AFOSR 69-1481TR

AD689280

UNIVERSITY of PENNSYLVANIA

PHILADELPHIA, PENNSYLVANIA 19104

1. This document has been approved for public release and sale; its distribution is unlimited.

175

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

REAL ENGLISH: A TRANSLATOR TO ENABLE NATURAL LANGUAGE MAN-MACHINE CONVERSATION

by

Harvey Cautin

May 1969

The work described in this paper has been supported by the Air Force Office of Scientific Research, Information Sciences Directorate (system studies) and by the Army Research Office-Durham (implementation tasks). AF-49(638)-1421

The Moore School Information
Systems Laboratory

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

The Moore School Information
Systems Laboratory
University of Pennsylvania

Principal Investigator
Morris Rubinoff

Participating Faculty

Pier L. Bargellini
Aravind K. Joshi
George A. Mihram
P. K. Niyogi
Elisabeth P. Schoff
Richard F. Schwartz
Warren D. Seider

Student Staff

D. Callahan
H. Cautin
J. Cimprich
J. Crowley
J. Felsen
L. Hirschfeld
E. Hockstein
T. Johnson
G. Kaplan
J. Katzen
R. Margolies
J. O'Donnell
D. Pirog
T. Purdom
H. Schneps
D. Utman
A. Vivatson

ABSTRACT

REAL ENGLISH: A TRANSLATOR TO ENABLE

NATURAL LANGUAGE MAN-MACHINE CONVERSATION

Author - Harvey Cautin

Supervisor - Morris Rubinoff

This dissertation presents a pragmatic interpreter/translator called Real English to serve as a natural language man-machine communication interface in a multi-mode on-line information retrieval system. This multi-mode feature affords the user a library-like searching tool by giving him access to a dictionary, lexicon, thesaurus, synonym table, and classification tables expressing binary relations as well as the document file representing the field of discourse. The user is thereby allowed a greater freedom in search strategy.

Real English will 1) syntactically analyze the user's message by means of a string analysis grammar to produce a tree representing the interrelationships of the grammatical entities comprising the message, 2) use this tree together with a pragmatic grammar to establish the set of commands necessary to fulfill the request, and 3) form the proper syntax for each command. The strong linguistic foundation of the syntax analyzer endows the system with the power of flexibility. As experience shows that certain new structures occur and should therefore be a part of the system, they may be incorporated into the system by the grammarian without a major overhaul of the procedures to date.

The user is permitted to phrase his requests in any convenient form (i.e. declarative, imperative, interrogative, or fragmented sentence referred to as conversationally dependent sentences). Thus instead of placing the user in the difficult position of learning a new language, the system is given the responsibility of responding in and to the user's language, i.e. the man-machine conversation is carried out in a natural language.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to several people who gave their time, advice, assistance and encouragement.

To Dr. Morris Rubinoff, dissertation supervisor, the author wishes to express his thankfulness for the constant encouragement displayed throughout the development of the work.

To Carol Raze, the author offers his appreciation for her constant cooperation and helpfulness during long discussions concerning the performance of the New York University syntax analyzer.

The author is indebted to Dr. Avarind Joshi and Dr. Naomi Sager for their advice and encouragement during the preparation of the grammar used in this work. An additional note of appreciation is expressed to Dr. Joshi for his critical review of the final work.

INDEX

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

BIBLIOGRAPHY

1. Bookchin, B.: "Computer Outputs for Sentence Decomposition of Scientific Texts", String Program Reports No. 3, New York University, March 1968.

2. Cautin, H. and Rapp, F.: Description of Easy English, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, April 1967.

3. Chomsky, N.: "On Certain Formal Properties of Grammars", Information and Control 2, No. 2, pp. 137-167, June 1959.

4. Craig, J. A., et al: "DEACON: Direct English Access and Control", Proc. FJCC, pp. 365-380, 1966.

5. Green, B. F., et al: "Baseball: An Automatic Question Answerer", Computers and Thought, Feigenbaum and Feldman eds., 1963.

