choice for another "next word", Thus the first COND in LEXIC tests for whether LEXIC has been called in this mode to enumerate another alternative, and if so branches to location ALT.

Normally, dictionary entries are stored on the property lists of atoms which are provided by the LISP system. However, numbers and the special atom NIL are not permitted to have property lists in LISP nor can pieces of list structure have property lists. However, we would like to be able to recognize such constructions when they occur in input sentences, and therefore LEXIC tests for these special types of LEX, If the input "word" is one of these types, then the functions of the morphological analyzer which are stored on MORPHTESTS can recognize them, and LEXIC will consider them known possible next words. In addition, for pieces of list structure, LEXIC will consider the possibility that the parentheses in the input were superfluous and will generate an ALTCOMP alternative in which the parentheses have been removed. This alternative will not be tried, however, unless there is no other way to parse the sentence.

If the input word is not one of the special forms discussed above, then LEXIC checks to see if the word has a dictionary entry (PLIST determines whether the property list is empty) and, if not, whether the word can be derived by regular inflection from a known word (the function MORPH performs this type of morphological analysis). If the word turns out to be known for any of these reasons, then the routine branches to location SUBSTITUTE? to consider possible substitutions or compound phrases.

If the "word" is not known, one possible reason is that it contains some punctuation marks that were not separated from it by a space or that it has been run together with another word with only punctuation marks separating them. The next thing which LEXIC does is to look for such punctuation by unpacking the characters of the word and processing them to look for punctuation.

When a word is known to the system by virtue of having a dictionary entry, then LEXIC looks to see whether the dictionary specifies a substitution to be performed. If so, it will find on the property list of LEX the property SUBSTITUTE followed by a list of alternative substitutions. Each substitution is a list (possibly NULL) of words to be inserted in place of the current word in the input string. If there is more than one substitution, then the first one is taken and an

ALTSUB alternative is generated for the rest. Following the testing for substitutions, LEXIC checks for the presence in the dictionary of a COMPOUNDS entry which indicates that LEX can begin a compound phrase. The value of the property COMPOUNDS is a search tree for the possible compounds that can begin with LEX, and LEXIC compares this tree with the sequence of words following LEX in the input string. If it finds a match, it chooses the longest one and generates ALTCOMP alternatives for any shorter ones.

(LEXPAIRS STRINGPOS)

LEXPAIRS is the function called by LEXIC to produce the list of (lex . stringpos) pairs which can be found at the indicated position STRINGPOS. Each pair indicates a possible lex together with the stringpos which immediately follows it.

(LIFTR REG FORM WHERE)

LIFTR is a function for setting register contents at higher levels on the stack. REG is the name of the register, FORM is the value to which it is to be set, and WHERE is a specification of the level on the stack at which the register is to be set. WHERE permits the same options for LIFTR that it does for GETR with the esception of "NEAREST".

(LONGBLOCK)

LONGBLOCK is a function which is called by ASSIST to determine the blocked configuration which got the farthest through the input string when a sentence is unparsable. This is a likely site for the error which caused the sentence not to parse, especially if the error has to do with the dictionary entry for a word.

(MARKER X Y)

MARKER is a function which checks whether the word X has the semantic marker Y.

(MEMBSTACK PTR STACK)

MEMBSTACK is a function used by PARSER as part of the well-formed substring facility to test whether a given alternative could possibly add to a given position in the string. STACK is the stack of an alternative from the ALTS list and PTR is a pointer to a list of well-formed substrings. MEMBSTACK returns T if STACK contains the pointer in its WFST3 entry at some level.

(MODAL)

MODAL is a predicate for use on the arcs of the grammar and is true if the current value of * is a modal verb.

(MODESET MODE)

MODESET is a function for initializing some standard mode settings. MODE T causes all parsings to be obtained and interpreted and executed. MODE 1 does parsing only, MODE 2 does parsing and semantic interpretation, and MODE 3 does parsing, semantic interpretation, and execution.

(MORPH LEX CATEGORY CMODE)

MORPH is the function which performs morphological analysis for rugularly inflected words. LEX is the word being MORPH'ed. If CATEGORY is a single category name, then the analysis is performed for that category only; but if CATEGORY is NIL, then the analysis is performed for all possible categoreis. MORPH makes use of two tables -- MORPHTESTS and MORPHTABLE. The first contains arbitrary LISP tests for particular types of words, while the second contains inflectional endings.

MORPHTESTS is used both in MORPH and in CATCHECK; it consists of a list of entries for different syntactic categories, with each entry consisting of the name of the category and a series of two-element lists which specify a predicate to be tested and a form to be returned as the form-fetures list if the predicate if true. A simple MORPHTESTS entry would be (INTEGER ((NUMBERP *) (LIST *))), indicating that any wrd * which passes the test (NUMBER *) will be considered as an instance of the syntactic category INTEGER, with a standard (root) form identical to itself (*) and with no inflectional features. CMODE is a flag which can be set to skip the MORPHTESTS analysis.

MORPHTABLE indicates the possible inflectional endings for regularly inflected words, and the procedures for obtaining the underlying root forms for inflected wrds. MORPHTABLE also contains entries for several different syntactic categories. Each entry specifies a syntactic category and then a sequence of entries of the form:

(E- E+ CATEGORY CONDITION FEATURES*)

where E- is an ending to remove from the end of the word (if it is not there, then the rule doesn't apply), E+ is an ending to add to the stem that results from

subtracting E-, CATEGORY is the syntactic category of the root which is to be checked, CONDITION is a condition whch must be true of the root when viewed as category CATEGORY; and FEATURES (there may be any number of them) are the inflectional features whichare to be associated with the work if the condition is satisfied. The CONDITION in the rules is used to verify that the tentative root is indeed in the class of wrds which undergo the regular inflection represented by the rule. For example, the entry (N ((S) NIL N (PLURAL -S) (NUMBER PL))) says that if we re looking for a noun (N) and if the wrd ends in S, then we remove the S from the end, add nothing (NIL) and look at the resulting wrd as a noun (N) to test the condition (PLURAL -S) (which tests the dictionary entry for the word for the property N with value -S) to see if the word undergoes this type of inflection. If so, then the inflectonal features associated with the word consist of the single feature (NUMBER PL).

If a dictionary entry is computed for a word by means of morphological analysis, then it is added to the property list for that word for the duration of the console session with the system. Thus, the morphological analysis described will be done only once for each word for which it is required.

(MORPHTABCHECK TABLE)

MORPHTABCHECK is the function which tests the entries in the MORPHTABLE for MORPH. It returns a list of appropriate form-features lists for the dictionary entry if a line of the table is successful and NIL otherwise.

(MORPHTSTCHECK TAB)

MORPHTSTCHECK is the function which checks entries in the MORPHTESTS table for MORPH. It also returns a list of form-feature lists.

(NEGADV WORD)

NEGADV is a function which tests for negative adverbs such as "hardly", etc, which cause subject/verb inversion when the begin a sentence.

(NEXTWRD WRDS)

NEXTWRD is a function which is used on arcs of the grammar and returns the next word in the input string.

(NPBUILD)

NPBUILD is the function which builds the syntactic tree structure for a noun phrase. It is used on POP arcs in the grammar.

(NPCHECK NODE TERMINALS)

NPCHECK is a function used in PNCHECK for testing constituents of a noun phrase node. It uses the free variable NP which is bound to a noun phrase node by PHCHECK and looks for a constituent of the noun phrase of type NODE. It checks whether the immediate constituent of this NODE is a member of the list TERMINALS.

(NPREP ARGS)

NPREP tests whether a given preposition is usually associated with the head noun of a noun phrase. ARGS may be a preposition or a prepositional phrase, in which case its preposition is extracted and tested. This is a primitive foray into correct modifier placement, but it works in more cases than not.

(NULLR REG)

NULLR is a predicate for use in conditions in the arcs of the grammar for testing whether the register REG is empty.

(ORFLAG X)

ORFLAG is a function which can set a special mode that tells the system to interpret all "and" conjunctions as if they were "or" conjunctions. This mode can be used by some users who habitually say "and" when they mean "or" in document requests. When X is T the special mode is set, and when X is NIL it is reset.

(PARSELIST SENTLIST)

PARSELIST is a function for debugging a system. It takes sentences successively from the list SENTLIST and processes them as queries.

(PARSER STRING MODE ALTS)

PARSER is the controlling routine of the parsing component. STRING is the sentence to be parsed and MODE is a variable which governs the mode in which the parsing is to proceed. (ALL causes all parsings to be found, SPLIT causes all parsings to be followed in parallel, and non-null values in general cause automatic selecton of a

new alternative number, whenever a blocked configuration is encountered.) ALTS is a list of alternatives which is NIL unless PARSER is being called to continue looking for parsings, in which case it will be list of alternatives generated by a previous call to PARSER, PARSER returns a list whose first element is a list of parsings found, and whose second element is a list of alternatives which it did not try It is this list of alternatives which can be used to continue looking for additional Parsings if the first one is found not be Satisfactory.

PARSER manages a list of active configurations (ACFS) which its calls the function STEP to advance. A configuration consists of a complete record of a state of the machine -- i.e., a list of the state, stack, registers, contents of the HOLD list, and a path entry which records the history of how the cofiguration was reached from the initial configuration.

PARSER runs in two modes depending on the setting of a flag LEXMODE. In the normal mode, LEXMODE is NIL and the parser proceeds by calling LEXIC to determine the next word in the string. LEXMODE is set when the parser is operating on a reduced conjunction during the part of the processing when the suspended configuration for the first conjunct is being resumed on a tail of the string consumed by the second conjunct (See SYSCONJ and CONJOIN). At this time, the parser follows the trail (TRAIL) left by the previous parsing of this substring, and the normal lexical analysis is bypassed. This is due to the fact that the two components of the conjunction are required to analyze the shared substring in the same way.

If after calling LEXIC, the current word LEX is still an unknown word, then the configuration is added to a list of blocked configurations and the parsing is aborted under the assumption that no other alternatives will be able to parse beyond the unknown word in the sentence. When LEX is a known word, however, PARSER calls the function STEP to advance the active configurations and produce a new list of active configurations at the next position of the input string, it advances the input string to the next position and repeats. If at any time there are no new active configurations, then depending on the setting of the flag MODE (which is normally set to the value of the global flag PMODE) it either goes into a break at location HELP, or it selects an alternative to be tried by calling the function DETOUR at location ALT. If there are no more alternatives, but there have been some complete parsings found (if so they are stored on VALUES by the function POP which is executed in interpreting the POP arcs in the

grammar), then PARSER returns those parsings. If this call to parser was not itself an attempt to find additional parsings (in which case ALTFLAG would be set), then the failure to find any parsings of the sentence will cause a call to ASSIST. This is the function which would eventually contain facilities for making helpful diagnostic comments to the user as to the likely cause of the error, and perhaps even correct them and continue. At the moment it merely goes into a break (if the flag AHELP is set) at the blocked configuration which got the farthest into the string before blocking.

The various locations ALTCONJ, ALTLEX, and ALTARC know how to restart their corresponding types of alternatives, which have been found on the ALTS list by DETOUR.

(PATH.ARC PATH)

PATH.ARC is a function which extracts the last arc followed from a path entry.

(PATH.STRING PATH)

PATH.STRING is a function which extracts the current string position from a path entry.

(PATH.VAL PATH)

PATH.VAL is a function which extracts the VAL (i.e. the word or construct "consumed" by the last transition) from a path entry.

(PLOG N FILE)

PLOG is a function which prints out a record of the sentence processing to a file. N is the number of phases of the processing to be printed (1. parsing, 2. interpretation, and 3. execution). It is called by QGO when LOGFLAG is set.

(PLURAL ENDING)

PLURAL is a function for use as a condition in the MORPHTABLE of the morphological analysis component. It tests whether the dictionary entry for the current word * is marked as a noun with regular inflection of the type ENDING.

(PNCHECK NP PNCODE)

PNCHECK (person-number check) is the function which checks for person-number agreement between a noun phrase

(NP) and a person-number code (PNCODE). It is used in the grammar to check person-number agreement between verbs and their subjects.

(POP POPVAL POPFEATURES)

POP is the function which returns from a recursive call in the transition network grammar. It is used by the function STEP for the interpretation of POP arcs, and can occasionally be used as an action on an arc of the grammar. POPVAL is the structure that is to be returned from the recursive call (and bound to the current constituent pointer *) and POPFEATURES is the list of features which is to be associated with the current constituent.

POP restores the configuration which was saved on the stack at the time of the PUSH which initiated the present level of computation and performs the actions on the push arc, after setting the flag NOMOVEFLAG which indicates that the function TO at the end of the PUSH arc is not to advance the input string (since it has already been advanced by the recursive computation). If the stack is empty, and the STRING is also empty, then POPVAL is a complete parsing of the sentence, and is added to the list VALUES which is being maintained by PARSER. If the string is not empty at this time, then the configuration is blocked.

(POPARC.FEATURES ARC)

POPARC.FEATURES is a function for extracting from a pop arc a form which evaluates to a list of features to be associated with the construction which is being returned by the pop arc.

(POPARC.FORM ARC)

POPARC.FORM is a function which extracts from a pop arc a form which is to be evaluated to produce the structure which is to be returned by the pop arc.

(POPCONJ)

POPCONJ is one of the functions used for the facility which handles reduced conjunctions (see CONJOIN and SYSCONJ). A call to POPCONJ is placed on the stack by CONJOIN when it resumes the suspended configuration for the first conjunct in a conjunction. This call to POPCONJ will be invoked when the first conjunct has been completed, at which time it will determine whether the two components of the conjunction are compatible, compute the syntactic representation of the conjoined phrase, set

up a configuration on the alternatives list for the computation which is to be resumed at this point, and abdicate control by returning *END as its value. This will enable DETOUR to pick up the configuration from the ALTS list and continue parsing. (This method of proceeding with the parsing is used to restore the value of the input string, which has been temporarily destroyed by the operation of STEP in LEXMODE mode, without interfering with any other active configurations which may be on the current ACFS list.)

(PPATH.ACONFIG PPATH)

A partial path (PPATH) is a path entry without a VAL (that is, it represents a path which has decided what arc to take next but does not yet have the result of the transition). It is saved in the stack entry when a computation pushes to a lower level and is used to build the full path entry when the embedded computation returns. PPATH.ACONFIG is a function which extracts the previous augmented configuration from the PPATH entry.

(PPATH.ARC PPATH)

PPATH.ARC is a function which extracts the arc followed from a PPATH entry.

(PPATH.BACK PPATH)

PPATH.BACK is a function for backing up along the path entries for a configuration. Its argument is a partial path (PPATH) which consists of a record of an augmented configuration (ACONFIG) and the arc which was followed from that configuration. It lacks the information about the computation of that arc which is a part of a complete path. (See the listing FORMATS in the computer listing for the specification of the LISP structure formats for partial paths, paths, configurations, and augmented configurations.) If the partial path PPATH is not the first one in a call to the network, then its last element is the full PATH entry recording the configuration prior to the current one, and CDR of this is the partial path associated with it. If there is no previous path as the last element of PPATH, then this configuration is the first one after some PUSH (or indeed the first one in the analysis of the string) and the preceding partial path is taken from the STACK assoociated with PPATH.

(PPATH.CONFIG PPATH)

PPATH.CONFIG is a function which extracts the previous configuration from a PPATH entry.

(PPATH.HOLD PPATH)

This function extracts the HOLD list from a partial path.

(PPATH.PATH LIST)

This function extracts the previous path entry from a partial path.

(PPATH.REGS PPATH)

This function extracts the registers list from a partial path.

(PPATH.STACK PPATH)

This function extracts the stack from a partial path.

(PPATH.STATE PPATH)

PPATH.STATE extracts the previous state from a PPATH entry.

(PPATH.STRING PPATH)

PPATH.STRING extracts the string position at the beginning of the transition from a PPATH entry.

(PPT XTR FILE)

PPT (pretty print tree) prints a parse tree (XTR) in a pretty format to the file FILE. The function which actually does the printing is PPT1.

(PPT1 XTR XID FILE)

See PPT.

(PRINTPARSES FILE)

PRINTPARSES is the function used by SENTPROC to printout the result of the parsing when the appropriate flags are set. If the global flag PPTFLAG is set, then this happens using PPT to obtain the printout in a pretty format. Otherwise, the printing is in the ordinary parenthesis notation corresponding to the internal list structure.

(PUNCTALIZE STRING)

PUNCTALIZE is a function called by LEXPAIRS as part of the LEXIC package to perform punctuation analysis on the first atom in the list STRING. PUNCTALIZE is called when the next "word" in the sentence is not in the dictionary to see if it might really be a known word with punctuation at the end or two words run together with punctuation. If this is the case, then PUNCTALIZE returns an updated string (or a list of alternative such strings) with the word and the punctuation separated. Otherwise it returns NIL.

(PUSH PS)

PUSH is the function used by STEP to interpret PUSH arcs in the grammar. It can also be used as an action on the arcs of the grammar under certain circumstances. In the normal mode (when LEXMODE is not set) it saves the current state, register contents, actions to be performed, HOLD list, and partial path on the stack and starts a new configuration at the lower level with the initial register contents from SREGS (those register contents sent down by calls to SENDR). When LEXMODE is set, then PUSH must take its constitutent from the trail which the parser is following. If nothing was sent down with a SENDR, then it merely takes the value stored in TRAILVAL for the trail being followed. If there were register contents sent down, however, then it calls REDO to follow the path associated with the computation of that constituent to construct the new constituent based on the new initial registers sent down.

(PUTLEX STRINGPOS LEXLIST)

PUTLEX is a function called by LEXIC to add the list of alternative lexical analyses at the current stringpos to the table LEXTABLE. Thus, when other paths encounter the same string postion, the lexical analysis will be available there and will not be recomputed.

(Q . QUERY)

Q is the function called by TALKER for processing input sentences. It sets the variable SENTENCE, calls SENTPROC, and logs the resulting output if LOGFLAG is set.

(QGO LABEL)

QGO is the function called by TALKER to continue looking for more parsings, to repeat a semantic interpretation, etc. It calls SENTPROC with a label LABEL which specifies a location within SENTPROC at which processing is to be started.

(QSTART)

QSTART is a predicate used in the grammar at the beginning of a sentence to determine whether it looks like a question -- i.e., it starts with an interrogative word or with an auxiliary verb.

(REDO TRAIL REGS)

REDO is a function called by PUSH when it is following a trail during the LEXMODE phase of the recognition of a reduced conjunction. It will redo the computation indicated by TRAIL starting with the register contents REGS instead of those which were originally used by TRAIL.

(RELATEDWORDS WORD)

RELATEDWORDS is a function used by DICTFETCH to determine the list of words related to a given word that should also be fetched into the in-core dictionary with it. If returns any words which are used in substitute or compounds entries in the dictionary entry for the word, and if the word is irregular it returns the root.

(RELATIVIZE FORM)

RELATIVIZE does a top level search of the list FORM for the first NP node. When found, it converts it into an appropriate form for sending down into relative clauses, i.e. it replaces the determiner with (DET WHR) and removes any prepositional phrase or relative clause modifiers. RELATIVIZE is used in state S/QP1 to make the rest of the sentence following a fronted questioned prepositional phrase, a relative clause on the head noun of the PP.

(REQUESTDEF LEX)

REQUESTDEF is the function which is called to interact with the user of the system when an unknown word is encountered by the parser. It prints out a comment followed by the unknown word, and goes into a break to allow the user to define the word (using DDEF) or to change it (using CHANGEWORD).

(RESUME ARGS)

RESUME is a function which can be called on an arc of the grammar to resume a PUSH computation which has been assigned a feature RESUME by the function RESUMETAG. This provides for the termination of a PUSH computation at one point in the string and resuming it later at

another part of the string, and it provides a mechanism for handling certain phenomena which would be called right-extraposition transformations in transformational grammar theory. ARGS is a list of registers which are to be sent down to the lower network when the PUSH is resumed.

(RESUMETAG STATE)

RESUMETAG is a function for computing a RESUME feature for a configuration which will enable it to be resumed later, beginning in state STATE. It is used by arcs of the grammar in conjunction with the function RESUME.

(RFEAT . ARGS)

REFEAT is a function for retrieving syntactic features from the dictionary entries for words, ARGS is a list whose first element is the name of the syntactic feature desired, and whose second element indicates the word whose dictionary entry is to be consulted (which may be indicated either by the name of a register which contains it, but the special pointer *, or by some other LISP expression).