6. Harris, Z.: String Analysis of Sentence Structure, Mouton & Co., The Hague, 1962.

7. Lamson, B.G. and Dimsdale, B.: "A Natural Language Information Retrieval System", Proc. IEEE, Vol. 54, No. 12, December 1966.

8. Lindsay, R. K.: "Inferential Memory as the Basis of Machines Which Understand Natural Language", Computers and Thought, Feigenbaum and Feldman eds., 1963.

9. Loveman, D. B., et al: CUE: A Customized System for Restricted, Natural English, Proc. Information Systems Symposium, Washington, D.C., September 1968.

10. Lowe, Thomas C.: Design Principles for an On-Line Information Retrieval System, Ph.D. Dissertation presented to The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, 1966.

11. Raphael, B.: "A Computer Program Which 'Understands' ", Proc. FJCC, pp. 577-589, 1964.

12. Raze, C.: "The FAP Program for String Decomposition of Strings", String Program Reports No. 2, New York University, October 1967.

13. Rubinoff, M., et al: "Easy English, a Language for Information Retrieval Through a Remote Typewriter Console", Comm. ACM, Vol. 11, p. 693, 1968.

14. Sager, N.: Syntactic Analysis of Natural Language, Advances in Computers, Vol. 8, pp. 153-188, 1967.

15. Smith, John M.: An Oral Experiment on Retrieval Dialogue, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, June 1967.

16. Smith, John M.: A Written Experiment on Retrieval Dialogue, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, August 1967.

17. Thompson, F. B.: "English for the Computer", Proc. FJCC, pp. 349-356, 1966.

18. Weizenbaum, J.: "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine", Comm. ACM, Vol. 9, No. 1, January 1966.

PREFACE

The research represented in this dissertation was carried out in response to a problem posed by Dr. Morris Rubinoff of The Moore School of Electrical Engineering of the University of Pennsylvania. His long association with the field of information retrieval has focused his attention on the problem of man-machine communication in an attempt to provide a library-like environment which may be readily learned by users.

The work presented by the author is to be incorporated into the information retrieval system of the Moore School Information Systems Laboratory. This system is presently being programmed for the RCA Spectra 70/46.

CHAPTER 1

INTRODUCTION

As man's knowledge continues to grow at an increasing rate, it becomes ever more desirable that persons in need of information have at their disposal a rapid and accurate method of acquiring it. A computerized information retrieval system seems to offer the best chance of achieving this goal. However, problems are immediately encountered. One important problem is that requests for information from such systems have to be formed in a language that the system can 'understand' but which by and large is quite foreign to the user. Therefore an intermediary is desirable to translate the user's request from his own natural language into the language used by the system. Another important problem is that turnaround time is something less than ideal in most computer systems; also, the user generally does not receive the desired information either because the system loses something in the translation process or the user himself does not have a good idea of what he wants nor means for clarifying his ideas. Furthermore, the system serves only one mode of operation, namely, a document file search, thereby limiting the user's search strategy.

What the searcher needs is a library facility built into the retrieval system in which he can converse with the system as he would with a librarian and can find out how the library is organized, what information can be found, how it can be found, and what to do when in trouble with search procedures. Instead

of putting the user into the difficult position of learning a new language, the system is given the task of responding in and to the user's language, i.e. the man-machine conversation is carried out in a natural language.

To accommodate the different search strategies of many users, the conventional library generally offers not only the main body of documents but also a thesaurus, dictionary, various indexes, and/or citation references. As a step towards this goal in computerized information systems, this dissertation presents a pragmatic translator to enable truly natural language man-machine multi-mode* conversation in an on-line information retrieval system through a remote teletypewriter console. The techniques discussed in this pragmatic translator are collected together under the label 'Real English' and can be incorporated into various types of information systems. To illustrate these techniques, Real English has been designed to be used in the environment established by the Moore School Information Retrieval System. Basically, Real English accomplishes: 1) a syntactical analysis (i.e. parse) of the user's message using a string analysis grammar, 2) use of this parse, and previous dialogue if necessary, to determine what the user wishes, and 3) formation of a series of system commands to fulfill the request.