(SAMPLEP WORD)

SAMPLEP is a function used by the morphology component to recognize words that look like sample numbers -- i.e. an S followed by five digits.

(SBUILD)

SBUILD is the function called by the grammar for building the syntactic structures of sentences. It gathers up the various pieces of the structure from the registers in which they have been stored and assembles them into a syntactic tree using the function BUILDQ.

(SCANSTACK TEST)

SCANSTACK is the function which scans the stack looking for a stack level which satisfies the test TEST. It is used in LIFTR and GETR for locating levels of the stack where registers are to be set or interrogated.

(SCOMP V)

SCOMP is a function which tests whether a verb V takes a sentence complement by checking the dictionary entry for V.

(SENDACTP ACTION)

SENDCTP is a predicate used by STEP and REDO for identifying actions which send register contents down to lower levels (i.e. SENDR and SENDRQ).

(SENDR REG FORM)

SENDR is a function which sets the contents of the register REG to the value of FORM at the next lower level to which control will be passed by a PUSH arc.

(SENDRQ REG FORM)

SENDRQ is like SENDR except that FORM is not evaluated.

(SENTPROC SENTENCE LABEL)

SENTPROC is the major dispatching routine for the processing of an input sentence. It dispatches the input to the various routines PARSER, SPROC, and EXECUTE, times computations, prints out intermediate results and timings, and logs the results, as appropriate. It also provides for the feedback to the parser to obtain additional parsings if the semantic interpretation of the first parsing fails (up to maximum number of times specified by MAXREPARSES), and for the redoing of a previous execution or interpretation or the continuation of parsing by calls which specify a LABEL = EXECUTE, INTERP, or PARSE, respectively.

(SETR REG FORM)

SETR is the function which sets the contents of a register REG to the value of form at the current level of processing.

(SETRE REG FORM)

SETRE is like SETR except that REG is evaluated to obtain the name of the register to be set.

(SETRQ REG FORM)

SETRQ is like SETR except that FORM is not evaluated, but taken literally.

(SETUP FILENAME)

SETUP is the function to be called by a user when he enters the sytem to set up the lower fork. FILENAME is the name of the lower fork file -= usually

<WARNER>LOWFORK.SAV.

(SHOWTIME CONSES TIME FILE)

SHOWTIME is the function which prints timing information to a file in the BBN LISP system.

(SPLIT . SPLITARCS)

SPLIT is a function which can be used on arcs of the grammar to cause two or more alternatives to be followed at once. SPLITARCS is a list of alternative continuations of the arc on which the SPLIT action occurs, and all such alternatives will be followed in parallel. This feature has not been used in the current grammar.

(SPOP POPVAL POPFEATURES)

SPOP is a function which is used to perform the selective modifier placement triggered by the SPOP arcs in the grammar. It locates the alternatives (if any) to the arc which pushed for the constituent about to be popped and determines whether that configuration could have popped instead. If so, if follows out the possibilities of that alternative to see if any of the configurations that could be reached by successive JUMPs and POPs (or SPOPS) could also push for the same consituent. If it finds any other configurations which could have pushed for this constituent, it considers them all as candidates and decides which one to follow on the basis of semantic entries in the dictionary.

(SPROC P)

SPROC is the function which begins the semantic interpretation of the node P and returns the list of possible semantic interpretations. It calls the function INTERP which does the work.

(STACKELT.PPATH LIST)

This function is used to extract the PPATH entry from an element of the stack.

(STACKELT.REGS STACKELT)

STACKELT.REGS is a function for extracting the register contents of a higher-level computation saved on the stack from the stack entry.

(STEP CONFIG ALT)

STEP is the major function of the transition network parser. Its job is to take a single configuration (CONFIG) from the active configurations list of PARSER and compute from it a list of configurations which are possible at the next point in the input string. It takes the list of arcs for the state of the configuration and considers each in turn until it finds one which can be followed. It also interprets the conditions and actions on the arcs, and generates ALTARC alternatives on the ALTS list for any arcs which remain untried when it decided to follow one.

STEP is also the function which is called to pick up the processing of an alternative taken from the ALTS list. In this case, the argument ALT will be set to the alternative to be restarted, and the setting of CONFIG will be irrelevant. In this case, STEP will branch to location ALT where it determines the type of alternative and does the appropriate thing to resume the processing.

At location LØ, STEP unpacks the configuration CONFIG into its component parts (STATE, REGS, HLIST, and PATH), and at location L1, it begins the determination of the list of arcs to be considered. If, however, the time already spent in the parsing exceeds a global limit MAXTIME, the parsing is terminated with an appropriate comment. If the current LEX (the current word in the string) is marked with the property LEXARCS, then it is an "interrupt word" and the list of arcs to be tried is not taken from the value of the state name as would usually be the case, but is instead computed by the expression which is the value of the property LEXARCS. This facility allows for the convenient handling of special function words which can occur at almost any point in a sentence with a regular effect. For example, the conjunction scope indicators "both" and "either" are handled by this facility in the current system. Another special case for the determination of a list of arcs other than that listed for the state is the SYSCONJ facility. If the flag SYSCONJFLAG is set and the current LEX is a conjunction and there are no CAT CONJ arcs leaving the current state, then the SYSCONJ facility provides its own special default CAT CONJ arc in place of the normal list of arcs. This does not happen when LEXMODE is set, however (i.e. when STEP is already interpreting a part of a reduced conjunction). When the global flag SPLIT is set, the list of arcs will be moved to a list of "untried" split alternatives (SPLITS) and control will branch to END where there is a test for uncompleted SPLITS (i.e. alternatives to be followed in parallel) before returning. Normally, however, the list of arcs is taken from the value of the state, and control passes to L2.

L2 begins the basic loop which tries successive arcs from the list ARCS. If there are no more arcs, then depending on the settings of various mode variables and other parameters, control either passes to END or HELP. Also if MODE is non-null, the blocked configuration is added to the list BLOCKS (for later use by LONGBLOCK in ASSIST). If the number of blocked configurations exceeds the global parameter MAXBLOCKS, the parsing is terminated.

L3 begins the processing of the arc selected by initializing the values of *, FEATURES, SREGS, and NOMOVEFLAG. The atom * is the pointer to the current constituent (initially it is equal to the current word LEX, but after popping from a lower level it is the value of the constituent returned, and on a virtual ARC it is the value of the constituent which is taken from the HOLD list). FEATURES is the list of features associated with the current value of *, and SREGS is the list of registers which have been sent down to the lower level by SENDR actions immediately prior to a PUSH to a lower level). NOMOVEFLAG is a flag which indicates whether the input string is to be advanced after the transition caused by the arc (it is initially set to NIL indicating that the string is to be advanced, but various actions on the arc can cause it to be reset). The major part of the function STEP consists of the SELECTQ at location L3 which determines the type of arc and performs the appropriate actions.

A CAT arc is followed if LEX can be a member of the syntactic category indicated by the label on the arc (ARC.LABEL ARC). If LEXMODE is set, however, this arc can only be taken if the word was also taken as a member of this category in the trail which is being followed. The value TEMP which is set by the call to CATCHECK or taken from the trail (TRAILVAL) is a form-features list whose CAR is the root form of the word LEX and whose CDR is a list of inflectional features for the word. These values are bound to * and FEATURES, respectively, as a result of choosing a CAT arc, and control passes to location TST which checks the conditions associated with the arc.

A PUSH arc indicates a recursive application of the network to find a phrase of the type recognized by the state which is given as the arc label. The condition on the arc is tested before the PUSH in order to determine whether to perform the PUSH. If LEXMODE is set, then the PUSH does not occur unless the corresponding entry on the trail being followed was also a PUSH to the same state. To facilitate the use of SENDR's to send register contents down into the lower level prior to the push,

there is an optional constituent of the PUSH arc immediately after the condition on the arc which consists of a list of actions to be executed prior to the actual call to PUSH. This list of actions is indicated by an initial element "!". Also for the same reason, any initial sequence of actions of the SENDR type (tested by SENDACTP) are executed prior to the PUSH. The call to the function PUSH will save the current state and register contents and the uncompleted actions on the arc on the pushdown stack for continuation after the embedded phrase has been recognized.

POP is a "pseudo" arc in the sense that it has no "destination" at the end. Rather it indicates a return from an embedded computation to the configuration which PUSH'ed for it. It is represented as an arc so that its choice can be ordered with respect to those of the other arcs and so that it can be made conditional on the context by using a test on the arc. POP arcs are not permitted when LEXMODE is set, since the PUSH's are never actually executed in this case (and therefore, the trails which are being followed never have POP arcs on them). POP's are also forbidden if there are entries on the HOLD list put on at this level which have not yet been used by any virtual (VIR) arc (this is part of the HOLD facility for dealing with left-extraposition transformations). SPOP is a variant of POP which is used in some systems for selective modifier placement but is equivalent to POP in the current system.

A JUMP arc is an arc which performs some actions but does not advance the input string. The label on the arc names the state to which control is to go after the actions are executed if there is no terminating action on the arc.

A VIR (virtual) arc is an arc which picks up a constituent from the HOLD list (placed there by a call to HOLD on some arc of the grammar) and treats it as if it had just pushed for and found the constituent at this point in the string. It sets * and FEATURES to the values taken from the HOLD list and then executes the actions on the arc (after setting NOMOVEFLAG to prevent the input string from advancing).

A WRD arc tests for the presence of a particular word in the input string. Similarly a MEM arc tests for one of a specific list of words. A TST arc allows for the testing of an arbitrary condition expressed in LISP as the condition on the arc. The label on a TST arc has no effect on the operation and can be used for purely mnemonic purposes by the grammar writer. A SUSPEND arc is an arc which suspends the processing of the current

state with an incremented weight (incremented by the amount indicated in the arc label). This can be used to control the order in which parsings are discovered by suspending "unlikely" alternatives to be tried only after more likely possiblities have been tried. There is also a SUSPEND action which can be used on an arc to suspend the processing of just that arc.

A SPLIT arc is essentially a group of arcs grouped together with the "conjunction" SPLIT to indicate that the arcs in that group are to be followed in parallel. It is similar to the SPLIT action which can be used on arcs to indicate parallel alternative "tails" for a single arc.

A DO arc is an unconditional list of actions to be performed with a destination specified at the end.

The location TST performs the checking of conditions on the arcs for a number of different arc types, and similarly the location ACT executes the actions on the arcs. The location ALT performs the appropriate actions for resuming an alternative, and HELP provides a break for user interaction in certain cases.

The location END is entered when a given configuration either blocks or is completed. It checks whther there are any uncompleted configurations (UCFS) placed there by SPLIT actions on arcs, and if so processes them. It also tests for any unprocessed configurations (SPLITS) placed there by a SPLIT arc or by the mode flag MODE being set to SPLIT, and it processes all of these before returning. When all "parallel" computations have been performed, it returns the list (VCFS) of resulting configurations which have been constructed. (The actual construction of the resulting configurations is performed by the function TO when it occurs as terminal action on an arc.)

(STORALT ALT)

STORALT is the function used in many places for placing alternatives on the ALTS list.

(SUBJLOW VERB)

SUBJLOW is a predicate which indicates whether the indicated verb undergoes subject lowering (as opposed to object lowering) when it occurs with both a direct object and a to complement. SUBJLOW is true of "promise" type verbs where the interpretation of "I promised John to go" means that I will be going (as opposed to "I ordered John to go").

(SUPFORM ENDING)

SUPFORM is a predicate for use in MORPHTABLE entries for testing the type of conjugation which an adjective undergoes for the superlative form.

(SUSPEND N)

SUSPEND is an action for use on arcs of the grammar for suspending a given computation in favor of "more likely" ones. It increments the weight associated with the current computation by the amount N and generates an ALTARC alternative on the ALTS list.

(SUSPENDW WEIGHT INCREMENT)

SUSPENDW is the function which computes the resulting weight determined by the current weight WEIGHT and the increment specified on a SUSPEND arc (INCREMENT). It currently adds the two, but is factored out as a separate function so that we could experiment with multiplicative rather than additive weights.

(SYSCONJ STATES)

SYSCONJ is the action which invokes the system conjunction (SYSCONJ) facility for reduced conjunctions. It can either be used on CAT CONJ arcs by the grammar writer, or it will be supplied automatically for states which don't have CAT CONJ arcs if SYSCONJFLAG is on. It is the first function of the trio of SYSCONJ functions (SYSCONJ, CONJOIN, and POPCONJ) to be executed. It causes the insertion of a special stack entry with a call to CONJOIN into the stacks of a set of restart configurations (computed by CONJSTARTS) and the generation of an ALTCONJ alternative for each such configuration. It then returns *END so that STEP will terminate the current configuration and pick up one of the generated ALTCONJ alternatives.

(T.NODE NODE)

T.NODE is a function which extracts the node name from a node in a tree fragment.

(T.REF NODE)

T.REF is the function which assigns to nodes in the syntax tree a reference which is used for associating information with that node in the TAGLIST. It is currently identical with the LISP pointer to the node.

(T.SONS NODE)

T.SONS is a function for computing the list of the sons of a node of a tree. It depends on the notation being used for trees in the system. If it is the two-paren notation, then the sons are the CADR of the node; otherwise they are the CDR.

(TAILS LIST)

TAILS is a function for enumerating the tails of a list. (E.g. the tails of (A B C) are (A B C), (B C) and (C).) It is used by CONJOIN for computing the possible tails on which the suspended first conjunct of a conjunction may be resumed.

(TAILS1 LIST)

TAILS1 is like TAILS but it omits the singleton tail. (E.g. TAILS1 of (A B C) gives (A B C) and (B C) , but not (C).)

(TALKER MODE)

TALKER is the major executive of the English Language preprocessor The first thing to be done by a user after loading the system is to call TALKER with an argment MODE (usually NIL) to indicate the mode in which he wants to operate. (MODE of NIL indicates use of the mode settings as they exist at that time, without change, while a non-null MODE will cause MODESET to be called to set the appropriate mode variables.) TALKER takes care of the interaction with the user, accepting sentences and LISP commands as input, and performing the appropriate actions for each. In the BBN LISP system, it also takes care of saving the input sentences on a history list which enables the user to refer to and reuse the results of his previous typein.

(THATCOMP VERB)

THATCOMP is a predicate which tests whether a verb can take a THAT complement.

(TIMEP X)

TIMEP is a function which can be called for morphological analysis of an atom which looks like a time (i.e. a number less than 24 followed by a colon followed by a number less than 60).

(TO S)

TO is the function used by arcs in the grammar to indicate the destination (next state) for an arc. It computes the configuration which results from the transition and returns a label to STEP (through ACT) indicating what location it should pass control to (which depends on such factors as whether NOMOVEFLAG is set, whether it is at the end of the string, etc.)

(TOCOMP VERB)

TOCOMP is a predicate which tests whether a verb can take a TO complement.

(TRACER . ARGS)

TRACER is a function which is called at many points in the parser for providing a tracing of the course of the parsing when the flag TRACE is set. It is an extremely valuable tool for debugging grammars, and is also a useful instructional tool for teaching the operation of the parser and the grammar.

(TRAIL PATH)

TRAIL makes a list of the path entries in a path in the order in which the transitions occur so that they can be followed by the parser in LEXMODE mode. (The normal order of entries in a path is reversed and "right nested".)

(TRAIL1 PATH)

TRAIL1 is like trail except that it skips the configurations that occur immediately after JUMP and VIR transitions.

(TRAILS PATH)

TRAILS is a function called by CONJOIN to make a list of the trails at different levels of the analysis of the right-hand shared portion of a reduced conjunction. These trails are the possible trails on which the suspended first component of the conjunction can be resumed.

(TRANS VERB)

TRANS is a predicate which tests whether a verb is transitive (i.e. can take a direct object).

(TRANSCOMP VERB)

TRANSCOMP is a predicate which tests whether a verb can take both a direct object and a complement.

(VPARTICLE . ARGS)

VPARTICLE is a function for testing whether a verb combines with a particle to form a verb (e.g. "call up," etc.). ARGS is a list whose CAR specifies the verb in question (usually by naming the register which contains it) and whose CADR specifies the particle in question (again by naming a register or by reference to the pointer *).

(VPASSIVE V)

VPASSIVE is the predicate which tests whether a verb V can be passivized. This is true either if it is so marked in the dictionary or (as a default) if it is totally unmarked for syntactic features.

(VPREP ARGS)

VPREP is a function which tests whether a verb can take a prepositional modifier with a given preposition, ARGS.

(VTRANS . ARGS)

VTRANS is a function which tests whether a given verb is transitive (i.e. whether it can take a direct object). This is true either if the verb is so marked in the dictionary under the property FEATURES, or (as a default) if the verb is not marked with any syntactic features at all.

(WRD . ARGS)

WRD is a function for use in conditions in the grammar to test whether the current word * or a word in some register is a member of a list of words. CAR of ARGS is the list of words to be tested, and CADR of ARGS specifies the word to be tested.

II. SEMANTIC INTERPRETATION FUNCTIONS

(# N)

The function # is used in templates of semantic rules to reference the node of the tree that matches the node numbered N in the tree fragment used for the match. For example, in a template (NP.N (TESTFN (# 1))), the expression (# 1) will evaluate to the node of the tree which matches node 1 of the fragment NP.N. TESTFN in this case is a hypothetical condition which is to be true of node (# 1).

(*FLAG X)

*FLAG is used by the interpreter in interpreting a node as a topic. If the user has emphasized any phrases by saying, for example, "vugs, in particular" or "especially vugs", "vugs", as a topic, is starred:

(FOR EVERY X1 / DOCUMENT : (ABOUT X1 (VUG *)...) ;
(PRINTOUT X1)).

(AGREEMENT ANTECEDANT SPECIFIER NOUN ADJ SEMARKERS)

AGREEMENT is a predicate which tests anaphorism - antecedant agreement. The criteria for agreement are:

1. The candidate ANTECEDANT has semantic markers which match those required of the true antecedant (SEMARKERS). It is not always possible to decide what semantic criteria the antecedant should meet. Hence, SEMARKERS is an optional argument. Semantic requirements are sometimes made by the dominant verb. For example, in "Does it contain Aluminum?" the antecedant of "it" must be a sample, for this question to make sense to the system. SEMARKERS for the above would be (SAMPLE).

2. A pronominal anaphorism matches any candidate antecedant.

3. If the anaphorism rests on its determiner e.g. "those analyses", the head noun of the antecedant must match the head noun of the anaphorism.

4. If the anaphorism contains any adjectives e.g. "those barium analyses", the antecedant must contain at least one of them.

(ANTECEDANT ANAPHORISM MARKERS)

ANTECEDANT locates the antecedant of an anaphorism. MARKERS is an optional argument. When present, it contains a list of semantic markers, one or more of which must be characteristic of the antecedant.

(ANTEQUANT VARIABLE)

ANTEQUANT inserts the quantified antecedant found by ANTECEDANT into the semantic interpretation under construction. If the antecedant is a proper noun, ANTEQUANT acts like QUOTE in the right hand side of a rule, otherwise it acts like QUANT.

(ANTORDER V1 V2)

ANTORDER is a compare function used by SORT in ordering the list of possible antecedants, SORT is called following the interpretation of a request. ANTORDER returns T is V1 should precede V2 in the list, ANTECEDANTS. For example, V1 should precede V2 if V1 is a variable used in the latest request and V2 is not.

(AVERAGE?)

AVERAGE? is a help function that is called by the interpreter when the user refers to "the concentration of X". Since the system knows about many concentrations of X, one for each analysis of X, AVERAGE? asks the user if he means "the average concentration of X". If he does, the system returns the average over the concentrations given in all analyses of X.

(CCHECK TEMPLATE MLIST)

CCHECK is the function which checks semantic conditions in templates during the matching of semantic rules by the interpreter. TEMPLATE is the template in question, and MLIST is a LIST of possible matches for the template which are to be screened by CCHECK. Each element of MLIST is an ALIST whose elements are dotted pairs of node numbers in the template and their corresponding matches in the tree.

(CONSTITUENTS NODE)

CONSTITUENTS is the function which when applied to a node of a parse tree yields a list of the immediate constituents (daughters) of that node. For the tree notation currently in use, this is simply the CDR of the node.

(DEFAULTSEM P TYPEFLAG)

This function is used in the semantic interpretation system to provide the default interpretation T for the restrictions on the range of quantification of a noun phrase, when there are no restrictions implied by the RRULES. It is called by INTERP.