The structure of a pragmatic interpreter/translator such as Real English depends upon such environmental factors as: 1) the linguistic style of the users, i.e. the actual grammatical

---

* A mode is a variety of conversation associated with a particular data file.

structures most likely to be used as input to the translator, 2) the actual symbolic commands acceptable to the retrieval system, and 3) the computer initiated and directed dialogue occurring as a result of retrieving the desired information. The acceptance of a syntactical structure into the grammar of a syntax analyzer of a pragmatic translator depends upon the preceeding three conditions, whereas the generated list of commands which when executed will fulfill the request depends entirely upon item 2 above.

The formal character of the grammar incorporated into the syntactical analysis algorithm differs from that of phrase structured grammars, i.e. the structural description obtained for a given sentence will be different. It is possible to go from one grammar to the other although the mapping is by no means trivial.

A well developed syntax analyzer program incorporating a significant part of English grammar was available to the author. The structural description provided by this grammar seemed well suited for the construction of the pragmatic interpreter (e.g. the recognition of the information clauses which usually appear as complete adjunct strings is accomplished through the occurrence of specific nodes on the tree). The basic principles of constructing the pragmatic interpreter would be equally applicable if the structural description were of the kind provided by a phrase structured grammar.

Several systems have been developed over the past decade that attempt to retrieve information based upon natural language

requests[4,5,7,8,9,11,18].

The SIR (Semantic Information Retriever) system demonstrates a conversational and deductive ability through the use of a model which represents the semantic content from a variety of subject areas[11]. The data base is highly structured involving binary relations expressed as attribute-value pairs. SIR recognizes only a small number of sentence forms, each of which corresponds to a particular relation. Like Lindsay's SAD SAM system, the data base for SIR is first generated by input sentences and then later queried through natural language statements[8].

In the DEACON system, natural language is treated as a formal language from which a technique developed by Thompson is used to determine the meaning of sentences in that language[4]. Thompson's hypothesis is that "English essentially becomes a formal language as defined if its subject matter is limited to material whose interrelationships are specifiable in a limited number of previously structured categories (memory structures)"[17]. Again, a highly structured data base is used (ring structures in this case) for storing the information.

The CUE (Computer Utilize English) system is a document retrieval system allowing a more varied sentence structure than the other systems[9]. It is one of the few systems that uses a complete syntactical decomposition as a basis for its interpretation. The system by Lamson is predicated on the principle "that syntactic structure can be replaced by a much simpler kind of structure without a great loss of meaning in many technical

specialities"[7]. Similar techniques are used in other systems to avoid the essential syntactical analysis needed to achieve a flexible system.

1.1 Background

As of 1966, the information retrieval system of the Moore School Information Systems Laboratory (MSISL) performed its man-machine interaction by means of a Symbolic Command Language. This language was a formal one requiring the proper syntactical use of left and right parentheses, logical symbols (such as & - and, + - or, ! - and not), and index terms.

Experiments were designed to determine the degree of difficulty the uninitiated user would have in learning to use this Symbolic Command Language oriented system. These experiments showed that persons having little exposure to any form of abstract algebra, logic or programming found the system extremely difficult to operate. Based upon these results, it was decided that instead of putting the user in the difficult position of learning a new language, the system itself would be given this task. To test the feasibility of such a proposal on a small scale, Easy English was developed as the successor of the Symbolic Command Language[2,13]. Easy English, a somewhat restricted but nevertheless real version of English, allowed the user to have his queries, which were written in English, translated into the Symbolic Command Language equivalent at which point the system executed the request.