(DOCP X)

DOCP is a predicate that tests whether X is a document according to LSNLIS conventions, i.e. X has the form Dyy-yyy.

(DRULEF P)

DRULEF returns a list of DRULES to try when interpreting the determiner structure on the noun phrase P.

(EQU . ARGS)

EQU is a function for use in the templates of semantic rules. ARGS is a list whose first element is a number. EQU checks whether the terminal string dominated by the node corresponding to this number in the template match is identical with the remainder (CDR) of the list ARGS.

(GETREFS P REFLISTS)

GETREFS is a function used in the semantic interpretation component by the function SORTREFS, which sorts the list of sub nodes to be interpreted into left-to-right order. It serves double duty as a predicate indicating whether P is a node which is to be interpreted, and if so, returning the list of the alternative reflists which belong to that node. For more detail, see SORTREFS.

(GETTAG P TAGNAME)

GETTAG is a function for retrieving items from TAGLIST, a global variable which holds tags associated with nodes in the tree and behaves like a property list for tree nodes. GETTAG returns the value of the tag TAGNAME for the node P.

(HELPER SS COMMENT)

HELPER is an interactive help routine which many semantic inter- preter functions call when they need help from the user. For example, INTERP calls HELPER when it can't interpret a node. RMATCH calls HELPER when a node can have several interpretations though only a single interpretation is allowable (FAIL mode). ANTEQUANT calls HELPER when an antecedant is missing its INTENSION.

(IMPORT VBL)

IMPORT assigns an importance number to a quantified variable. IMPORT is called by ANTORDER which uses it to order the variables according to their likelihood of being referenced anaphorically. The importance of a variable is raised if it has any class restrictions. It is also raised if the class has to be computed, rather than merely being read off a list.

(INTERP P TYPEFLAG)

INTERP is the main function of the semantic interpretation component. It computes the interpretation of the node P "as" or "with respect to" the flag TYPEFLAG. That is, TYPEFLAG is a parameter which tells INTERP how to interpret the node P. For example, to interpret a noun phrase as a set instead of the normal quantification over individuals one can use the typeflag SET. For interpreting normal noun phrases, the three phases of interpreting the determiner structure, the noun itself, and the restrictive modifiers, are indicated by the typeflags NIL, NRULES, and RRULES. The NIL typeflag is the normal interpretation which is assumed if no other typeflag is specified. The typeflag TOPIC is used to indicate the interpretation of a node as a Boolean combination of keyphrases.

INTERP's first action is to determine if the node P has already been interpreted with this typeflag, in which case it recovers the interpretation from the TAGLIST and returns without redoing the interpretation. Also for special TYPEFLAG's HEAD, TERM, AND IDENTITY, where the interpretation does not require the use of semantic rules, INTERP returns immediately with the appropriate interpretation. INTERP returns as its value a list of alternative which consists of pairs of semantic interpretations and governing quantifiers. Each pair , called an SQ-PAIR consists of a semantic interpretation (SEM) which is to be attached to the current node, and a quantifier (QUANT) which is to be passed up to a governing sentence node. When the typeflag is not one of the three special cases listed above, the semantic interpreter called the function RULES with the arguments PAND TYPEFLAG to determine the list of semantic rules to use for the interpretation, and calls MATCHER to perform the matching and return the SEMLIST WHICH IS TO BE THE VALUE. If there is no semantic interpretation, then DEFAULTSEM may supply a default interpretation (currently only in the case of the typeflag RRULES), but if not, then INTERP either return NIL or goes into a break depending on the setting of the flag HELP.

(ISOTOPE X)

ISOTOPE tests whether X is an isotope of some element. If so, it returns the element, otherwise, NIL.

(KEYPHRASE TREELIST)

KEYPHRASE returns a list of the significant terminal nodes of TREELIST. It ignores determiners, auxiliary verbs, tenses, number and non-restrictive modifiers.

(LEAFMEMB X LIST)

LEAFMEMB is a function for determining if any of the "leaves" of the list structure X are members of the list LIST.

(LINEP NODE)

LINEP is a predicate which checks whether NODE has the form (N LINE n), where n is an integer.

(MARKERS WRD)

MARKERS returns the list of semantic markers characteristic of WRD. If WRD is a sample (e.g. S10003), MARKERS returns (SAMPLE). If WRD is a document (e.g. D70-221), MARKERS returns (REFERENCE). Otherwise, it gets the semantic markers off the property list of WRD.

(MATCHER RULELIST P MODE)

MATCHER is the function which matches semantic rules against nodes in the tree. RULELIST is the list of rules to be matched, P is the node to be matched against, and MODE is a flag which indicates what to do with multiple matches. If MODE is AND then multiple matches are ANDed together; if it is OR, then they are OR'ed; if it is SPLIT, then they are split into distinct (semantically ambiguous) interpretations, and if it is FAIL, then multiple matches cause an error. MATCHER accumulates a list SEMLIST of possible interpretations (S-Q pairs), calling the function MATCHGROUP for each (non-null) element of RULELIST. NIL's in RULELIST serve as "barriers" which terminate the testing of rules if a matching interpretation has already been found in the list, but allow the testing to continue if there have been no matches yet. MATCHER calls the user help function NO-MATCHES if none of the rules on RULELIST match and the flag USERFLAG is set to T.

(MATCHGROUP RGROUP)

The elements of the list RULELIST in MATCHER may be either single semantic rules or "groups" OF SEMANTIC RULES WHICH ARE GROUPED TOGETHER WITH A MODE OPERATOR WHICH SPECIFIES HOW SIMUL TANEOUS MATCHES OF DIFFERENT RULES ARE TO BE HANDLES WITHIN THAT GROUP. MATCHGROUP is the function which handles the matching of such a group. If RGROUP is an atom, then it is a rule to be matched; otherwise it is a group whose first element (like MODE) specifies that simultaneously matching rules are to be SPLIT into different interpretations, AND'ed, OR'ed, or cause FAILure. The first element of the RGROUP is saved on CONJ, and all of the rules in the group are tried. The WHILE expression eleminates the results of non-matching rules, and if there are not more than one, then the result of the matching rule is returned. In general, each rule in the group may have returned several distinct interpretations, and the function COMBINATIONS takes all combinations of these. The function SEMCONJ performs the task of combining these interpretations with the operator CONJ.

(MEANING? NPNODE)

MEANING? is a predicate which returns T if the noun phrase NPNODE has been interpreted and the flag USERFLAG is T.

(MEM N MARKER)

MEM is used in the left hand side of semantic rules to check whether a numbered node in a tree fragment belongs in one of the semantic classes in MARKER, MARKER is a list of semantic markers. The node may belong to a semantic class for one of several reasons:

1. the head of the node has the same name as one of the markers. E.g. ROCK belongs to the semantic class ROCK.
2. the head of the node has on its property list one of the semantic markers in MARKER.
3. for a partitive construction, e.g. "Which of the type/A samples" one of the above is true of the head of the prepositional phrase.
4. if the head of the node is a pronoun, one of the above is true of its antecedant.

(NEWCLASS AVAR)

Both NEWCLASS and NEWPX are used in interpreting the anaphoric pronoun "one", as in:

"Which breccias contain aluminum?"

"Which ones contain krypton?" "Ones", in this example, refers to "breccias", not to "breccias which contain aluminum". After the interpreter finds the

antecedant of "ones" (AVAR), NEWCLASS returns the class of AVAR, modified to refer to the current variable QVAR. The class of AVAR is gotten from its INTENSION property. NEWPX returns the class restrictions on the node, those restrictions on AVAR not made by the verb phrase. In the above case, NEWPX would not find any such restrictions. If the first request were

"Which breccias that are over 400 million years old contain aluminum?" NEWPX would return the restrictions associated with the phrase "that are over 400 million years old".

(NEWFRAG NAME FRAG)

NEWFRAG is used to update the list of tree fragments TREEFRAGS. It sets NAME to FRAG and then adds it to TREEFRAGS. For example,

(NEWFRAG S.OBJ.NPR (S ((VP ((NP ((NPR NIL 1))))))))).

(NEWPX AVAR)

See NEWCLASS.

(NEWRULE ARGS)

NEWRULE adds new semantic rules to the system and indexes them properly. (CAR ARGS) is the name of the rule and (CDR ARGS) is its value.

(NO-MATCHES)

NO-MATCHES is called by MATCHER if nonee of the rules which might be used to interpret the current node match it. NO-MATCHES gives the user the choice of quitting or breaking. If he breaks, he can fiddle with the semantic rules to see why non matched, then reset the variable RULELIST. Upon his return from the break, MATCHER will be re-run on the current node and the new set of rules in RULELIST. The MATCHER - NO-MATCHES cycle can be repeated by the user indefinitely many times.

(NXTVAR)

NXTVAR is the function which gets the next available variable name for use in the quantifiers during the semantic interpretation. It uses variables cyclically from a list called VARIABLES.

(ODDP N)

ODDP is a predicate which test if N is an odd integer.

(ORMATCH TEMPLIST)

ORMATCH is a routine for matching OR'ed templates in RMATCH. That is, when in place of a single template in a semantic rule, there is an OR of several templates, then ORMATCH is called to perform the matching of all of them. It also provides for a standard DEFAULT interpretation as the last component of an OR. It will take the default interpretation if and only if there are no other matching templates in the OR.

(PRED SEMFORM)

PRED is one of three functions (PRED, QUANT, and SSUNIONF) which are used in the right-hand sides of semantic rules to indicate what is to happen to quantifiers.

QUANT indicates that the right-hand side is a quantifier that is to be passed up to a higher constituent, with the semantic interpretation of the current node being the variable assigned to that quantifier.

PRED indicates that the right-hand side is a predicate which is to "grab" any quantifiers passed up by constituents -- that is, any such quanthfiers will be treated as quantifying the expression SEMFORM which is the argument to PRED.

SSUNIONF indicates that the right-hand side is a successor function which is to "grab" some quantifiers, but wrap others tightly around itself. The interpretation of the current node is then one big successor function over the sets given by the quantifiers and the original successor function. SSUNIONF is used to distinguish the scope of "each" which is usually a maximum, from those of "every" and "all" which generally follow left to right order. For example, the right-hand side of N:ANALYSIS is (SSUNIONF (DATALINE (WHQFILE (# 3 2 SSET))(# 3 2 SSET)(# 4 2)(# 5 2 SSET))). The interpretation of the request "Howmany analyses of Krypton are ther for all samples?" is

```
(FOR THE X9 / (SEQL (NUMBER X1Ø / (SSUNION X8 / (SEQ
SAMPLES) : T ; (DATALINE (WHQFILE X8) X8 OVERALL KR)
: T)) : T ; (PRINTOUT X9)).
```

There is a single answer for the entire set of samples. The interpretation of "Howmany analyses of Krypton are there for each sample?" is however:

```
(FOR EVERY X8 / (SEQ SAMPLES) : T ; (FOR THE X9 /
(SEQL (NDMBER X1Ø / (DATALINE (WHQFILE X8) X8
OVERALL KR) : T )) : T ; (PRINTOUT X9))).
```

There is one answer for each sample in the set. (The

typeflag SSET indicates that if the node can be interpreted as a set, it should be. "All sample" can be interpreted as a set, "Each sample" can not be.)

(QUANT SEMFORM)

QUANT is a function used in the right-hand side of semantic rules to indicate that SEMFORM is to be interpreted as a quantifier. (See PRED.)

(QUIT)

QUIT effects a quick return to TALKER from wherever one is processing in the upper fork.

(REFLOC RHSFRAG RVECTOR)

REFLOC is a function used by SEMSUB in the semantic interpreter to make up REFLISTS for a given right-hand side of a rule and a given vector of matches (RVECTOR). If RHSFRAG (a fragment of the right-hand side of the rule) is a REF (i.e., an expression which refers to the semantic interpretation of some constituent of the node being interpreted), then REFLOC returns the REFLIST for that constituent. Otherwise it scans RHSFRAG for instances of REF's. The REFLIST which is returned consists of the pointer (SUBP) to the constituent in the tree to which the REF refers (for this particular match specified by RVECTOR), the REF itself (RHSFRAG), and the interpretation of the node SUBP using the typeflag specified by the REF.

(REFP RHSFRAG)

REFP is a predicate which tests a fragment of a right- hand side of a semantic rule to determine whether it is a REF (i.e., whether it refers to a constituent of the tree being interpreted whose semantic interpretation is to be used as a part of the current interpretation). This is true if the frag- ment is either a dotted pair of integers, or a list beingging with the atom #.

(REFPTR RHSFRAG)

REFPTR is like REFP, except that it also returns a pointer to the node in the tree to which the REF refers (the matching pointer being obtained from the current RVECTOR).

(REFQUANTS REFVECTOR)

REFQUANTS is a function which gathers the quantifiers from all of the REFLIST's in REFVECTOR into a

single quantifier "collar" which is to be wrapped around the expression which it governs. It assumes that the REFLIST's on REFVECTOR have been sorted into the order in which they are to occur. This sorting is accomplished by the function SORTREFS in a call from SEMSUB. REFQUANTS also separates the type of quantifier arising from "each", from those arising from "every" and "all". It does this so that "each" will nave a maximum scope, independent of its surface structure location. "Each" will produce a quantifier of the form

(FOR EVERY X / ...)

while the others will produce one of the form

(SSUNION X / ...).

(REFSUB REFVECTOR)

REFSUB is a function which takes the current value of RHS (maintained by SEMSUB), substitutes the semantic interpre- tations of its REF's, and evaluates the result to obtain the semantic interpretation of the current node. REFSUB1 actually performs the substitution, and prior to the execution of the substituted right-hand side, REFQUANTSis used to construct the appropriate quantifier "collar". The call to EVAL will result in these quanti- fiers being "grabbed" and wrapped around the semantic interpretation of the current node if the right-hand side of the rule (RHS) is embedded in a PRED; if it is embedded in a QUANT, then the call to EVAL will result in the quantifier being inserted into the "hole" of the collar (substituted for DLT) and the semantic interpretation of the current node will be set to the variable name associated with the quantifier.

(REFSUB1 RHSFRAG REFVECTOR)

REFSUB1 performs the substitutions in the RHS of a semantic rule before it is evaluated by REFSUB. It substitutes the current value of the variable (QVAR) for occurrences of the atom "X" wnen interpreting restrictions on the range of quantifi- cation in interpreting noun phrases, and substitutes the semantic interpretations for REF's.

(REFSUB2 RHSFRAG REFVECTOR)

REFSUB2 is used by REFSUB1 to walk across a sublist of a RHS and spply REFSUB1 recursively.

(REFTYPE REF)

REFTYPE is a function for extracting the reftype of a REF -- i.e. the typeflag that is to be used for interpreting the node to which the REF refers. For dotted pairs, the REFTYPE is NIL, while for REF's that begin with #, the reftype is the element of the list which follows the numbers that denote the node to be interpreted.

(RELTAG PLIST)

RELTAG is the function used by the semantic interpreter for locating the relative pronoun of a relative clause to be inter- preted and tagging that node with the variable of quantification (QVAR) associated with the noun phrase which the relative clause modifies.

(RMATCH RULE P MODE)

RMATCH is the basic semantic rule matching function. It matches the single semantic rule RULE against the node P with mode MODE, (unless MODE is reset by the first element of the rule itself). It calls TEMPMATCH to match each of the templates of the rule or ORMATCH to match OR'ed groups of templates, and if a successful match is found it calls SEMSUB for each possible way in which the rule can match. If there are multiple matches, then it combines them in the way indicated by MODE.

(RULES P TYPEFLAG MODE)

RULES is used by INTERP to furnish the list of rules to use in interpreting the node P, according to TYPEFLAG. TYPEFLAG specifies what kind of interpretation is required. The following kinds of interpretation are recognized by RULES:

1. ALL - for partitive constructions determined by "all".
2. SET, SSET?, SSET - for nodes to be interpreted as sets.
3. AVERAGE, MAXIMMM, MINIMUM, NUMBER, OLDEST - for partitive constructions headed by one of these words.
4. REFS? - for nodes to be interpreted as requests or topics.
5. REFS - for nodes to be interpreted as topics, if possible.
6. S - for sentence nodes.
7. TOPIC - for nodes to be interpreted as topics.
8. NP - for interpreting the determiner structure on noun phrases.
9. NRULES - for interpreting the class of noun phrases.
10. RRULES - for interpreting the restrictions on noun phrases.

11. SRULES - for interpreting the main verb of a sentence. RULES sets the MODE of interpretation which is then used by RMATCH. RULES calls DRULEF for the list of DRULES to use in interpreting the determiner structure on a noun phrase.

(SAMPLEP X)

SAMPLEP is a predicate which tests if X has the form of a sample according to LSNLIS conventions, that is, X is of the form Syyyyy.

(SCOPEFINDER FORM CONTEXT)

SCOPEFINDER makes intension and scopevars entries on the property lists of variables in quantifiers.

(SCOPEVARS X)

SCOPEVARS accumulates the closure of the scopevars of a variable.

(SEMCONJ CONJ SEMLIST)

SEMCONJ is the function which combines multiple semantic interpretations with the conjunction CONJ. It conjoins the SEM's of the interpretations under the conjunction CONJ, and produces a quantifier which is the nexting of all of the quantifiers which are associated with the individual interpretations.

(SEMIANAPHOR ANAPHOR)

SEMIANAPHOR is used to resolve one type of partial anaphora: a pronoun modified by a prepositional phrase. For example, "Give me analyses for krypton in breccias." "Give me those for magnesium" The antecedant of "those for magnesium" is "analyses for magnesium in breccias". SEMIANAPHOR finds the node dominating "analyses for krypton in breccias" as the partial antecedant of "those", replaces "for krypton" with "for magnesium", and calls for the reinterpre- tation of the new node.

(SEMNET N1 N2)

SEMNET test whether N1 and N2 are semantically similar, e.g. they share semantic markers or they are both samples or documents. SEMNET, at present, is only a bare attempt at doing the sort of things that could be done with a semantic network.

(SEMSUB RHS RVECTOR)

SEMSUB is the function which substitutes semantic inter- pretations for their REF's in the right-hand sides of semantic rules. It is the major dispatcher among the functions REFLOC, SORTREFS, and REFSUB.

(SORTREFS REFLISTS P)

SORTREFS is the function which sorts REFLISTS into the order in which the quantifiers associated with the REF's are to be incorporated into the interpretation -- namely in order of their left-to-right position in the structure P. This is accomplished by the function SORTREFS1 which walks the tree P and adds REF's to the list in the order in which it sees them.

(SORTREFS1 REFLISTS P)

See SORTREFS.

(SPROC P)

SPROC is called on the output of the Parser, P. It clears TAGLIST before beginning and returns a list of possible semantic interpretations of P. SPROC also calls for the reordering of the antecedant list ANTECEDANTS following the interpretation of P.

(SSUNIONF SEMFORM)

SSUNIONF is a function used in the right-hand side of semantic rules to indicate that SEMFORM is a successor function. (See PRED.)

(SUPERLATIVE N)

SUPERLATIVE tests whether the head of the node numbered N is a superlative adjective. A superlative adjective with a definite determiner is parsed as a noun, while its associated surface structure noun is made the head of a dependant partitive construction. For example, "the oldest sample" is parsed as if it were "the oldest of the samples". MEM calls SUPERLATIVE on a node to see whether its semantic properties should be gotten off the head noun or the head of a partitive construction.

(SYNONYMS? HEAD TYPEFLAG)

SYNONYMS? is a help function called by RULES when it cannot find any semantic rules to use in interpreting the head noun or head verb of a node. It tells the user it cannot understand the word and asks if it is a synonym of one of the words it knows. If it is, the system will get the semantic rules off the synonym and continue.

(TAG P TAGNAME VALUE)

TAG is the function which places tags on the TAGLIST. It associates with the node P the property TAGNAME with the value VALUE.

(TEMPMATCH TEMPLATE P)

TEMPMATCH is the semantic interpretation function which matches templates with nodes of the tree (it is called by RMATCH and ORMATCH). It calls the functions TMATCH to perform the tree matching of the tree fragment with the node P, and CCHECK to check the semantic conditions of the template for any resulting tree matches. It also provides the results of a simulated match in the case of a DEFAULT template.

(TERM TREELIST)

TERM is a function which returns the list of "leaves" or "terminal nodes" of a list of tree structure nodes (TREELIST). It does so by walking the tree structure and gathering up the "leaves".

(TMATCH PLIST FLIST)

TMATCH is the function which performs the subtree matching for the semantic interpreter (called by TEMPMATCH). PLIST is a list of nodes on the tree which are to be matched against the fragment nodes (from the semantic rules) in the list FLIST. It returns a list of all possible matches--each match being represented by a vector (ALIST) of correspondences between numbered nodes in the tree fragment and the nodes in the tree being interpreted.

(USED? ADJ)

USED? is a predicate which tests whether the adjective ADJ has been tagged by MATCHER as being used in the interpretation of some node.

III. RETRIEVAL FUNCTIONS

(ABOUT DOCUMENT TOPIC)

ABOUT is a predicate for a DOCUMENT being indexed under one or more of the keyphrases in TOPIC. For example, the interpretation

```
(FOR EVERY X7 / DOCUMENT : (ABOUT X7 (OR (
 FERROUS IRON)
(AND (FERROUS)(IRON)))) ; (PRINTOUT X7))
```

retrieves all documents which have been indexed under "FERROUS IRON" or under both "FERROUS" and "IRON".

(AGE ARGS)

AGE can take two or three arguments, a sample number, a radiometric clock, and a restart pointer. If the clock is specified, AGE returns the age of the sample as may have been measured by that radiometric clock. Otherwise, AGE returns the age of the sample according to each of the radiometric clocks in AGELTS, for which such measurements have been made. AGE calls DATALINE to find all the necessary age analyses. For example, (AGE (NPR* X3 / (QUOTE S10071))(QUOTE *PB207) INDEX) will return the first age analysis of sample 10071 by lead isotope dating if INDEX is NIL, subsequent analyses otnerwise.

(ANALYSES FN GAZ)

ANALYSES is a successor function like DATALINE, which returns, one by one, all the chemical analyses in all the files on FILEDIRECTORY. FN and GAZ are both restart pointers, the first to the remaining analyses in the current analysis file, tne second to the remaining files in FILEDIRECTORY. It is not currently used, but could be, instead of DATALINE, in situations where it would be more efficient. For example, in response to "Give me all the analyses in your files."

(AROUNDVAL Q1 Q2)

AROUNDVAL is a predicate which tests whether Q1 is within an engineer's approximation of Q2. That is, .9*Q2 < Q1 < 1.1*Q2. Q1 and Q2 may be scalar quantities, number-unit pairs or analyses. In the latter case, the value of the analysis is fetched from the appropriate file.

(ASSOCNEXT LISTV NTRY)

ASSOCNEXT returns the tail of LISTV whose CAAR is equal to NTRY. LISTV is a list of lists. ASSOCNEXT differs from ASSOC in returning the whole tail and not just the head of the tail. For example,

```
(ASSOCNEXT (QUOTE CPX)(QUOTE ((OVERALL . 2877)(CPX .
 2900) (PLAG .
2958)(*** . 3017))))
((CPX . 2900)(PLAG . 2958)(*** . 3017))
```

ASSOCNEXT is called by DATALINE.

(AVERAGE *X* / CLASS : PX)

AVERAGE calculates the numerical average of the members of CLASS which meet the requirements stated in PX. AVERAGE is used to compute average ages, average concentrations and average ratios. For example, "What is the average potassium / rubidium ratio in low-alkali rocks?" is interpreted as

```
(FOR THE X1 / (SEQL (AVERAGE X2 / (SSUNION X3 / (SEQ
 TYPEAS) :
T ; (RATIO ( QUOTE K2Ø) (QUOTE RB) X3 (QUOTE
 OVERALL))) : T))
:T : (PRINTOUT X1)).
```

(AVGSTEP X1)

AVGSTEP is used by AVERAGE to compute average analyses (concentrations). It fetches the value of the current analysis X1, does any unit conversion necessary, and increments the accumulator with the value.

(BOOLGET X)

When the quantifier function FOR is quantifying over a set of documents (i.e. the CLASS in "DOCUMENT"), it calls the function BOOLGET with X set to the restriction on the range of qualification (i.e. the PX term in the quantification). BOOLGET searches the expression X for all instances of the predicate ABOUT (which represents the prediction of a document being about a topic) and gathers up the corresponding Boolean combination of topics. This is used as an argument to BOOLRET (after being converted to conjunctive normal form) and is used to enumerate the appropriate set of documents by performing the Boolean operations on the inverted file lists for the keyphrases of the topics.

(BOOLREQ CNF)

BOOLREQ converts a normal conjunctive normal form Boolean expression CNF into a modified form suitable for BOOLRET, by sorting negations to the end and raising negations if necessary so that they are always components

of an AND and not of an OR. The latter is done in order to provide a Boolean request which can always be done by intersecting inverted files and never requires constructing the complement of an inverted file. BOOLREQ1 performs the bulk of this operation, NEGSORT sorts the NOT's to the ends of clauses, and converts such clauses to instances of the operator SDIFF (which represents the operation of taking the set difference between two Boolean expressions).

(BOOLRET BOOLEXP)

BOOLRET is the function which performs the Boolean operations indicated in BOOLEXP on the inverted file lists of documents associated with the key phrases in BOOLEXP. It returns a list of all documents which satisfy the Boolean expression.

(BUILDCA CA FLDNS)

BUILDCA is used when building the lower fork to set up the codearrays and do the appropriate coding, FLDNS is a list of field names in the same order as their corresponding code array in CA. Each member of FLDNS is bound to a list of its field values to be coded.

(CHANGELINE ARGS)

CHANGELINE changes a single field on each line meeting the specifications given in ARGS to a value also specified in ARGS. The format of ARGS is <fieldname> <new-value> <file> <old-spec> *, where file meeting the specifications in <old-spec>*. The order of specifications must match the order of fields in a file record. For example, (CHANGELINE ELT AL2Ø3 APOLLO11 S1ØØ84 OVERALL AL213) will change the value of every AL2Ø3 analysis of SJØØ84 OVERALLwhich has been incorrectly specified as an A1213 analysis. To change the value of more than one field on a single line, one should use EDITLINE instead.

(CHANGE1LINE FN FIELD VALUE)

CHANGE1LINE is called by CHANGELINE after CHANGELINE has found a line that meets the given specifications. If the file is not sorted on FIELD, CHANG1LINE changes the current value of FIELD to VALUE. Otherwise, CHANGE1LINE deletes the line and inserts the corrected line at the end of the corresponding patch file. The patch file has the same name as the main file, with the extension PATCH, e.g. APOLLO11 and APOLLO11.PATCH. The argument FN is (FILE . LINE-NUMBER).

(CLEARMAP)

CLEARMAP is called by RETRIEVER to remap file pages and reset the GAZETTEER after a request has been serviced. In case the user inadvertently leaves the lower fork via an interrupt, he can type EXECUTE() to run CLEARMAP.

(CNF BOOLESP)

CNF is the function which converts Boolean expressions to conjunctive normal form. The result of CNF is a list of lists of keyphrases or their negations. Each element of the top level list is a disjunction (OR) of its contained phrases, while the top level list is a conjunction (AND) of these disjunctions, although the AND's and OR's are implicit in the list structure and do not explicitly appear. BOOLREQ uses the output of CNF to construct its Boolean request.

CODE VAL ARY)

CODE replaces the CODE and ENCODE functions of the previous LSNLIS system. CODE examines the CODES property on VAL for the occurrance of ARY, a code array in which VAL has presumably been coded.

(COMBINATIONS LIST)

computes the cross product of a list of lists. For example, COMBINATIONS (((A B C))) = (B) (C)) COMBINATIONS (((A B)(C D))) = ((A C)(B C)(A D)(B D)) COMBINATIONS (((A B)(C)(D E))) = ((A C D)(B C D)(A C E)(A C E))

(COMPLEMENT LITERAL)

This function returns the complement of the literal key-phrase LITERAL. If LITERAL is negated, then COMPLEMENT drops the negation; otherwise, it adds a negation. The function is used in BOOLREQ1.

(CONTAIN *X* *Y* *Z*)

CONTAIN tests whether sample *X* contains the phrase or element *Y* (if *Z* is NIL) or whether it contains element *Y* in phase *Z* (if *Z* in non-NIL). If only an element is specified, CONTAIN checks the ELTS property on *X* for *Y*. If *Y* is a phase, CONTAIN checks the index on *X* since files are indexed on sample-phase pairs. If both *Y* and *Z* are given, CONTAIN issues a call to DATALINE.

(CONTAIN' SAMPLE ELPH ARG1 ARG2)

CONTAIN' is used to answer "how much" questions and to compare the average concentration of some mineral or element in a sample against some given amount, Like CONTAIN, CONTAIN' may be called with a mineral or an element or an element within some phase of SAMPLE as arguments. If a phase is specified, CONTAIN' takes four arguments; otherwise, it takes three. The last argument to CONTAIN' is either (HOW), for a "how much" question, or a test (E.G.(MORETHAN 5 PPM), (ATLEAST 3.5 PCT)). In the first case, CONTAIN' returns the average concentration, in the second, T or NIL, depending on whether the average concentration passes or fails the test. For example, "Which rocks have greater than 50 PPM Nickel?" is interpreted as:

```
(FOR EVERY X3 / (SEQ VOLCANICS) : (CONTAIN' X3
 (QUOTE NIO)
(GREATERTHAN 50 PPM)) ; (PRINTOUT X3))
```

(CONVERT Q1.U1 U2)

is a neat call to CONVERTU.

(CONVERTN N U1 U2)

sets up a call to CONVERTU on (N . U1) and U2.

(CONVERTU NUP UNIT DUMMY1 DUMMY2)

CONVERTU converts the number-unit pair NUP to the unit specified in UNIT. DUMMY1 and DUMMY2 are dummy variables. CONVERTU is called by CONVERT, CONVERTN and UQUOTIENT.

(DATALINE ARGS)

ARGS is a list with the format <file><fldspec>* <index>. DATALINE is a successor function which searches a file for lines which meet the specifications given in fldspec *. Any number of fields in a file re may be specified, but they must follow the order of the fields in a file record and be non-NIL. The format of index , the restart pointer, is ((<file> . <line#>) , filetop). DATALINE does a binary search on sorted fields, and a linear search on non-sorted fields. It takes advantage of whatever indexing has been done on the file, recorded on the property list of the primary sort key. For example, (DATALINE APOLLO11 S10056 OVERALL AL203 INDEX), if INDEX is NIL, will return a pointer to the first OVERALL analysis of samples 10056 for AL203, in this case ((APOLLO11 . 7763) . 13248). If DATALINE is called again for the next line meeting this description, it will

be called with INDEX set to ((APOLLO11 . 7763) . 13248), the answer returned above.

(DECODENUM ARRAY CODE)

DECODENUM returns the decoded value of CODE according to the code array ARRAY. For example, DECODENUM (MARR11 27) returns SILICA.

(DELETELINE ARGS)

ARGS is a list with the format file fldspec *. DELETELINE will delete all lines in file which meet the description given in fldspec *. For example, (DELETELINE APPOLLO11 S1ØØØ3 CPX FEO) will delete all CPX analyses of sample 1ØØØ3 for FEO, if there are any, in the APOLLO11 file.

(DELETE1LINE FN)

FN is the dotted pair (<file> . <line#>). DELETE1LINE inserts a deletion indicator, 1ØØØ, in the primary sort field of the line indicated in FN. DELETE1LINE is called by CHANGE1LINE, DELETELINE and EDITLINE.

(DO X)

calls for the evaluation of X. DO IS THE INTERPRETER"S RESPONSE TO AN IMPERATIVE QUESTION.

(DOCP X)

DOCP is a predicate for X being a document (citation) number, i.e. having the form Dnn-nnn, where n is any integer.

(DOCUMENT INDEX)

DOCUMENT is a successor function which, one by one, returns the document numbers known to the system.

(EDITLINE FILE N)

EDITLINE allows the user to make any number of changes to line N of file FILE. If the user has changed the value of a field on which the file is sorted, EDITLINE makes a new entry corresponding to the changed line in the appropriate patch file. If an altered field is not a sort key, then the new value just replaces the old one in the original file.

(ELT:LINE FN)

FN is the dotted pair (<file> . <line#>). ELT:LINE computes the entry in MAPARRAY corresponding to the given file-line number pair. ELT:LINE also sets the value of FORMAT, the file format description, for its calling functions FETCH, FETCHLINE, STORELINE and STOREVAL.

(EXECUTE FORM)

EXECUTE is the function which performs the execution of query language expressions. It may be either of two distinct functions LOCEX or REMEX which perform the execution in the same (local) fork or in f a remote fork, respectively. In either case, EXECUTE provides for the opening of a file HITFILE in which the answer is to be recorded and the maintenance of a counter COUNTwhich is incremented by the functions which write information onto HITFILE. When the execution is completed, EXECUTE closes HITFILE, and if COUNT is not greater than 5 copies the HITFILE to the teletype as the answer. If COUNT is greater than 5, then EXECUTE types a message giving the number of hits and asking the user whether he wants to see them on the TELETYPE. (If not, he has the option of listing HITFILE offline or saving its value for later listing.)

In the case of REMEX, which is the way that the LSNLIS system is currently running, the above procedure is additionally complicated by the fact that the retrieval component in which the query language expression is executed resides in a complete separate fork of the TENEX system. In this case, EXECUTEw writes the query language expression into a buffer file QBUF, and calls the LISP function RUNFORK to wak up the retrieval fork. The retrieval fork then reads the expression from QBUF and executes it as discussed above, with the answers being written onto HITFILE. When the execution is completed, the retrieval fork writes the value of COUNT into another file buffer ABUF, and returns control to the language processing fork. At this point, EXECUTE regains control, reads the value of COUNT FROM ABUF, and proceeds to type the answers or notify the user of the number of hits as above.

(EXIST X)

EXIST is a predicate which is universally true of every argument.

(EXPANDARRAY ARNAME)

EXPANDARRAY expands a code array if it runs out of space by recopying it into a larger array. EXPANDARRAY is called by CODE.

(FETCH FN FIELD)

FETCH returns the value in the field FIELD for a given file- line number pair FN. FETCH saves the value of FN in the global variable OLDFN, so that if the value of FN doesn't change on subsequent calls, FETCH will not have to call ELT:LINE to compute a new pointer into MAPARRAY. The value of the pointer is saved in OLDELT, and the value of the file format in OLDFORMAT.

(FETCHLINE FN)

FETCHLINE returns a list of the binary words making up the record for the given file-line number pair FN. FETCHLINE is used by LINEORDP to save time, since alphanumeric order is matched in the APOLLO11 and APOLLO11.PATCH files by binary order.

(FETCHVAL FN FIELD)

FETCHVAL returns the value in FIELD for the file-line number pair FN. If the value is encoded, FETCHVAL calls CODE to decode it.

(FLTBOX LOC)

FLTBOX returns the boxed value of the quantity in memory location LOC. FLTBOX is called by FETCH when the field type of the field to be retrieved is FLT.

(FOR ARGS)

FOR performs quantification in the retrieval component. FOR has been made an NLAMBDA so that the user can specify a new quantifier in case the one produced by the interpretive component is incorrect. This will happen often when there are many values of *X* for which a statement is true and the user has requested "the" value, meaning the one.

(GETDOCS PHRASE)

GETDOCS returns the inverted file indexed on PHRASE from the external file PHRASETABLE. It is called by BOOLRET.

(GETPAGE FILE PAGE)

GETPAGE determines whether page PAGE of file FILE has been mapped onto MAPARRAY. If it has not, GETPAGE calls PAGEMAP to do the mapping. The record of which pages of which files are currently mapped onto MAPARRAY is kept on the list GAZETTEER.

(GREATER Q1 Q2)

GREATER is a predicate which tests whether Q1 is greater than Q2. Q1 and Q2 may be scalar quantities, number-unit pairs, or analyses. If the latter, the value of the analysis is fetched from the appropriate file.

(INDEXF FILE)

INDEXF indexes all sample-phase combinations in a coded chemical analysis file FILE, printing the size of each sample-phase block for later decision on deeper indexing. The index is sectioned off by sample number, and each section is maintained on the property list of the corresponding sample, under the property APOLLO11.

(INTERSECT LIST)

takes the intersection of all the lists on LIST. For example, INTERSECT (((A C D F)(C F) (B F H C))) = (C F)

(LESSVAL Q1 Q2)

LESSVAL is a predicate which tests whether Q1 is less than Q2. In all other ways, it resembles GREATER.

(LINEORDP FN1 FN2)

Both FN1 and FN2 are of the form (<file> , <line#>). LINEORDP returns T if the first line precedes the second in alphanumeric order. This version of LINEORDP takes advantage of the fact that a chemical analysis file is sorted on all fields and only fields in the first two words of each record. Given this, alphanumeric order corresponds to binary order on the first two words of the record. Thus, LINEORDP can use LESSP on the first two words of the record, rather than consecutive, time-consuming checks on the sorted fields.

(LINE#OF X)

LINE#OF is intended to return the line number of X. In the current system, it is assumed that X is a line number, and LINE OF returns X.

(LOADLOW LOADFILES)

LOADLOW is used when building the lower fork to load in all the necessary functions and global variables. It also sets up the array and values needed for page mapping.

(LOCEX FORM)

LOCEX would be the executive program for executing semantic interpretations in undivided retrieval system. See EXECUTE for further details.

(MATCHLINES ARGS)

ARGS has the format <FILE><LINE ><FLDSPEC>*. FLDSPEC may have a value or be NIL, but there must be a FLDSPEC for each field in the file record. MATCHLINES tests whether the given line of the given file matches the field specifications. MATCHLINES returns WORTHLESS if the given line would follow a line with these specifications in the file, ALMOST if the given line would proceed it, and OK if the lines match. A match occurs if the specification is NIL or if the value of the specification is the same as the value of the field on the given line.

(MAXIMUM X / SAMPLESET : PX)

MAXIMUM calculates the maximum value over those members of SAMPLESET which meet the conditions specified in PX.

(MEMBER* INDIV CLASS)

MEMBER* tests whether INDIV is a member of CLASS. It differs from MEMBER in that CLASS may be a variable bound to an atom whose value is a list, or a list itself. Otherwise, MEMBER* returns an error message. MEMBER* calls MEMBER.

(MINREAD INPUTFILE OUTPUTFILE)

MINREAD is a function to create a binary chemical analysis file for the new data base from a LISP written file for the old one. OUTPUTFILE must have been opened for I/O prior to the call to MINREAD.

(NEGSORT CLAUSE)

NEGSORT is a function used in the Boolean request generation to move negated phrases to the end and convert them to calls to SDIFF, the function which indicates the

set difference between two Boolean descriptions. See BOOLREQ for more details.

(NEWLINE ARGS)

ARGS has the format <file> <fldspec> *. NEWLINE sets up successive calls to STOREVAL to store the record specified in fldspec * in the next empty record of file . NEWLINE updates the property TOP on file to one higher than its previous value, and this value is returned by NEWLINE. For example,

```
(NEWLINE APOLLO11.PATCH S10022 OVERALL AL203 D70-194
 0 PCT 7.1)
6
```

The order of <fldspec> * must follow the order of fields in a record. NEWLINE is called by EDITLINE and MINREAD. NEWLINE should be used otherwise only on unsorted files.

(NEXNUM INDEX)

NEXNUM finds the leftmost terminal element which is also a number in the tree INDEX. NEXNUM is used by DATALINE in searching the file index on a sample for the start of the next sample-phase block.

(NPR ARGS)

NPS is a variant of QUOTE. If ARGS have a special significance in the system, indicating a line number, a citation, a specimen, a phase, or a constituent, NPR returns the standard form. Otherwise, NPR acts like QUOTE. For example,

```
(NPR SAMPLE 10003)
S10003
(NPR LINE NUMBER 10)
10
(NPR BERYLLIUM)
BE
```

(NPR* *X* / CLASS)

NPR* sets the value of *X* to the value of CLASS. It is used to associate a scope variable with a proper noun for referencing purposes. It is produced by the interpretive component for all proper nouns. For example, (NPR* X7 / (QUOTE S10056)).

(NUMBER ARGS)

counts the number of times the form ARGS IS TRUE. NUMBER returns a one-place list containing the count, and is used to answer "how many" questions.

(OLDEST *X* / SAMPLESET : PX)

OLDEST calculates the average age of each sample in SAMPLESET which meets the conditions specified in PX, and returns that sample which has the greatest average age. OLDEST records its answer in the list DEFDESC, so that the oldest member of SAMPLESET need only be calculated once.