The success attending subsequent use at the Moore School provided the incentive to develop an information retrieval system with a more sophisticated pragmatic interpreter and translator to serve as the link between man and machine in a conversation-oriented environment. This new translator, Real English, is designed to eliminate many drawbacks of Easy English, namely:

1) Easy English operated only in the search mode, i.e. the user could only query the system's data files which referred to the documents stored. Through written requests in English, Real English will give the user access to definitions with semantic expansion*, a thesaurus, synonyms, and relationships expressed through classification tables.

2) Easy English could correctly interpret only those messages that were written in declarative form; Real English will handle declarative, interrogative, and imperative.

3) The rather crude grammar used in Easy English led to a system not flexible enough to accommodate significant changes in the structure of the user's request.

1.2 Real English System

As illustrated in Figure 1, the Real English System consists of three distinct components: 1) syntax analyzer, 2) pragmatic

---

* See Section 2.3.3 for an explanation of "semantic expansion".

interpreter, and 3) command formatter.

The syntax analyzer requires three inputs: the grammar, the word dictionary, and the input sentence; it produces a tree. The grammar is composed of strings and restrictions. A grammar string consists of definitions of the grammatical entity which the string represents; a restriction consists of tests to determine whether or not a particular definition of the string is valid for the sentence being analyzed. The word dictionary, insofar as the syntax analyzer is concerned, supplies a category list for each word in the sentence. This list consists of all the categories of the word: e.g. noun, verb, etc. The syntax analyzer parses the sentence by constructing a tree (from the string definitions) whose terminal nodes correspond to the categories of the successive words of the sentence.

The pragmatic interpreter or command-set generator determines the actual system commands to be executed in order to fulfill the request. In its task, the pragmatic interpreter uses the tree representation of the sentence developed by the syntax analyzer together with the pragmatic codes* of the various words of the sentence from the word dictionary, and the pragmatic grammar** which is a series of tests upon the tree to determine the commands associated with the given sentence.

---

* 'Pragmatic' codes of a word refer to those codes that are added to a word's dictionary record solely to aid in interpreting the intention of the user's message.

** 'Pragmatic' refers to the grammar used to interpret the 'intended' meaning of the user's message.

The syntax of each generated command is formed by the command formatter. Use is again made of the word dictionary which also holds information relating to conjunctions, bibliographic data (author, editor, etc.), and other considerations explained below.

1.3 Contributions

For the first time, a user of an on-line retrieval system will be able to communicate with a multi-mode system by means of a natural language thereby allowing greater freedom of search strategy. For example, consider a user who may have only a vague idea of his needs in the form of a list of topic areas (i.e. index terms). This user may proceed along several paths in his quest for information:

1) he may extract from a thesaurus terms related to his index set in an effort to narrow or broaden his search area and then query the document file with this newly created list.
2) he may immediately search the file and from the results of this initial search refine his index set for subsequent searches.
3) he may initially search the file using this set of index terms and from the results continue searching from a different path (such as by author or publisher).
4) choosing any of the above paths, he may encounter terms which are unfamiliar to him. In such a case, he may request clarification in the form of a definition or example.

It is to be noted that in the above four search strategies, different modes of system operation are involved:

1) Search mode: the user actually queries the document file for relevant documents pertaining to his needs.
2) Dictionary mode: clarification of uncommon terms by definition or example.
3) Synonym mode: synonyms are supplied for stated terms.
4) Relational mode: terms related to a given list of terms are found. The relation itself may be varied (e.g. whole-part, generic-specific, noun-modifier).

The burden of determining which mode is desired by the user and also the particular command within that mode to be used falls upon the system and is a function of Real English. In this way, the system becomes a library readily accessible through a remote console typewriter.

Also, for the first time, an information retrieval system will be able to use previous dialogue as an aid in resolving ambiguities. Consider the following examples.