(OUTPUTLINE FN)

FN has the format (<file> . <line#>). OUTPUTLINE is in general printing function for file records which gets the information about the record format and printing format from the property list of the file name and prints out the record accordingly. OUTPUTLINE is called by PRINTOUT and PRINTFILE.

(PAGEMAP MEMPAGE FILE FILEPAGE)

PAGEMAP maps page FILEPAGE of FILE onto page MEMPAGE in the lower fork.

(PHRASEMATCH PHRASE PHILE POS)

PHRASEMATCH tests whether the inverted file indexed on PHRASE starts at position POS of the file PHILE, or would start earlier in the file or later. PHRASEMATCH is called by GETDOCS to determine the next move in its binary search.

(PRINTFILE FILE)

PRINTFILE takes the place of the functions POLD and PNEW in the former LSNLIS system, to print, in readable format, the binary files in the data base.

(PRINTLINE N)

The values to be printed by PRINTLINE are the values in PRINTFIELDS. The values printed on the prior line by PRINTLINE are stored in OLDPRINTFLDS. The printing of a field is suppressed if its value is the same as the former value, unless the field is a floating point number, which always prints. N is the number of fields to be printed, and is present to make PRINTLINE more general. In the current system, all binary files have the same number of fields and it corresponds to the length of PRINTFIELDS.

(PRINTOUT *X*)

PRINTOUT prints out the answers to requests made to the retrieval component and increments the variable COUNT maintained by RETRIEVER. PRINTOUT HAS SEVERAL OPTIONS. If *X* represents the result of a DATALINE computation, PRINTOUT prints out the associated analysis, If *X* is within the scope of any other variables, PRINTOUT prints out the values of those variables in addition, If *X* has any additional information associated with it, produced during the evaluation of the request, that additional information is printed out along with the answer. If *X* is not a variable, PRINTOUT merely prints out its value.

(RATIO ELT1 ELT2 SN MIN POINT)

RATIO is a successor function which returns, one by one, the ratios it computes of ELT1 to ELT2 in the MIN phase of sample SN. POINT is the restart pointer. Ratios are only computed between analyses with the same reference - i.e. done by the same set of authors.

(REF* FN REFR)

REF* is a predicate which tests whether the reference associated with the analysis FN is equal to REF. FN has the format (<file> . <line#>).

(REFILEPHRASES INFILE OUTFILE)

REFILEPHRASES converts the set of inverted files on INFILE into one, OUTFILE, suitable for binary search. Inverted files on the same keyphrase are merged, and each inverted file is bounded on both ends by square brackets.

(REMAP MEMPAGE)

REMAP remaps page MEMPAGE of the lower fork onto its associated filepage.

(RETRIEVER)

RETRIEVER sets up the lower fork of the two fork LSNLIS system. It exits to the Exec with HALTFN so that the lower fork can be saved. When the lower fork is called by REMEX, it is entered in RETRIEVER following the call to (HALTFN).

(SAMPLEP X)

SAMPLEP is a predicate for X being a sample number - a string beginning with "S" and followed by a five-digit number.

(SCOPEFINDER FORM CONTEXT)

makes INTENSION and SCOPEVARS entries on the property lists of the variables in quantifiers. SCOPEFINDER is used in determining correct anaphoric referents.

(SEQ *L* *I*)

SEQ is a successor function which enumerates the members of a list. *I* is its restart pointer. SEQ returns (CDR (EVAL *I*)) if *I* is non-NIL. Otherwise, SEQ returns (EVAL *L*). This value is then used by SEQ's calling function as its restart pointer. The variable AGAINFLG prevents SEQ from restarting *L* after *I* is NIL.

(SEQL *L* *I*)

SEQL is equivalent to (SEQ (LIST...)), It is produced in the semantic component for single-membered sets.

(SETLIST *X* / CLASS : PX; QX)

SETLIST returns a list of the members of CLASS which satisfy condition PX, if QX = T. Otherwise, SETLIST returns QX applied to each of those members. SETLIST is similar to SETOF; however, it returns a list of its answers, rather than each one in turn.

(SETOF *X* / CLASS : PX ; QX POINT)

SETOF is a successor function which returns the members of the class CLASS which satisfy PX, after having applied QX to them. POINT is the restart pointer.

(SORTNEW FILE)

SORTNEW does a bubble sort of the unsorted portion of FILE into the sorted portion. The boundaries of the unsorted portion are the values of SORTTOP and TOP, both on the property list of FILE. SORTTOP is reset to TOP after the file is sorted. SORTNEW is called by DATALINE.

(SSUNION *X* / CLASS : PX ; QX INDEXO)

SSUNION is a successor function. For each member of CLASS meeting the conditions in PX, QX is evaluated. QX is either T or another successor function. INDEXO is a restart pointer for both CLASS and QX.

(STORELINE FILE LINE# LINE)

STORELINE inserts the binary record given in LINE on line LINE of FILE. It is used by CHANGELLINE to save time, instead of a series of STOREVALS, CHANGELLINE gets the binary record with a call to FETCHLINE.

(SUNION ARGS)

SUNION is a successor function which, one by one, enumerates all the members of all the sets listed in ARGS.

(TABFORM X)

returns the standard form of X, as it appears in the mineral analysis data base.

(TABFORM ORTHOPYROXENE)
OPX

(TAG* FN TAG)

TAG* is a predicate which test if the value of the tag field of FN is equal to TAG. FN has the form (<file> . <line#>).

(TEST SENT)

calls for the evaluation of SENT. If SENT is non-null, TEST types "YES". If null, TEST types "NO". TEST is the function used to answer yeS-no questions.

(UNADDPROP X Y Z)

UNADDPROP is the reverse of ADDPROP. It removes the entry Z from the property Y on the atom X.

(UQUOTIENT NUP1 NUP2)

UQUOTIENT returns the quotient of the two number-unit pairs, NUP1 and NUP2, first performing any unit conversion necessary.

(WHQFILE SN)

WHQFILE returns the chemical analysis file on which the analyses of sample SN are stored. For example, (WHQFILE (QUOTE S1ØØ17)) returns <WARNER>APOLLO11.

Appendix E.

THE ORGANIZATION OF THE DICTIONARY

The following description is intended to serve two purposes: first, to provide a general picture of the dictionary, indicating what types of information must be specified for lexical entries; and second, to demonstrate the precise format in which this information must be represented.

I. An Overview

The dictionary entry for a given word is stored on its LISP property list as a sequence of property-value pairs (see the appendix for a formal specification of the syntax required in a definition. Usually the properties will be the names of lexical categories (e.g. N, V, ADJ), indicating that the word can be a member of the category, but three other properties are allowed: SUBSTITUTE, COMPOUNDS, and FEATURES. SUBSTITUTE supplies a mechanism for mapping abbreviations and alternative spellings of a word into a single form, which contains the full dictionary entry. If a word can be the first word of an idiom or compound expression (e.g. "United" in "United States"), then the property COMPOUNDS denotes the following word in the compound and a standard form which will replace the whole sequence when it is found in a string. Thus the pair COMPOUNDS ((STATES UNITED-STATES)) on the property list of UNITED would convert all occurrences of the sequence UNITED STATES into the single word UNITED-STATES, which then must be entered separately in the dictionary. The implementation of the lexical category properties, SUBSTITUTE, and COMPOUNDS, all support the general philosophy that the dictionary information for a number of related items should be stored on only one standard form but should be accessible by any of the items.

II. Lexical Categories

The lexical categories are the properties explicitly referenced by the grammar and the parsing algorithm. When the grammar asks if a word is in a particular lexical category, the dictionary look-up routines provide a yes-no answer and, if yes, two kinds of information: (1) the root form of the word, and (2) a set of inflectional features. Thus if the grammar asks if BOOK is a noun, the answer is "yes--with root BOOK and inflectional feature (NU SG)". For the verb TALK the root would be TALK with inflectional features (TNS PRESENT) and (PNCODE 3SG).

The value of the lexical category property encodes the root and feature information in several ways (see Section VI). The most transparent notation is simply a parenthesized sequence (a list) whose first element is the root and whose succeeding elements are the features. If the word has a number of different interpretations within a single category (e.g. SAW as a verb), the value of the category property is a list of root-feature lists, one for each interpretation. If the value of the property is an atom, (a character-sequence instead of a list), then the root features are supplied by default dictionary routines: If the value is "*", then the word itself, is taken as its own root and the set of features is the empty set. For any other atomic value, the root is still the word itself, but a default set of features is provided, depending on the category (e.g., nouns are marked as singular by default).

Atomic values have another side-effect: they specify the morphological paradigm of which the word is the root. Thus for verbs, the atomic value S-ED indicates that the third-person singular is formed by adding an S, the past tense and past participle result by adding ED, and the present participle is formed with ING. With the inflectional paradigm encoded in this way, only the root forms of regular verbs, nouns, adjectives, and adverbs

must be entered in the dictionary. Definitions for inflected forms are constructed as needed by removing suffixes to obtain a potential root and making sure that the potential root is in the dictionary and is marked to allow the removal of that suffix. If so, the inflected form is defined as having that root and features determined by the suffix in a regular way.

Having outlined the general structure of definitions, we can now look at the lexical categories in some detail. We distinguish two kinds of lexical categories, open and closed. Open categories are large, potentially infinite classes of words (such as nouns and verbs) which will never all be in the dictionary. These classes are quite productive, with new members arising almost daily, as technology progresses. The closed categories are finite, and, for the most part small, and they are not growing. These categories include prepositions, determiners, conjunctions, and modals.

A. Open Category Properties.

N	=	noun (man, airplane, city)
NPR	=	proper noun (John Smith, USAF)
V	=	verb (walk, fly, see)
ADJ	=	adjective (tall, happy, green)
ADV	=	adverb (quickly, suddenly, certainly)

B. Closed Category Properties

CONJ	=	conjunction (and, or, but)
PREP	=	preposition (to, for, over)
PRO	=	pronoun (I, you, they)
DET	=	determiner (the, a, those)
ORD	=	ordinal (first, second, last, final)
NEG	=	negative (not)
COMP	=	comparative (more, less, greater)
OP	=	operation (plus, times)
QWORD	=	question noun (who, what, why)
QDET	=	question determiner (which, what)
MODAL	=	modal verb (should, would, can)
INTEGER	=	integer (one, two, three)

III. Open Categories

A. N - Noun

The property N indicates that the word can be interpreted as a common noun. Every noun is mapped into its root form and supplied with an inflectional feature for number. This feature is encoded as follows:

(NU SG) if this is a singular form
(NU PL) if this is a plural form (e.g. OXEN)
(NU SG/PL) if this form is considered both singular and plural (e.g. FISH, SHEEP)

The property N should not have the value *; the feature (NU SG) is supplied by default for other legal atomic values.

The atomic arguments specify how the plural, if any, is formed The following atoms are recognized (the hyphens are required):

-S nouns with regular S plurals (e.g. BOOK, BOOKS, MULE, MULES)
-ES nouns with regular ES or IES (if the root ended in Y) plurals (e.g. CHURCH, CHURCHES, PONY, PONIES)
MASS mass and abstract nouns which have no plural (e.g. WATER, HEALTH)
IRR nouns whose plural form is irregular (e.g. DATUM, OX) (note that the plural form must have a separate dictionary entry)

The definitions given for the following words illustrate these conventions:

```
BOOK      (N -S)
HEALTH    (N MASS)
OX        (N IRR)
OXEN      (N (OX (NU PL)))
```

B. NPR - Proper Noun

Proper nouns have a very simple structure, since they do not have inflectional forms or features. The basic entry for a proper noun is (NPR *), although it is possible to use the root retrieving routines to provide a SUBSTITUTE effect. Thus if JOSEPH were defined as (NPR *), (NPR (JOSEPH)) and (SUBSTITUTE ((JOSEPH))) would be equivalent definitions for JOE.

C. V - Verb

The inflectional structure of verbs is more complicated than that of nouns. A verb is marked as a tensed form (present or past), an infinitive, and/or a participle (present or past). In addition, if the verb is marked as tensed, it must also be marked for person and number. These features are specified in the following way:

(TNS PRESENT) (TNS PAST)	for tensed forms
(PRESPART T)	for present participles
(PASTPART T)	for past participles
(UNTENSED T)	for untensed infinitive forms
(PNCODE 3SG)	for third-person singular forms
(PNCODE X3SG)	for every thing except the third-person singular
(PNCODE ANY)	for all person-number combinations

If the value of an inflectional feature is T, the T need not be specified. Thus, the following abbreviations may be used where appropriate: (PRESPART), (PASTPART), (UNTENSED). In addition, the grammar has been designed so that a tensed verb that has no PNCODE specified will be interpreted as if it has (PNCODE ANY) permitting the elimination of this very common feature.

As for nouns, the atom * is not a permissible property value. For other legal atomic values, the default features are

(TNS PRESENT) (PNCODE X3SG) (UNTENSED), which correspond to

the normal behavior of the infinitive (root) form. The legal atomic values are (note the absence of an initial hyphen):

S-D — regular verbs which add S for the third-singular, D for the past tense and past participle, and ING for the present participle (e.g. INCLUDE, INCLUDES, INCLUDED, INCLUDED, INCLUDING).

S-ED — regular verbs like the above except that they add ED for the past tense and past participle (e.g. HAPPEN, HAPPENS, HAPPENED, HAPPENED, HAPPENING)

ES-ED — the same as S-ED except that the third-singular is formed with ES. Verbs that change Y to I and add ES or ED are also included. (e.g. PASS, PASSES, PASSED, PASSED, PASSING, STUDY, STUDIES, STUDIED, STUDIED, STUDYING)

IRR — infinitive forms of irregular verbs--all the other forms must have separate entries. (e.g. GIVE, MAKE, RUN)

Illustrative examples:

```
INCLUDE  (V S-D)
HAPPEN   (V S-ED)
GIVE     (V IRR)
GAVE     (V (GIVE (TNS PAST)(PNCODE ANY)))
SAW   (V ( (SEE (TNS PAST))
           (SAW (TNS PRESENT) (PNCODE X3SG) (UNTENSED)) ))
SAWED (V (SAW (TNS PAST)))
```

Notice that if the root of one verb and an inflected form of another are homographs (i.e. they are spelled the same), the regular inflectional machinery cannot be used--all the forms of the homographic root must be explicitly defined. There is one other restriction: the features (UNTENSED) and (PASTPART) are mutually exclusive, so that the few verbs whose infinitive and past participle are the same must be handled specially, as illustrated below.

```
(RUN (V ( (RUN (TNS PRESENT) (PNCODE X3SG) (UNTENSED))
          (RUN (PASTPART)) ))
```

RUN is thus defined as an ambiguous verb whose two interpretations have the same root but different features.

D. ADJ - Adjective

Ordinary adjectives in English do not have any features, but many of them have inflected comparative and superlative forms. These are marked by the features (COMPARATIVE T) and (SUPERLATIVE T), which may be abbreviated (COMPARATIVE) and (SUPERLATIVE). Adjectives which do not admit these inflections in a regular way are simply marked as (ADJ *), for example, EXTREME and ESSENTIAL. Otherwise, adjectives can have the atomic values

R-ST if they form the comparative by adding R and the superlative by adding ST (e.g. CLOSE, CLOSER, CLOSEST)

ER-EST if they add ER and EST instead (e.g. PINK, PINKER, PINKEST) Adjectives which change Y to I are also included.

These atomic values do not supply any default features for the root form. Examples:

```
HAPPY    (ADJ ER-EST)
GOOD     (ADJ *)
BETTER   (ADJ (GOOD (COMPARATIVE)))
BEST     (ADJ (GOOD (SUPERLATIVE)))
```

It should be noted that nouns which can be used as modifiers need not be categorized as adjectives, since the grammar recognizes noun-noun modification.

E. ADV - Adverb

Like adjectives, adverbs can also be inflected for comparative and superlative, but the root itself has no features. Thus, irregular adverbs or adverbs that do not have comparatives or superlatives are marked (ADV *), while the regular forms use the same atomic value codes (ER-EST and R-ST) as adjectives. Examples:

```
HARD        (ADV ER-EST)
FAR         (ADV *)
FURTHER     (ADV (FAR (COMPARATIVE)))
```

IV. Syntactic Features

The property FEATURES is required in the definitions of root forms to specify the syntactic behavior of the root and all its inflected forms. At present, the grammar only examines FEATURES on verbs so that the property need not appear on roots in other categories. The value of FEATURES is a simple unordered list of atoms, each one denoting a different characteristic. The features which may be included for verbs are:

TRANS	if the verb can be transitive (e.g. HIT, KICK)
INTRANS	if the verb can be intransitive (e.g. WALK, GO)
INDOBJ	if the verb can take an indirect object (e.g. GIVE, BUY, TELL)
COPULA	if the verb can be a copular (i.e. can be followed by a predicate adjective) (e.g. BE, SEEM, APPEAR)
PASSIVE	if the verb can be passivized (e.g. DISCOVER, FIND) (Note: all PASSIVE verbs are TRANS, but not all TRANS verbs are PASSIVE--e.g. COST)

Section VII contains a set of sentence frames which define these verbal characteristics.

With one exception, if a feature does not appear in the list, the grammar assumes that the verb does <u>not</u> have the characteristic in question. Thus, if the root WALK is not marked INTRANS, the grammar will not be able to parse the sentence "John walked." The exception is that in the special case when the only features a verb has are TRANS and PASSIVE, the whole FEATURES property may be ommitted. The following two definitions for KICK are equivalent:

```
KICK    (V S-ED FEATURES (PASSIVE TRANS))
KICK    (V S-ED)
```

Since a large proportion of verbs have only these two features, this convention reduces the total size of the dictionary. Examples of complete dictionary entries are:

```
GIVE      (V IRR FEATURES (PASSIVE TRANS INDOBJ))
GIVEN     (V (GIVE (PASTPART)))
GO        (V IRR FEATURES (INTRANS))
BECOME    (V IRR FEATURES (COPULA))
```

V. SUBSTITUTE and COMPOUNDS

The properties SUBSTITUTE and COMPOUNDS change the words in the sentence, before the grammar has even looked at them. If none of the substitutions or compounds lead to a valid parse, the parser restores the sentence to its original form. In this case, the grammar examines the lexical category information in the word's definition; thus a definition can contain lexical category properties as well as SUBSTITUTE and COMPOUNDS.

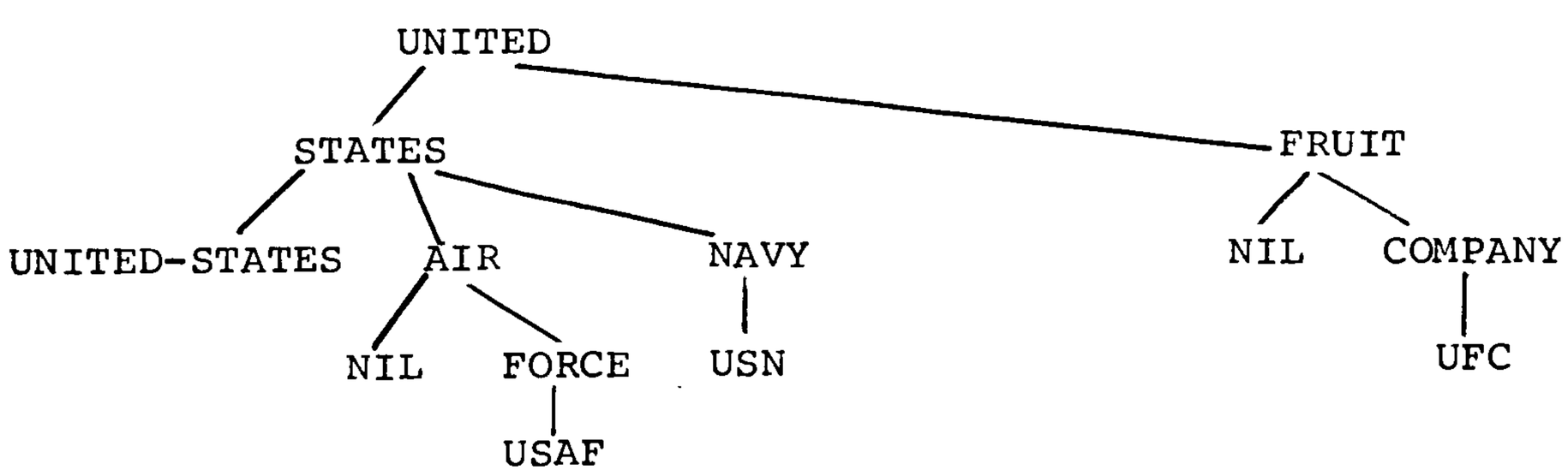
The value of SUBSTITUTE is a list of lists, each list being a possible string to be substituted for the word. Whereas COMPOUNDS causes a sequence of words to be replaced by a single word, SUBSTITUTE can have the opposite effect: If SRO were define as (SUBSTITUTE ((STANDING ROOM ONLY)), every occurrence of SRO in a sentence would effectively lengthen the string to be parsed.