Example 1:

	Mr. A	Mr. B
Message 1:	Give me anything written about radar.	What is the meaning of radar?
Message 2:	How about sonar?	How about sonar?

Notice that in both cases, the second message is syntactically identical. However, due to the dialogue involved Mr. A will receive information from the documents stored in the file dealing

with sonar whereas Mr. B will receive the definition of sonar.

Example 2:

Message 1: I want all the papers on cosmic radiation after 1965.

Message 2: Who wrote them?

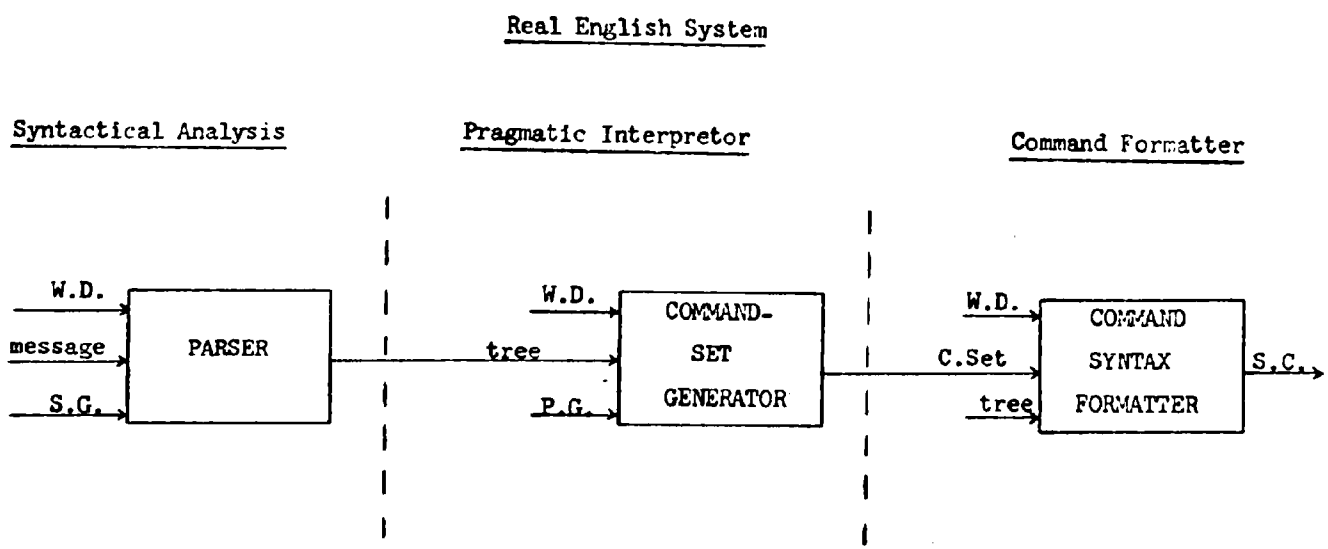
Once again, a follow-up request is based on a previous message and as such the user would receive the authors of all the papers selected by the first message.

In addition, a rather extensive grammar has been altered to meet the special requirements of an on-line information retrieval system. The environment existing with such a system inherently limits the grammatical structures likely to occur. For example:

1) Subjects are likely to be pronouns (e.g. I, You) or index terms (e.g. Jones, Association for Computing Machinery).
2) Verbs are those that occur in requests of various kinds (e.g. like, want, give).
3) Objects are most likely to be simple noun or pronoun phrases whose adjuncts are either verbal phrases, prepositional phrases, THAT - clauses, or adjectival phrases.
4) Strings representing index term sequences (e.g. cosmic radiation, the theory of heat transfer) must be added to the grammar.
5) Incomplete sentences (e.g. Papers by Smith; How about sonar?; Cosmic radiation) must be made acceptable to the grammar.

6) Conjunctions are limited as to the elements they may join. In addition, index terms are more easily associated with the proper sector designator code when they occur as part of a common subtree joining the same node representing the sector designator.