As indicated earlier, COMPOUNDS provides a means of mapping idioms and compound expressions (sequences of words whose joint meaning is not simply the composition of the meanings of the individual words) into a single word representing the joint meaning. Thus in the earlier example, the sequence UNITED STATES was mapped into the "word" UNITED-STATES, which was then explicitly defined. The COMPOUNDS mechanism is general enough to handle arbitrarily long sequences and sequences which are identical to the initial

segments of longer sequences (e.g. UNITED STATES and UNITED STATES AIR FORCE). The various possibilities are expressed in the value of the COMPOUNDS property.

The compounds value (defined in Section VI can be thought of as a tree structure rooted in the word being defined (e.g. UNITED). Non-terminal nodes in the tree specify intermediate words in the compound expression, so that the non-terminal nodes encountered in tracing a path in the tree down from the root denote the sequence itself. The terminal node at the bottom of the path is the joint meaning of the sequence. There is a terminal node under each non-terminal to specify the joint meaning of subsequences that can occur independently; if the terminal is the atom NIL, then the non-terminal in question cannot be the last word in a sequence. The following example shows the value for COMPOUNDS and the corresponding tree structure necessary to recognize the expressions UNITED STATES, UNITED STATES AIR FORCE, UNITED STATES NAVY, and UNITED FRUIT COMPANY:

```
(UNITED   COMPOUNDS ((STATES UNITED-STATES (AIR NIL (FORCE USAF))
                                           (NAVY USN))
                     (FRUIT NIL (COMPANY UFC))))
```



Of course, UNITED-STATES, USAF, USN, and UFC must be defined separately (probably as (NPR *)).

VI. Dictionary Formats

The following is a formal specification of the syntax for dictionary definitions. The notation is similar to that used to describe context-free languages, except that nonterminal symbols are enclosed in angle-brackets and alternations are represented by the vertical bar. The only addition is the Kleene * operator, used to denote an arbitrarily (zero or more times) repeatable constituent.

(1) <definition> → (<definiens> <pair>*)

(2) <pair> → <lexical category><legical category value> |
SUBSTITUTE <substitute value> |
FEATURES <feature value> |
COMPOUNDS <compounds value>

(3) <legical category> → N | V | ADJ | ADV ...

(4) <legical category value> → <morphology code> |
"*" |
<root-feature list> |
(<root-feature list> *)

(5) <root-feature list> → (<root> <inflectional feature> *)

(6) <inflectional feature> → (<inflectional feature name>
<inflectional-feature value>)

(7) <substitute value> → (<substitution> *)

(8) <substitution> → (<word> *)

(9) <feature value> → (<feature> *)

(10) <compounds value> → (<tree> *)

(11) <tree> → (<word> <result> <tree> *)

(12) <result> → <word> | NIL

<lexical category>, <morphology code>, <root>, <inflectional-feature name>, <inflectional-feature value>, <word>, and <feature> are all atoms as specified in the text. <definiens> is the word being defined.

VII. Frames for Syntactic Features

The following is a suggestive set of sentence frames for the determination of the syntactic features which must be specified in the definition of verbs. If a verb can fit into the open slot in a frame, then the root form of the verb must be marked with the syntactic feature (under the property FEATURES) with which the frame is associated. It should be noted that for some verbs it might be necessary to change the pronouns or substitute other noun-phrases in the frame in order to arrive at meaningful sentences; if a grammatical sentence results after these modifications, the verb must still be marked with the feature in question.

A. TRANS: A verb must be marked TRANS if it can be immediately followed by a direct object noun-phrase.
They _____ it. (e.g. "hit" but not "go")

B. INTRANS: A verb is intransitive if it does not require a direct object. (Note: a verb can be both TRANS and INTRANS, if the direct object is optional.)
They _____. (e.g. "ran", but not "surprised")

C. INDOBJ: A verb can take an indirect object and must be marked INDOBJ if (1) it can be followed by two noun-phrases, and (2) if interchanging the two noun-phrases and inserting the word "to" between them does not change the meaning of the sentence.
They _____ him it. (e.g. "gave", but not "hit")
They _____ it to him.

D. COPULA: A verb is a copula if it can be immediately followed by an adjective which is predicated of the subject.

They _____ tall. (e.g. "are", but not "weigh")

E. PASSIVE: Transitive verbs which can be passivized must be marked both TRANS and PASSIVE.

He _____ them.
They were _____ by him. (e.g. "see (saw, seen) but not "cost")

Appendix F

THE RETRIEVAL COMPONENT

THE RETRIEVAL COMPONENT

I. INTRODUCTION

In this appendix we will give a detailed description of the general operation of the LSNLIS retrieval component and of the data structures and storage techniques used in the system. We make no claims that the file storage and accessing techniques used are the best ones (or even good ones) for our data base, but rather, we include this appendix for the sake of giving a complete specification of the current system. Since the goals of the LSNLIS project involved the language processing capability rather than file structures and data management techniques, we have adopted, wherever possible, techniques which are straight-forward and convenient, and we have capitalized extensively on facilities of TENEX such as the page-mapping facility and random file I/O.

The current data base consists of two files compiled by Dr. Jeffrey Warner at the Manned Spacecraft Center in Houston. The first is a formatted, fixed-record-length file of chemical analysis data on the Apollo 11 samples and the second is an inverted index by keyphrases to a small collection of documents. The former file is the data base of primary interest since it contains the specific factual material to answer questions. The second file was a peripheral effort in order to combine both fact retrieval

and document retrieval in the same natural language querying facility. The current state of the keyphrase file would be inadequate for an effective document retrieval system since the keyphrases were originally extracted by machine and there is no standardization of vocabulary (or even of inflection) in the file and we have not introduced any compensating synonym facility.

In the NASA LSNLIS, the retrieval component resides in a separate fork of the TENEX time-sharing system which we will call the lower fork or retrieval fork. This fork is under the control of the language processing fork. When the semantic interpretation component has finished constructing the interpretation of a request, it calls a function EXECUTE with this interpretation as its argument. EXECUTE passes the interpretation to the retrieval fork by means of a buffer file QBUF (for query buffer) and wakes up the retrieval fork. When the retrieval fork has completed processing the query, it will have written the answer(s) onto a file HITFILE, and it will then write the number of hits into a buffer file ABUF and return control to the upper fork. The function EXECUTE then prints out the answer if there are fewer than 5 hits, or notifies the user of the number of hits otherwise and asks him whether he wishes to see the answers. The function EXECUTE, thus serves as the access port to the lower fork.

II. THE LSNLIS DATA BASE

A. OVERVIEW

In the first LSNLIS prototype demonstrated in Houston in January, 1971, the entire data base was contained in the virtual core memory of the retrieval component, This system, while adequate for the demonstration, placed a limit of approximately 100K on the size of the data base that could be stored due to the limit of 256K for the total retrieval component. The Apollo11 data which the system then contained nearly filled that capacity.

In the current prototype, the data base has been moved from the virtual core memory to external disk file storage. This facility provides access to any number of independent disk files, each of which may contain up to 256K words of data. The current system contains conventions for both fixed and variable record-length files. Accessing of the files uses both the page mapping facility of the TENEX system and its random file I/O capability.

Due to our use of the hardware page-mapping facility in TENEX and more detailed indexing, moving the chemical analysis data base to external files has not hurt the retrieval component's performance time. With the function optimization that has accompanied this major change in the retrieval component, the average retrieval time for a request has actually been reduced. For example, the form

constructed from the request, "Give me chromite analyses for samples containing chromite.", now takes 11.5 seconds, on the average, (828 conses), to execute, whereas previously it took 24 seconds (1025 conses). "Potassium / Rubidium ratios for breccias" executes in 22 seconds (1830 conses), rather than 28 seconds (654 conses). (Part of the increase in speed was due to the correction of a bug in the original retrieval program which caused wasted searching to take place.)

To avoid the nuisance of constant, time-consuming updates to large files, we have carried over the "main table - patch table" idea from the previous system into the new one. Each main file may have associated with it a patch file, to which updates may be added sequentially. Facilities are then provided for sorting the patch file into the same order as the main file, merging the two files and resetting the patch file to accept a new set of updates. The retrieval functions have been written to search for information in the patch file, before going on to its associated main file. In this way, new information or corrected information is found first.

In the new system, we have again employed field coding and record packing wherever possible to reduce the size of the files. In the Apollo11 chemical analysis table, each line (or record) contains seven fields (for the sample

number, phase, constituent, content, unit, citation, and tag). If each of these entries were represented by one machine word, then an entry would require at least seven machine words of storage per record. However, the number of different possible values for a given field is usually far less than the number of distinct numbers that can occupy a machine word (36 bits) or even a LISP pointer (18 bits). Thus, we can save significant space by assigning each possible value for a given field a unique code number and reserving for that field just enough space to hold the largest such number (plus perhaps some margin for growth). Such field coding significantly reduces the number of bits required for each field. Record packing involves compacting a record to fit into the minimum number of words possible. Several fields may be assigned locations within a single word, rather than each field requiring one or more words to itself. In the current system, the only type of field which still requires a full word to itself is one which contains a real number. The records of the Apollo 11 file, with 7 fields per record, require only 3 machine words per record of storage.

A code number does not have to be decoded until the field of the record in which it is located is accessed. The decoding process is a very simple one and does not add appreciably to the cost of retrieving the information. The saving in file space is immense. The original symbolic file

of Apollo 11 chemical analysis information has been reduced by a factor of 3 by employing field coding and record packing.

The above discussion of field coding and record packing applies only to the fixed record length files of the system. At present, the one free-record-length file that is in the data base (the inverted file of documents by key phrase) is not bit packed in any way. Similar techniques could be applied to reduce the storage for such variable length records if space for their representation became critical.

The files currrently in tne data base are (1) a 13,248 record file APOLLO11, (2) its empty patch file, APOLLO11.PATCH, (3) the inverted file of documents indexed by key phrase, PHRASETABLE, and (4) one auxilliary file, LOWFORK.SYSOUT, used for maintaining updates to the lower fork.

B. FILE HANDLING

1. FIXED RECORD-LENGTH FILES

A process running under the TENEX system has a virtual memory of 256K, divided into 512 word units or pages. A user's files are also segmented into pages, and it is on these facts that the LSNLIS data handling is based. Given information about a file's organization, an algorithm can

compute the page on which any record within that file is located. It will also note on what word, within that page, the particular record starts. TENEX allows one to map pages from an external file onto pages within an ongoing process. Within the lower fork (retrieval component), 10 pages have been reserved for file page mapping. When a page of a file is needed by the retrieval function, it is mapped onto one of these 10 pages and is then available to all the standard LISP functions as if it were part of core memory. A gazetteer keeps track of which pages of which files are currently mapped onto the reserved area. If a given file page is already mapped onto the reserved area, it is available for use immediately, and no further mapping must be done in order to access it. (* 1)

The following discussion details the current implementation:

1.1 FILE DESCRIPTION INFORMATION

The names of the data files available to the system are

(* 1) The mapping area was reserved by setting up a LISP array across 11 pages and freezing it there via the LISP MAKESYS command. Once a page from a file is mapped into the lower fork, it is accessed by means of the standard LISP array functions. Recently, a new facility, GETBLK, was added to LISP for assigning a block of storage guaranteed not to move during garbage collections, just for the purpose of such page mapping as we are doing.

on the list FILEDIRECTORY. FILEDIRECTORY is a list of dotted pairs, the first member of each being a file name, and the second being the current version. For example, FILEDIRECTORY is currently set to:

```
((APOLLO11 . <WEBBER>APOLLO11)
 (APOLLO11.PATCH . <WEBBER>APOLLO11.PATCH))
```

A user is not expected to know the current version of any file. If a name isn't on FILEDIRECTORY, the system assumes it has no further information about that name as a file, and returns an error message.

All the file names on FILEDIRECTORY have on their property lists the information that the retrieval functions need to locate the file-record-field combination they require. In addition, the property list of each file name contains the information needed to print out that file in a legible manner. Each of these properties is detailed below, with the property name being followed by a description of its value and examples of its use:

TOP - the number of records in the file, plus one. This value changes as new records are added to the file.

SORTTOP - the number of sorted records at the top end of the file. This value changes only when the recently added, unsorted records at the bottom of the file are sorted in with the rest, using the function SORTFILE. At that time, SORTTOP is reset

to TOP. This property is only useful information for patch files; main files are always assumed to be sorted.

NREC/PAGE - the number of records per page of the file. This number must be an integer, as records are not allowed to cross page boundaries in this implementation. AS there are 512 words per page, NREC/PAGE is equal to the integer quotient of 512 and the length of tne file record in words. The APOLLO11 file has three word records, so NREC/PAGE is 512/3 = 170.

FORMAT - the format of the file record - its length and field specifications. The first item in FORMAT is the record length in words, and the remaining items are format specifications of the fields within a record. A field is specified as:

(<fieldname><fieldtype><sortkeyflg><word-increment>
<msb><lsb><codearray>)

where <fieldtype> :=: FLT | INT | CODE

<sortkeyflg> :=: T | NIL

<word-increment> :=: 1 | 2 | 3 | ...

<msb> :=: 0 | 1 | 2 | ... | 35

<lsb> :=: 0 | 1 | 2 | ... | 35

<fieldtype> specifies whether the value in the field is a floating point number, an integer or a code number. All alphanumeric

strings, as mentioned previously, have been coded to save space.

<sortkeyflg> indicates whether the file is sorted on that field. Over the sorted fields, the order of the sort is from left to right.

<word-increment> indicates in which word of the record the field is located. <msb>,<lsb> are the bit boundaries of the field, if it does not occupy a full word. Only floating points numbers currently occupy a full word. <codearray> is only included for coded fields. The code number in the field is a pointer into this array, which contains the uncoded, alphanumeric information.

PRINTFORMAT - the width of each field, as it is to be printed, one number per field. A field must be as wide as its longest alphanumeric value.

For example, the property list of the file of Apollo 11 data, APOLLO11, is:

```
(APOLLO11
                TOP             13248
                SORTTOP         13248
                NREC/PAGE       170
                FORMAT          (3 (SN CODE T 1  0  13  SARR11)
                                   (MIN CODE T 1  14  24 MARR11)
                                   (ELT CODE T  1  25  35 EARR11)
                                   (REF CODE T  2  0  13 RARR11)
                                   (TAG INT T  2  14  24)
                                   (UNIT CODE NIL 2  25  35
                                    UARR11)
                                   (VAL FLT NIL 3)  )
                PRINTFORMAT   (8 8 10 10 10 10 10 4) )
```

1.2 CODING ARRAYS

Each file in the data base may have its own code arrays or share them with another file. SARR11, MARR11, EARR11, RARR11 and UARR11 are the code arrays of the sample, mineral, element, reference and unit fields, respectively, of the APOLLO11 file. All Apollo11 samples are coded via the SARR11 code array, all minerals via the MARR11 code array, etc. A value which occurs in more than one field or more than one file may be coded via several arrays. This information is stored on the property list of the field value, under the property CODES. For example,

```
(SILICA
        CODES ((MARR11 . 27) (EARR11 . 135)) )
```

says that SILICA is coded 27 as a mineral field value and 135 as an element field value.

A coding array contains in its first entry a pointer to its first empty entry. That entry will be used for the next

field value that needs to be coded. The remaining non-empty entries of the array point to the decoded field values. For example, entry 27 of MARR11 points to the string SILICA. The entries are accessed via the LISP array functions SETA and ELT.

The property list of a coding array contains the properties CODESFOR and SORTTOP. CODESFOR points back to a list of values for which it is a coding array. This list is used by the retrieval function FOR. Both the coding array and this list are updated when new field values come in. For example,

```
(MARR11
    CODESFOR PHASES)
```

When a new value is entered and coded in MARR11, it is also appended to the list PHASES.

As the chemical analysis files are expected to be sorted alphabetically and maintained in this order, the values in each coding array have been coded in this order too. In this way, the compacted, coded files maintain the alphabetic order. When new analyses with new field values are added to a chemical analysis file, the code assigned to the field value may not reflect its alphabetic order. The property SORTTOP indicates the end of the ordered values within the array.

An example of the complete property list on a coding array is:

```
(MARR11
    CODESFOR PHASES
    SORTTOP 67).
```

1.3 FILE INDEXING

To aid in locating information within a file, a file is indexed on its primary sort key. For each member of the primary field, the index to the file for that member is given on its property list. The property name is the name of the file: thus, several files may be indexed on the same primary sort keys. (This might be useful if one were to have some files of the samples by mission number and others by sample type.) The value of the property is the detailed index to the file for that value of the primary field. This index may vary in depth according to the size of the block delimited. (It is assumed that the file is sorted, at least on its initial field.)

The index structure is:

```
<index> :=: (<fieldindex>* <lastindex>)

<fieldindex> :=: (<fieldname> . <integer>) |
                 (<fieldname><index>)

<lastindex> :=: (*** . <integer>)
```

Examples of a possible index to two samples in the APCLLO11 file, whose primary sort key is sample number, are:

```
(S10003
     APOLLO11 ((OVERALL . 19)(CPX . 180)
               (GLASS . 88)(ILM . 212)(PLAG . 213)
               (*** . 216)) )
(S10017
     APOLLO11 ((OVERALL. 334)(BOT . 650)
          (BOTTOM . 655)(CPX (AL203. 660) (CAO .
          669)(CR203 . 678)(FEO . 682)(MGO . 691)
          (MNO . 700)(NA20 . 705) (O . 706)(O18 .
          707)(PB . 710) (PB206/20 . 711)(SI02 .
          715)(TH . 724)(TI02 . 726) (U . 735)(*** .
          737))(ILM . 737)...(*** . 928)))
```

(While the APOLLO11 file is currently indexed only by sample - phase combination, the retrieval functions have all been written for variable depth indexing.)

In the above examples, there is a pointer to the start of the S10017-BOT block, containing the five analyses of S10017 for the BOT phase. The S10017-CPX block however, which contains 137 analyses, has been indexed further by element. Thus there is a pointer to the start of the S10017-CPX-FEO block, the S10017-CPX-MGO block, etc. This variable indexing is a help in reducing the amount of information that must be searched to find a given item.

2. FREE RECORD LENGTH FILES

We have changed our manner of handling the inverted lists of documents by keyphrase in the new LSNLIS system. Previously, these lists were kept in core on the property

list of the word heading the phrase. For example, the inverted file associated with "Quenched terrestrial basalts" was on the property list of "Quenched". (We hope there will be no confusion because of our use of the word "file" in both its information retrieval and its computer implementation senses. We will try, however, to use the phrase "external file" to distinguish the computer sense.) In order to accommodate larger numbers of larger inverted files, it was decided to keep them on an external file and access it with a binary search. The dictionary file in the upper fork is also accessed in this way.

In the current system, there is a single external file PHRASETABLE containing all the inverted files of documents by keyphrase. Each inverted file is a single record on this external file. We assume a keyphrase will have only one inverted file associated with it (i.e. will occur only once on the list). The external file is sorted on the keyphrases that head the inverted files. To retrieve a record from the external file, we do a binary search on the file using random file I/O commands recently introduced into LISP.

While we have not done comparative timing studies with the previous inverted file retrieval functions, recent timings of the function GETDOCS, which returns the inverted file for a phrase given as input, have produced the following results:

phrase	timing	number of calls for random I/O
ABRASION	5.3 sec.	19
ZONED CRYSTAL	6.4 sec.	26
QUENCHED TERRESTRIAL BASALTS	4.9 sec.	16
MOLTEN SILICATE	1.Ø7 sec.	2

This is of course much slower than the corresponding in-core versions, but one cannot expect to keep an index to a realistic document collection in 256K of core. Some such sacrifice in speed of retrieval is inevitable.

C. IN-CORE DATA

In addition to the chemical analysis information stored on external files, we also have a small amount of information resident in core. This information includes lists of sample types, samples by type, elements, isotopes, minerals, rare-earths, phases and oxides, and for each sample, on its property list, a list of the phases and a list of the constituents which the sample contains. This latter information is derived from the chemical analysis files and represents for each sample those phases and constituents for which analyses have been made. All this in-core information greatly improves the system's performance in answering requests about the elements, oxides, phases, etc. that a sample contains, without increasing very much the amount of memory required by the system.