The strong linguistic basis for the syntactical analysis endows the system with the power of flexibility. As experience shows that certain new structures frequently occur and should therefore be a part of the system, they may be incorporated into the system without a major overhaul of the existing procedures. Also, as more commands are added to the symbolic commands of the system, the syntax analyzer provides the keys to determining the various pragmatic codes to be used.



Legend

W.D. - Word Dictionary

S.G. - Syntactical Grammar

P.G. - Pragmatic Grammar

C.Set - Command-Set

S.C. - System Commands to be executed

Figure 1

CHAPTER 2

ENVIRONMENTAL FACTORS

INFLUENCING THE PRAGMATIC INTERPRETER

2.1 Introduction

Inasmuch as Real English serves as a link between the user and the system, it must take into account the behavioral characteristics of each. To understand the user's messages, the system must have a knowledge of what the user is likely to say and also how the system is to respond, i.e. which symbolic commands are to be executed.

2.2 Syntactical Structure

In order to get a feeling for the syntactical structures likely to be used by searchers, various people both inside and outside the University community were asked to prepare written requests for information that they considered to be normally found in a library. Based upon these requests (approximately 100) and their stylistic variations, a grammar was developed. In order to test this initial grammar, both oral and written experiments were performed.

Both experiments were designed to simulate a man-machine dialogue. In the oral experiment, this dialogue was performed via telephone conversation whereas in the written experiment it was carried out through teletypewriter communication.

The purpose of the experiments was to learn how the user phrases his requests and not whether he actually retrieves any

information from his requests. In order to avoid giving the subject any hint as to how a request could be phrased and thus not bias his own natural approach, a bibliography compilation was used. Such a test involved very little additional explanation on the part of the experimenter.[15,16]

The requests obtained from the experiment were tested in the initial grammar and pragmatic interpretor and were found to be eighty percent acceptable. Some sample requests from the experiments are found in Appendix A.

2.3 System Commands

The commands of the system may be divided into groups, or modes, depending on the particular data base(s) involved in their execution. Within each mode of operation there may be any number of commands. The present breakdown of commands follows. Each command's syntax consists of the command's name followed by its specification part.

2.3.1 Search Mode

The search mode makes use of the document and inverted files. The document file contains the bibliographic and subject material relevant to each document in the system. The inverted file lists for each significant word (i.e. index item) in each of the various sections (e.g. author, title, abstract, etc.) of a document, all the document numbers having this particular index item in the given section of the document. The inverted file is used to form a list of accession numbers satisfying a criterion

based upon index terms. (An index term is a sequence of one or more index items.) The document file is used to supply the various bibliographic and subject matter information pertaining to the set of documents supplied by or to the user[10]. The commands of the search mode follow.

NUMBER

The NUMBER command will produce an internal list of accession numbers satisfying the command's criterion of a logical construction of informational clauses. Each informational clause consists of one index term or a logical combination of index terms preceded by a sector designator. The sector designator indicates the particular section of the document being referenced. The user receives the number of references in this internal list, called the Document List. The sector designators are listed below.

AUTH - indicates author section

DATE - indicates date section

TITL - indicates title section

EDIT - indicates editor section

ISSR - indicates issuer section

DESC - indicates subject content section

JOUR - indicates journal section

The informational clauses may be combined by the logical symbols + (inclusive or), & (and), and ! (and not). This is also true of index terms within any one informational clause. Parentheses serve the usual purpose of resolving ambiguities by establishing the desired logical construction.

Example:

1) NUMBER (AUTH SMITH + DESC RADAR & SONAR) & DATE 1967

Any document that was both

1) written in 1967, and

2) either

i) written by Smith, or

ii) pertaining to radar and sonar

will be selected by this request. The user receives the number of such documents selected.