For each sample, its list of constituents is on the

property ELTS on its property list. The phases that a sample contains are found on the property whose name is the same as the file to which the sample belongs. For example, the phases of S10017 can be found under the property APOLLO11. This property, as was mentioned in the previous section, is also the index to the file by sample - phase pair. It is also used to see what phases the sample contains. Both of these properties are put on the property list of each sample by the function INDEXF when it constructs the sample - phase index of the file.

```
[S10018
   CODES ((SARR11 . 6))
   APOLLO11 ((OVERALL . 923)
             (CPX . 1081)
             (GLASS . 1093)
             (ILM . 1173)
             (PLAG . 1185)
             (*** . 1196))
    ELTS (AL203 AL26 AU BA BE CAO CE CL CO CO56 CR203 CU DY ER EU
                FEO GA GD HF HO IN K2O LA LI LU MGO MNO MN54 NA2O
                NA22 NB ND NIO O POSITRON PR P2O5 RB RB87/SR8 S SC
                SC46 SIO2 SM SR SR87/86 TA TB TH TIO2 U V Y YB ZN ZR)
 ]
```

III. BUILDING THE LOWER FORK AND DATA FILES

The following sequence of instructions was used to build the lower fork and convert the LSNLIS chemical analyses file for Apollo 11 into a coded file and reformat the LSNLIS set of inverted files (* 2). (The external files are independent. If anything should happen to the chemical analysis or keyphrase files, they may be reload or recreated

without affecting the lower fork. The chemical analysis file does not have to be reindexed. If anything should happen to the lower fork, neither of the data files needs to be redone. Only the property list information for the chemical analysis file and its indexing must be restored.)

1. @LISP
2. ←LOAD(<WEBBER>LOADLOW)
3. ←(LOADLOW LOADFILES) LOADLOW will call for the loading of all the function and variable files needed in the lower fork. It will also set up all the global variables needed for page mapping.
4. ←(BUILDCA CODEARRAYS FIELDNAMES) BUILDCA will set up the code arrays for the APOLLO11 file listed in CODEARRAYS and code the appropriate field values, for the lists in FIELDNAMES, into the arrays.

(* 2) The facility for getting the JFN of a file from within LISP has not been implemented yet at the time of this report. Because of this, the following additional set of instructions must be performed before loading the first file:

```
@LISPX
BBN LISP-10  03-09-72 ...
1←DDT()
@DDT
FSCH=[←LISP$:FSCH]12006
↑C
@REE
2←PUT(FSCH COREVAL 120060)
5126
```

(WHATEVER NUMBER IS RETURNED FROM TYPING "FSCH=", SHOULD BE USED IN THE PUT COMMAND, FOLLOWED BY A "Q", INDICATING THE NUMBER IS IN OCTAL.)

5. MINREAD(<WOODS>MINTABLE <WEBBER>APOLLO11) MINREAD creates a compacted, binary-coded file from a symbolic (i.e. printable) chemical analysis file.

6. ←INDEXF(<WEBBER>APOLLO11) INDEXF indexes a coded chemical analysis file by sample-phase combination. (N.B. Although the retrieval functions accept variable depth indexing, we found we did not need to index beyond the second level, (i.e. the phase), for reasonable efficiency.

7. ←REFILEPHRASES(<WOODS>PHRASETABLE <WEBBER>PHRASETABLE) REFILEPHRASES converts the LSNLIS set of inverted files on the external file <WOODS>PHRASETABLE into one suitable for binary search.

8. ↑C

@SSAVE (PAGES FROM) 0 TO 777 (ON) LOWFORK.SSAV

@CONTINUE

MAKESYS (LOWFORK.SAV)

This final sequence of instructions sets up the lower fork for use by the upper fork.

IV. UPDATING THE DATA BASE

A. USER REQUESTS

In addition to its retrieval facilities, LSNLIS has facilities which allow the user to alter the data base of chemical analysis information. These facilities allow him to add and delete analyses, to change the value of one or more fields within a single analysis, and to print out the chemical analysis files, all using natural language, although with some restrictions.

As the main chemical analysis files are in alphanumeric order and must remain so, all changes in analyses which would upset this order cause the analysis in question to be deleted from the main file and put in its altered form in the corresponding patch file. This eliminates a painful reshuffling of the main file whenever an update is made. At a later date, the main file and the patch file may be merged, and the patch file cleared for new entries, thus combining ease of maintenance with efficient searching.

As far as the user is concerned, there is only one file for each mission. He does not need to know about the main-file / patch-file dichotomy. Hence, when he wants to specify additions to the analyses for a given mission, he only need specify the main file. Deletions and changes however can be made to all the files for a mission.

Since changes made to the data base may also change values in the lower fork, the currency of the lower fork must also be maintained. This is done using the LISP

function SYSOUT, which will save the pages on which these changes were made on a file. Whenever a new entry is made in a coding array using the function CODE, or a new file is sorted using SORTNEW, the command:

(SYSOUT (QUOTE <WARNER>LOWFORK.SYSOUT))

is executed automatically. At the start of a user session, this file is automatically overlaid over the lower fork, bringing it up to date.

1. ADDING ANALYSES

The user can add a new analysis to a file in one of two ways:

A. **(ADDLINE (↑ S10003 OVERALL DY 24.7 PPM D70-220 0) to APOLLO11)

B. **(ADD THE LINE (S10003, OVERALL, DY, 24.7, PPM, D70-220, 0) to APOLLO11)

(The English here, as might be expected, is somewhat constrained. It is difficult though to imagine what the casual form for such a request might be.)

The user can say ADDLINE or ADD, followed by a list of the field values enclosed in parentneses, preceded by an up-arrow, as in example A., mentioning the file to which the analysis should be added. (The up arrow at the beginning of a list is our analog of underlining a phrase. It makes the remainder of the list into a proper noun.) This is the first

way of adding an analysis.

The second way of adding an analysis involves the user saying ADD, as in example B. (When the line contents are indicated by a phrase "the line ...", the values of the fields should be separated by commas, and no up-arrow should be used.) Again the file to which the analysis should be added must be mentioned.

Both of these requests, when executed, return the line number (record number) in the patch file of the new analysis. For example:

```
SENTENCE:
(ADD (↑ S10003 OVERALL DY 24.9 PPM D70-675 0) TO APOLLO11)
PTIMING:
1095 CONSES
5.519 SECONDS
PARSINGS:
S IMP
  NP PRO YOU
  AUX TNS PRESENT
  VP V ADD
     NP DET NIL
        NPR  S10003
             OVERALL
             DY
             24.9
             PPM
             D70-675
             0
        NU SG
     PP PREP TO
        NP DET NIL
           NPR APOLLO11
           NU SG
ITIMING:
541 CONSES
2.366 SECONDS
INTERPRETATIONS:
(DO (APPLY (FUNCTION PRENEWLINE) (QUOTE ((NPR* X1 / (QUOTE
 APOLLO11))
S10003 OVERALL DY 24.9 PPM D70-675 0))))
EXECUTING
5
T
------
```

2. DELETING ANALYSES

To delete a line from any of the chemical analysis files, the user can either specify the line number and file name of the line he wishes to delete, as in:

**(DELETE LINE 17 OF APOLLO11)

or describe the analysis contained on the line, as in:

**(DELETE THE OVERALL ANALYSIS OF AL26 FOR S10002).

Using the second method, the user need not specify the file name, as it is obtainable from the sample number,

However, using the second method, the user faces the problem of potential ambiguity: there may be several overall analyses of A126 for S10002. He may specify the analysis further by giving tag and reference numbers, but the possibility of ambiguity still exists. Because the user has specified "the analysis", he seems to believe there is a unique referent for his description. If there is then more than one analysis satisfying his description, the retrieval component will tell him so, and how many. At this point, the user may specify all, one, the first or whichever of them he chooses to be deleted.

To avoid this ambiguity, the user may nave first requested to get the exact line number(s) of the analysis or analyses he wished to delete, as by:

```
**(WHAT ARE THE OVERALL ANALYSES OF AL26 FOR
S10002)
```

3. ALTERING ANALYSES

The user is also given the ability to make natural language requests for alteration of one or more analyses within a file. He can get at an analysis in one of two ways: by giving the file name and line number of the analysis in a call to EDIT, as in:

**(EDIT LINE 173 OF APOLLO11)

or by providing a description of the line or lines to be changed and the type of change, as in:

**(CHANGE THE ELEMENT IN ALL AL2Ø3 ANALYSES TO AL2Ø3).

Using the first method, he can make as many changes as he wishes, but to only one line. Using the second method, he can make only one specific change, but to as many lines as he wants.

If the user chooses the first method, the machine will respond by printing out the analysis on line 173 of the APOLLO11 file in the order: sample, phase, constituent, reference, tag, unit, content. To change any field, he can specify its position (sample = 1, phase = 2, etc), followed by the new value. For example, the above request to EDIT would produce

(S1ØØØ3 OVERALL *K/AR D7Ø-241 Ø M.Y.39ØØ.Ø)

EDIT

*

To change the phase, the user might say (2 CPX) or (2 GLASS). To change the content, the user might say (7 3850.8). He can inspect the current result at any point by typing P. When he is finished, he can type OK. If a field on which the analysis file is sorted has been changed during the editing, the system deletes the original line and adds a

copy of the altered line to the end of tne corresponding patch file. Otherwise, it simply makes the indicated changes to the original line. (Of course, the user can also edit lines in the patch file, but tne above considerations still hold.)

The following is complete example of a user requesting a change in the Apollo11 patch file. The system returns at the end of executing the request the line number of the original line if no recopying was necessary, the line number of the new line in the patch file, if the line was copied.

```
SENTENCE:
(EDIT LINE 3 OF APOLLO11.PATCH)
PTIMING:
971 CONSES
5.669 SECONDS
PARSINGS:
S IMP
  NP PRO YOU
  AUX TNS PRESENT
  VP V EDIT
     NP DET NIL
        NPR LINE
            3
        NU SG
        PP PREP OF
           NP DET NIL
              NPR APOLLO11.PATCH
                 NU SG
ITIMING:
542 CONSES
2.435 SECONDS
INTERPRETATIONS:
(DO (EDITLINE (NPR* X2 / (QUOTE APOLLO11.PATCH)) (QUOTE 3)))
BBN LISP-10  03-09-72 ...
EXECUTING
(S10058 CPX SIO2 D70-212 0 NIL 11.9)
EDIT
1*(7 13.2)
2*OK
3
------
```

4. PRINTING FILES

The chemical analysis files, being binary files, are not printable (that is, comprehendable when printed) by the TENEX executive command LIST. However, the user can request a file to be expanded, decoded, and printed by saying

**(PRINT APOLLO11) or **(PRINTOUT APOLLO11.PATCH)

that is, "print" or "printout", followed by the name of the file he wishes to have printed.

The following is an example of the above request:

```
SENTENCE:
(PRINTOUT APOLLO11.PATCH)
PTIMING:
352 CONSES
1.849 SECONDS
PARSINGS:
S IMP
  NP PRO YOU
  AUX TNS PRESENT
  VP V PRINTOUT
     NP DET NIL
        NPR APOLLO11.PATCH
        NU SG
ITIMING:
466 CONSES
2.159 SECONDS
INTERPRETATIONS:
(DO (PRINTFILE (NPR* X5 / (QUOTE APOLLO11.PATCH))))
EXECUTING
1        S10003   OVERALL   DY        24.7      PPM       D70-220   0
2        ******                      24.7
3        S10058   CPX       SIO2      13.2      NIL       D70-212
4        S10057             MNO       8.8                 D70-245   2
T
------
```

B. ADDING NEW DATA FILES

For each new chemical analysis file to be added to the data base, the following additional information must be stored. (We shall use a hypotnetical file of Apollo 12 data as an example, and assume both the original LSNLIS file, say MINTABLE12, and the coded file are to reside on the file directory <WARNER> . This is the only time the user need know the current version (i.e. the directory location) of the file.

1. @RUN LOWFORK.SSAV

2. ←SETQQ(APOLLO12 APOLLO12)

3. If any new code arrays are to be created for the file:
←SETQQ(CA12 ("list of new code arrays"))
←SETQQ(FLDNM12 ("list of field names in same order as code arrays. Each name should be bound to a list containing the values possible in the field."))
←BUILDCA (CA12 FLDNM12)

4. Add APOLLO12 to FILEDIRECTORY:
←SETQ(FILEDIRECTORY (CONS (QUOTE (APOLLO12 . <WARNER>APOLLO12)) FILEDIRECTORY))

5. Build the property list for APOLLO12. See previous section for the properties to be included.

6. If an empty patch file for the new main file is to be created concurrently, steps 4 and 5 should be repeated for the patch file. It is really not necessary to create the patch file until it is needed.

7. Convert LSNLIS file <WARNER>MINTABLE12 into a coded file:
MINREAD (<WARNER>MINTABLE12 <WARNER>APOLLO12)

8. Index the latter file by sample-phase combination:
INDEXF(<WARNER>APOLLO12)

9. Do any further indexing thought necessary.

10. Reset up the lower fork with the new information; see step 8 of the previous sequence.

APPENDIX G

Examples

The following examples illustrate a variety of the types of questions that the system can handle. They were run with a setting of flags which prints out the parse tree and the times involved in parsing and semantic interpretation as well as the resulting semantic interpretation and the answer. The parse times include the time required to access dictionary entries from external files when words are encountered that are not already in core (approx 1 sec. per word) as well as a considerable fluctuation in system overhead due to paging, which depends on system load at the time the example was run (times can fluctuate within a factor of four due to variations in system load). Since the examples were run at various times of day, and with varying numbers of words to be looked up from the external dictionary files, cross comparisons of times between sentences in this sample are meaningless, and the times themselves can give only a rough order of magnitude. The same system run in 256K of real core instead of virtual core would run much faster.

```
SENTENCE:
(WHAT IS THE AVERAGE COMPOSITION OF OLIVINE)
PTIMING:
1168 CONSES
4.9 SECONDS
PARSINGS:
S Q
  NP DET THE
     N AVERAGE
     NU SG
     PP PREP OF
        NP DET NIL
           N COMPOSITION
           NU PL
           PP PREP OF
              NP DET NIL
                 N OLIVINE
                 NU SG
  AUX TNS PRESENT
  VP V BE
     NP DET WHQ
        N THING
        NU SG/PL

ITIMING:
2285 CONSES
9.877 SECONDS
INTERPRETATIONS:
(FOR EVERY X8 / (SEQ MAJORELTS) : T ; (FOR THE X4 / (SEQL (AVERAGE
X5 / (SSUNION X6 / (SEQ SAMPLES) : T ; (DATALINE (WHQFILE X6) X6 (NPR*
X7 / (QUOTE OLIV)) X8)) : T)) : T ; (PRINTOUT X4)))

SIO2   (36.93518 . PCT)
TIO2   (.1544509 . PCT)
AL2O3   (.1236187 . PCT)
FE2O3   (0.0)
FEO   (28.97409 . PCT)
MNO   (.3488543 . PCT)
MGO   (33.82487 . PCT)
CAO   (.3093421 . PCT)
K2O   (0.0 . PCT)
NA2O   (.1381333 . PCT)
↑L
```

```
SENTENCE:
(WHAT IS THE AVERAGE PLAGIOCLASE CONTENT IN CRYSTALLINE ROCKS)
PTIMING:
1396 CONSES
6.85 SECONDS
PARSINGS:
S Q
  NP DET THE
     N AVERAGE
     NU SG
     PP PREP OF
        NP DET NIL
           ADJ NP N PLAGIOCLASE
           N CONTENT
           NU PL
           PP PREP IN
              NP DET NIL
                 ADJ CRYSTALLINE
                 N ROCK
                 NU PL
  AUX TNS PRESENT
  VP V BE
     NP DET WHQ
        N THING
        NU SG/PL

ITIMING:
2083 CONSES
10.925 SECONDS
INTERPRETATIONS:
(FOR THE X10 / (SEQL (AVERAGE X11 / (SSUNION X12 / (SEQ VOLCANICS)
: T ; (DATALINE (WHQFILE X12) X12 OVERALL (NPR* X13 / (QUOTE PLAG))))
: T)) : T ; (PRINTOUT X10))

(26.02778 . ***)
↑L
```

```
SENTENCE:
(WHAT IS THE AVERAGE CONCENTRATION OF ALUMINUM IN EACH BRECCIA)
PTIMING:
1256 CONSES
4.925 SECONDS
PARSINGS:
S Q
  NP DET THE
     N AVERAGE
     NU SG
     PP PREP OF
        NP DET NIL
           N CONCENTRATION
           NU PL
           PP PREP OF
              NP DET NIL
                 N ALUMINUM
                 NU SG
        PP PREP IN
           NP DET EACH
              N BRECCIA
              NU SG
  AUX TNS PRESENT
  VP V BE
     NP DET WHQ
        N THING
        NU SG/PL

ITIMING:
2223 CONSES
9.428 SECONDS
INTERPRETATIONS:
(FOR EVERY X17 / (SEQ TYPECS) : T ; (FOR THE X15 / (SEQL (AVERAGE
X16 / (DATALINE (WHQFILE X17) X17 (NPR* X18 / (QUOTE OVERALL)) (NPR*
X19 / (QUOTE AL203))) : T)) : T ; (PRINTOUT X15)))

S10018   (12.48526 . PCT)
S10019   (12.80726 . PCT)
S10021   (12.8297 . PCT)
S10046   (11.7149 . PCT)
S10048   (12.40324 . PCT)
S10056   (11.02208 . PCT)
S10059   (12.56518 . PCT)
S10060   (11.42593 . PCT)
S10061   (12.99457 . PCT)
S10063   (13.03755 . PCT)
S10064   (11.05357 . PCT)
S10065   (12.4707 . PCT)
S10066   (13.50992 . PCT)
S10067   (13.79335 . PCT)
S10068   (12.18727 . PCT)
S10070   (13.79335 . PCT)
S10073   (13.9823 . PCT)
S10074   (14.3602 . PCT)
S10075   (14.64362 . PCT)
↑L
```

```
SENTENCE:
(LIST MODAL PLAG ANALYSES FOR LUNAR SAMPLES)
PTIMING:
1039 CONSES
7.416 SECONDS
PARSINGS:
S IMP
   NP PRO YOU
   AUX TNS PRESENT
   VP V LIST
      NP DET NIL
         ADJ MODAL
         ADJ NP N PLAG
         N ANALYSIS
         NU PL
         PP PREP FOR
            NP DET NIL
               ADJ LUNAR
               N SAMPLE
               NU PL

ITIMING:
 659 CONSES
8.011 SECONDS
INTERPRETATIONS:
(DO (FOR GEN X14 / (SSUNION X15 / (SEQ SAMPLES) : T ; (DATALINE (WHQFILE
X15) X15 OVERALL (NPR* X16 / (QUOTE PLAG))) : T ; (PRINTOUT X14)))

 07      S10003  OVERALL   PLAG      33.8      ***       D70-154    0
 08                                   29.0                D70-173
513      S10017                      25.1                D70-155
514                                  21.5                D70-179
 679     S10020                      30.7                D70-159
 680                                 21.4                D70-173    31
 681                                 28.5                           40
1682                                 24.6                D70-305    0
2141     S10022                      15.6                D70-179
2727     S10024                      16.4
3109     S10044                      33.1                D70-154    41
3110                                 34.1                           42
3945     S10046                      4.7                 D70-305    0
4440     S10047                      37.8                D70-159
5491     S10057                      19.2                D70-173
5796     S10058                      37.1                D70-155
8354     S10071                      21.7                D70-173
8582     S10072                      20.4
8583                                 18.5                D70-179
9311     S10084                      22.0                D70-186
9312                                 15.0                D70-304
↑L
```

```
SENTENCE:
(REFERENCES ON TRITIUM PRODUCTION)
PTIMING:
984 CONSES
8.686 SECONDS
PARSINGS:
S NPU
  NP DET NIL
     N REFERENCE
     NU PL
     PP PREP ON
        NP DET NIL
           ADJ NP N TRITIUM
           N PRODUCTION
           NU SG

ITIMING:
1465 CONSES
8.847 SECONDS
INTERPRETATIONS:
(FOR GEN X17 / DOCUMENT : (ABOUT X17 (OR (TRITIUM PRODUCTION) (AND
(TRITIUM) (AND (TRITIUM) (PRODUCTION))))) ; (PRINTOUT X17))

D70-046
D70-051
↑L
```

```
SENTENCE:
(POTASSIUM / RUBIDIUM RATIOS FOR TYPE A ROCKS)
PTIMING:
1241 CONSES
9.127 SECONDS
PARSINGS:
S NPU
  NP DET NIL
     ADJ NP N N POTASSIUM
                 /
                 N RUBIDIUM
     N RATIO
     NU PL
     PP PREP FOR
        NP DET NIL
           ADJ TYPE/A
           N ROCK
           NU PL

ITIMING:
496 CONSES
9.412 SECONDS
INTERPRETATIONS:
(FOR GEN X20 / (SSUNION X1 / (SEQ TYPEAS) : T ; (RATIO (QUOTE K20)
(QUOTE RR) X1 (NPR* X2 / (QUOTE OVERALL)))) : T ; (PRINTOUT X20))

(524.4755 S10017 D70-205)
(558.437 S10017 D70-215)
(519.2241 S10017 D70-218)
(498.7992 S10017 D70-236)
(505.0877 S10017 D70-242)
(516.4452 S10017 D70-253)
(401.5333 S10017 D70-256)
(590.6333 S10017 D70-257)
(464.9298 S10022 D70-220)
(495.0289 S10022 D70-236)
(503.3557 S10024 D70-205)
(545.8454 S10024 D70-218)
(519.0071 S10024 D70-242)
(563.4355 S10049 D70-215)
(552.1041 S10057 D70-235)
(465.6154 S10057 D70-257)
(527.7797 S10057 D70-258)
(493.6746 S10069 D70-236)
(515.2281 S10069 D70-253)
(562.6813 S10071 D70-215)
(527.3129 S10071 D70-258)
(516.934 S10072 D70-205)
(534.7229 S10072 D70-218)
(612.8567 S10072 D70-235)
↑I
```

```
SENTENCE:
(WHAT IS THE AVERAGE K / RB RATIO IN BRECCIAS)
PTIMING:
1247 CONSES
Ø.Ø6 SECONDS
PARSINGS:
S Q
   NP DET THE
      N AVERAGE
      NU SG
      PP PREP OF
         NP DET NIL
          . ADJ NP N  N POTASSIUM
                      /
                      N RUBIDIUM
            N RATIO
            NU PL
            PP PREP IN
               NP DET NIL
                  N BRECCIA
                  NU PL
   AUX TNS PRESENT
   VP V BE
      NP DET WHQ
         N THING
         NU SG/PL

ITIMING:
2ØØ4 CONSES
9.387 SECONDS
INTERPRETATIONS:
(FOR THE X5 / (SEQL (AVERAGE X6 / (SSUNION X7 / (SEQ TYPECS) : T ;
(RATIO (QUOTE K20) (QUOTE RB) X7 (NPR* X8 / (QUOTE OVERALL)))) : T))
: T ; (PRINTOUT X5))

OVERALL  (476.9119)
↑L
```

```
SENTENCE:
(IN WHICH SAMPLES HAS APATITE BEEN IDENTIFIED)
PTIMING:
1523 CONSES
10.819 SECONDS
PARSINGS:
S Q
  NP DET WHICHQ
     N SAMPLE
     NU PL
     S REL
       NP PRO SOMETHING

                PERFECT
       VP V IDENTIFY
          NP DET NIL
             N APATITE
             NU SG
          PP PREP IN
             NP DET WHR
                N SAMPLE
                NU PL
  AUX TNS NIL
  VP V BE

ITIMING:
1878 CONSES
7.677 SECONDS
INTERPRETATIONS:
(FOR EVERY X15 / (SEQ SAMPLES) : (CONTAIN X15 (NPR* X17 / (QUOTE APATITE)
(QUOTE NIL)) ; (PRINTOUT X 5))

S10044
S10045
S10085
↑L
```

```
SENTENCE:
(WHICH ROCKS CONTAIN CHROMITE AND ULVOSPINEL)
PTIMING:
1423 CONSES
7.743 SECONDS
PARSINGS:
S NPQ
  NP DET WHICHQ
     N ROCK
     NU PL
     S QREL
       NP DET WHR
          N ROCK
          NU PL
       AUX TNS PRESENT
       VP V CONTAIN
          NP AND
             NP DET NIL
                N CHROMITE
                NU SG
             NP DET NIL
                N SPINEL
                NU SG

ITIMING:
1890 CONSES
9.782 SECONDS
INTERPRETATIONS:
(FOR EVERY X7 / (SEQ VOLCANICS) : (AND (CONTAIN X7 (NPR* X9 / (QUOTE
SPINEL)) (QUOTE NIL)) (CONTAIN X7 (NPR* X10 / (QUOTE CHROMITE)) (QUOTE
NIL))) ; (PRINTOUT X7))

S10020
S10045
↑L
```

```
SENTENCE:
(DO ANY BRECCIAS CONTAIN ALUMINUM)
PTIMING:
1100 CONSES
6.48 SECONDS
PARSINGS:
S Q
  NP DET ANY
     N BRECCIA
     NU PL
  AUX TNS PRESENT
  VP V CONTAIN
     NP DET NIL
        N ALUMINUM
        NU SG

ITIMING:
990 CONSES
5.011 SECONDS
INTERPRETATIONS:
(TEST (FOR SOME X2 / (SEQ TYPECS) : T ; (CONTAIN X2 (NPR* X3 / (QUOTE
AL203)) (QUOTE NIL))))

YES.
T
*L
```

```
SENTENCE:
(WHAT ARE THOSE BRECCIAS)
PTIMING:
474 CONSES
2.375 SECONDS
PARSINGS:
S Q
   NP DET THOSE
      N BRECCIA
      NU PL
   AUX TNS PRESENT
   VP V BE
      NP DET WHQ
      N THING
      NU SG/PL

ITIMING:
624 CONSES
2.814 SECONDS
INTERPRETATIONS:
(FOR EVERY X2 / (SEQ TYPECS) : (AND T (CONTAIN X2 (NPR* X3 / (QUOTE
AL203)) (QUOTE NIL))) ; (PRINTOUT X2))

S10018
S10019
S10021
S10046
S10048
S10056
S10059
S10060
S10061
S10063
S10064
S10065
S10066
S10067
S10070
S10073
S10074
S10075
↑L
```

```
SENTENCE:
(HOW MANY SAMPLES CONTAIN CHROMITE)
PTIMING:
618 CONSES
3.579 SECONDS
PARSINGS:
S NPQ
  NP DET HOWMANY
     N SAMPLE
     NU PL
     S QREL
       NP DET WHR
          N SAMPLE
          NU PL
       AUX TNS PRESENT
       VP V CONTAIN
          NP DET NIL
          N CHROMITE
          NU SG

ITIMING:
546 CONSES
8.277 SECONDS
INTERPRETATIONS:
(FOR THE X13 / (SEQL (NUMBER X13 / (SEQ SAMPLES) : (CONTAIN X13 (NPR*
X15 / (QUOTE CHROMITE)) (QUOTE NIL)))) : T ; (PRINTOUT X13))

(3)
↑L
```

```
SENTENCE:
(WHAT ARE THEY)
PTIMING:
376 CONSES
.536 SECONDS
PARSINGS:
S Q
  NP PRO THEY
     NU PL
  AUX TNS PRESENT
  VP V BE
     NP DET WHO
        N THING
        NU SG/PL

ITIMING:
 433 CONSES
7.296 SECONDS
INTERPRETATIONS:
(FOR EVERY X13 / (SEQ SAMPLES) : (CONTAIN X13 (NPR* X15 / (QUOTE
CHROMITE)) (QUOTE NIL)) ; (PRINTOUT X13))

S10020
S10045
S10084
↑L
```

```
SENTENCE:
(CAN YOU GIVE ME ALL CHROMITE ANALYSES FOR THOSE SAMPLES)
PTIMING:
1173 CONSES
4.729 SECONDS
PARSINGS:
S Q
  NP PRO YOU
     NU SG/PL
  AUX TNS PRESENT
      MODAL CAN
  VP V GIVE
     NP DET ALL
        PRO ONES
        NU SG/PL
        PP PREP OF
           NP DET NIL
              ADJ NP N CHROMITE
              N ANALYSIS
              NU PL
              PP PREP FOP
                 NP DET THOSE
                    N SAMPLE
                    NU PL
     PP PREP TO
        NP PRO I
           NU NIL

ITIMING:
2205 CONSES
12.421 SECONDS
INTERPRETATIONS:
(FOR EVERY X13 / (SEQ SAMPLES) : (CONTAIN X13 (NPR* X15 / (QUOTE
CHROMITE)) (QUOTE NIL)) ; (FOR EVERY X18 / (SSUNION X1 / (SEQ MAJORELTS)
: T ; (DATALINE (WHQFILE X 3) X13 (NPR* X20 / (QUOTE CHROMITE)) X1))
: T ; (PRINTOUT X18)))

1777    S10020  CHROMITE  TIO2      21.9       PCT        D70-160   0
1772                       AL203     7.05
1774                       FEO       44.4
1776                       MNO       .17
1775                       MGO       3.7
3632    S10045             TIO2      21.41                 D70-195   20
3633                       AL203     22.08                           21
3622                       FEO       6.32                            20
3623                       MNO       5.74                            21
3626                       MGO       44.36                           20
3627                                 44.41                           21
3630                                 .59                             20
3631                                 .69                             21
3628                                 3.22                            20
3629                                 3.2                             21
↑L
```

(* The other sample has only modal chromite analyses.)

```
SENTENCE:
(HOW MANY BRECCIAS DO NOT CONTAIN EUROPIUM)
PTIMING:
1223 CONSES
6.272 SECONDS
PARSINGS:
S NPQ
  NP DET HOWMANY
     N BRECCIA
     NU PL
     S QREL
       NEG
       NP DET WHR
          N BRECCIA
          NU PL
       AUX TNS PRESENT
       VP V CONTAIN
          NP DET NIL
             N EUROPIUM
             NU SG

ITIMING:
1570 CONSES
8.593 SECONDS
INTERPRETATIONS:
(FOR THE X3 / (SEQL (NUMBER X3 / (SEQ TYPECS) : (NOT (CONTAIN X3 (NPR*
X5 / (QUOTE EU)) (QUOTE NIL)))))) : T ; (PRINTOUT X3))

(1)
↑L
```

```
SENTENCE:
(WHICH BRECCIA IS THAT)
PTIMING:
921 CONSES
7.46 SECONDS
PARSINGS:
S NPQ
  NP DET WHICHQ
     N BRECCIA
     NU SG
     S QREL
       NP PRO THAT
          NU NIL
       AUX TNS PRESENT
       VP V BE
          NP DET WHR
             N BRECCIA
             NU SG

ITIMING:
1653 CONSES
6.783 SECONDS
INTERPRETATIONS:
(FOR THE X6 / (SEQ TYPECS) : (FOR EVERY X3 / (SEQ TYPECS) : (NOT (
CONTAIN X3 (NPR* X5 / (QUOTE EU)) (QUOTE NIL))) ; (EQUAL X3 X6)) ;
(PRINTOUT X6))

S10068
*L
```

```
SENTENCE:
(DOES S10004 CONTAIN EUROPIUM IN PLAG)
PTIMING:
1469 CONSES
0.525 SECONDS
PARSINGS:
S Q
  NP DET NIL
     NPR S10004
     NU SG
  AUX TNS PRESENT
  VP V CONTAIN
     NP DET NIL
        N EUROPIUM
        NU SG
        PP PREP IN
           NP DET NIL
              N PLAG
              NU SG

ITIMING:
952 CONSES
3.931 SECONDS
INTERPRETATIONS:
(TEST (CONTAIN (NPR* X3 / (QUOTE S10004)) (NPR* X4 / (QUOTE EU)) (NPR*
X5 / (QUOTE PLAG))))

↑L
```

```
SENTENCE:
(DOES IT CONTAIN IT IN OLIVINE)
PTIMING:
862 CONSES
4.508 SECONDS
PARSINGS:
S Q
   NP PRO IT
      NU SG
   AUX TNS PRESENT
   VP V CONTAIN
      NP PRO IT
         NU SG
         PP PREP IN
            NP DET NIL
               N OLIVINE
               NU SG

ITIMING:
 301 CONSES
6.375 SECONDS
INTERPRETATIONS:
(TEST (CONTAIN (NPR* X3 / (QUOTE S10004)) (NPR* X16 / (QUOTE EU))
(NPR* X17 / (QUOTE OLIV))))

↑L
```

```
SENTENCE:
(HOW MANY LUNAR SAMPLES ARE THERE)
PTIMING:
484 CONSES
2.Ø39 SECONDS
PARSINGS:
S NPQ
  NP DET HOWMANY
     ADJ LUNAR
     N SAMPLE
     NU PL
     S QREL
       NP DET WHR
          N SAMPLE
          NU PL
       AUX TNS PRESENT
       VP V EXIST

ITIMING:
998 CONSES
5.152 SECONDS
INTERPRETATIONS:
(FOR THE X12 / (SEQL (NUMBER X12 / (SEQ SAMPLES) : (EXIST X12))) :
T ; (PRINTOUT X12))

(43)
↑L
```

```
SENTENCE:
(WHAT TYPE OF SAMPLE IS S10046)
PTIMING:
2000 CONSES
11.075 SECONDS
PARSINGS:
S NPQ
  NP DET WHQ
     N TYPE
     NU SG
     PP PREP OF
        NP DET NIL
           N SAMPLE
           NU SG
     S QREL
       NP DET NIL
          NPR S10046
          NU SG
       AUX TNS PRESENT
       VP V BE
          NP DET WHR
             N TYPE
             NU SG

ITIMING:
1455 CONSES
7.544 SECONDS
INTERPRETATIONS:
(FOR THE X3 / (SEQ SAMPLETYPES) : (AND (MEMBER* (NPR* X4 / (QUOTE
S10046)) X3) T) ; (PRINTOUT X3))

BRECCIAS
↑L
```

```
SENTENCE:
(WHICH IS THE OLDEST ROCK)
PTIMING:
870 CONSES
4.35 SECONDS
PARSINGS:
S Q
  NP DET THE
     N OLD
       SUPERLATIVE
     NU SG
     PP PREP OF
        NP DET NIL
           N ROCK
           NU PL
  AUX TNS PRESENT
  VP V BE
     NP DET WHQ
        N THING
        NU SG/PL

ITIMING:
 097 CONSES
4.717 SECONDS
INTERPRETATIONS:
(FOR THE X14 / (SEQL (OLDEST X15 / (SEQ VOLCANICS) : T)) : T ; (PRINTOUT
X14))

S10047
↑L
```

```
SENTENCE:
(HOW MUCH TITANIUM DOES EACH BRECCIA CONTAIN)
PTIMING:
1131 CONSES
8.561 SECONDS
PARSINGS:
S Q
   NP DET EACH
      N BRECCIA
      NU SG
   AUX INS PRESENT
   VP V CONTAIN
      NP DET POSTART COMP ADV HOW
                          MUCH
         N TITANIUM
         NU SG
ITIMING:
858 CONSES
5.313 SECONDS
INTERPRETATIONS:
(FOR EVERY X16 / (SEQ TYPECS) : T ; (CONTAIN' X16 (NPR* X17 / (QUOTE
TIO2)) (HOW)))

S10018   (8.252875 . PCT)
S10019   (8.484685 . PCT)
S10021   (7.33964 . PCT)
S10046   (7.423045 . PCT)
S10048   (8.76608 . PCT)
S10056   (8.284897 . PCT)
S10059   (8.13199 . PCT)
S10060   (8.648534 . PCT)
S10061   (8.173667 . PCT)
S10063   (8.84093 . PCT)
S10064   (9.34136 . PCT)
S10065   (7.84007 . PCT)
S10066   (8.17369 . PCT)
S10067   (8.84093 . PCT)
S10068   (7.84007 . PCT)
S10070   (8.3405 . PCT)
S10073   (8.17369 . PCT)
S10074   (7.84007 . PCT)
S10075   (7.50645 . PCT)
↑L
```

```
SENTENCE:
(WHICH SAMPLES HAVE GREATER THAN 20 PERCENT MODAL PLAGIOCLASE)
PTIMING:
1140 CONSES
5.353 SECONDS
PARSINGS:
S NPQ
  NP DET WHICHQ
     N SAMPLE
     NU PL
     S QREL
       NP DET WHR
          N SAMPLE
          NU PL
       AUX TNS PRESENT
       VP V HAVE
          NP DET ART NIL
               POSTART COMP ADV GREATERTHAN
                            NP INTEGER 20
                               UNIT PCT
                      MUCH
             ADJ MODAL
             N PLAGIOCLASE
             NU SG

ITIMING:
1425 CONSES
6.811 SECONDS
INTERPRETATIONS:
(FOR EVERY X8 / (SEQ SAMPLES) : (CONTAIN' X8 (NPR* X9 / (QUOTE PLAG))
(GREATERTHAN 20 PCT)) ; (PRINTOUT X8))

S10003
S10017
S10020
S10044
S10047
S10058
S10071
↑L
```

```
SENTENCE:
(IN HOW MANY BRECCIAS IS THE AVERAGE CONCENTRATION OF ALUMINUM GREATER
THAN 13 PERCENT)
PTIMING:
2123 CONSES
9.749 SECONDS
PARSINGS:
S NPQ
  NP DET HOWMANY
     N BRECCIA
     NU PL
     S REL
       NP DET THE
          N AVERAGE
          NU SG
          PP PREP OF
             NP DET NIL
                N CONCENTRATION
                NU PL
                PP PREP IN
                   NP DET WHR
                      N BRECCIA
                      NU PL
                PP PREP OF
                   NP DET NIL
                      N ALUMINUM
                      NU SG
       AUX TNS PRESENT
       VP V BE
          NP DET ART NIL
                 POSTART COMP ADV GREATERTHAN
                              NP INTEGER 13
                          MANY
             N PCT
             NU SG

ITIMING:
3531 CONSES
20.805 SECONDS
INTERPRETATIONS:
(FOR THE X9 / (SEQL (NUMBER X9 / (SEQ TYPECS) : (FOR THE X10 / (SEQL
(AVERAGE X11 / (DATALINE (WHQFILE X9) X9 (NPR* X13 / (QUOTE OVERALL))
(NPR* X14 / (QUOTE AL203))) : T)) : T ; (GREATER X10 (QUOTE (13 .
PCT))))))) : T ; (PRINTOUT X9))

(7)
*L
```

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
THE LUNAR SCIENCES NATURAL LANGUAGE INFORMATION SYSTEM: FINAL REPORT

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific, final report, July 1, 1970 - June 15, 1972

5. AUTHOR(S) *(First name, middle initial, last name)*
W.A. Woods, R.M. Kaplan, B. Nash-Webber

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
15 June 1972	387	14

8a. CONTRACT OR GRANT NO.
NAS9-1115
b. PROJECT NO.
c.
d.

9a. ORIGINATOR'S REPORT NUMBER(S)
BBN Report No. 2378

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

11. SUPPLEMENTARY NOTES
This research was performed by BBN, subcontractor to Language Research Foundation, for NASA,MSC,Houston.

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The Lunar Sciences Natural Language Information System (LNSLIS) is a research prototype of a computer system to allow English language access to a large data base of lunar sample information. It allows a lunar geologist to ask questions, compute averages and ratios, make selective listings, etc. on a file containing currently some 13,000 chemical analyses of the lunar samples, as well as to retrieve references from a keyphrase index of documents (currently some 10,000 postings).

The emphasis of the LSNLIS project has been the development of the language processing techniques for understanding natural English requests, and the system contains a powerful language processing component. The language processor has a vocabulary of some 3500 words, a transition network grammar for a sizable subset of English, and a set of semantic interpretation rules for translating input English requests into formal procedures for answering them.

This report includes detailed descriptions of the system, and how it operates, together with examples of its performance and an evaluation of the future prospects for such systems.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
artificial intelligence						
computational linguistics						
computational semantics						
data management						
data retrieval						
English grammar						
English language processing						
fact retrieval						
information systems						
language processing						
LSNLIS						
lunar geology						
lunar sciences						
man-machine communication						
natural language						
natural language processing						
non-deterministic programming						
parsing						
query languages						
question-answering						
semantic interpretation						
semantics of natural language						
transition network grammar						