COMBINE

The COMBINE command performs a combinatorial search of the inverted file based on the informational clauses appearing in the command's specification part. Each information clause may contain only one index term as no logical symbols are used in this command. The informational clauses are separated by slashes (/). For each i, i = 1 to n where n is the number of index terms, the user receives the number of documents indexed by exactly i of the n index terms.

As an example consider:

COMBINE AUTH SMITH/ DESC RADAR/ SONAR/ DATE 1967

[Note: SONAR will be regarded as having the sector designator DESC.]

The user will receive information as listed below.

x documents retrieved indexed by exactly 4 of the terms.

y documents retrieved indexed by exactly 3 of the terms.

z documents retrieved indexed by exactly 2 of the terms.

w documents retrieved indexed by exactly 1 of the terms.

It is possible to limit the output to those lines desired. This is done by a range specifier placed immediately after the command name. The range specifier is part of the s- -fication part.

The range specifier consists of line specifier expressions joined by an A (representing the logical 'and') or O (representing the logical inclusive 'or'). A line specifier expression consists of quantifiers followed by number m, where m ≤ n. The quantifiers are:

G - greater than

L - less than

E - equal to

LE - less than or equal to

GE - greater than or equal to

The entire range specifier is enclosed in parentheses.

As examples consider:

1) (GE2AL4)

The user has limited the output to those documents indexed by exactly 2 or 3 of the command's index terms.

2) (203)

The user gets the same results as in 1.

3) (L5)

The user receives the information about the documents indexed by exactly 1,2,3 and 4 of the given index terms.

As an example of a complete COMBINE command, consider

COMBINE (GE 3) AUTH SMITH/ DESC RADAR/ SONAR/ DATE 1967

The user receives:

x documents retrieved indexed by exactly 4 of the terms.

y documents retrieved indexed by exactly 3 of the terms.

FORM

The NUMBER and COMBINE commands discussed above form lists of document numbers. Further dialogue determines the particular sections of each document desired by the user. The lists so formed were based on the index terms appearing in the specification part of the command.

The FORM command forms a list of accession numbers from those that appear in the specification part. Any number (≥ 1) of accession numbers, separated from each other by a comma, can comprise the specification part.

Example

FORM 2,1763,576

Bibliographic Commands

Each section of a document has its own corresponding command so as to enable that piece of information to be extracted from each document listed in the Document List. Each of the following commands will return to the user the indicated information for each document in the Document List.

AUTH

TITL

DAT

EDIT

ISSR

JOUR

DESC

In addition, DESC/BIBLIO and DESC/ALL will return, respectively all the bibliographic information and all the information of each document in the Document List.

2.3.2 Thesaurus Mode

The Thesaurus Mode commands make reference to the lexicon file which is an alphabetical listing of all words relevant to the field of discourse of the information retrieval system. In all the following commands, α and β are strings of letters.

THES/X - Returns to the user a given number of lexicon words, each beginning with the letter string α, as in*

THES/X α

THES/BF - Returns to the user a given number of lexicon words alphabetically before the letter string α, as in

THES/BF α

THES/AF - Returns to the user a given number of lexicon words alphabetically after the letter string α, as in

THES/AF α

THES/AR - Returns to the user a given number of lexicon words alphabetically surrounding the letter string α, as in

THES/AR α

---

* The actual number of words retrieved is established by the system administrator.

THES/BT - Returns to the user a given number of lexicon words alphabetically between α and β, as in

THES/BT α, β

These commands will accept any number of α's separated by commas. (The THES/BT command will accept any number of pairs of letter strings.) Examples are:

THES/X α_1, α_2, α_3

THES/BT α_1, β_1, α_2, β_2

2.3.3 Define Mode

Execution of the commands in this mode makes use of the DEFINE file whose records consist of a definition and several levels of semantic expansion for each word of the file. Semantic expansion is a tool to provide explanations and illustrations at several levels of detail which serve to instruct the searcher on the meanings of words and their use in the system. The definition of the unknown term may be considered first-level expansion. Further levels of expansion would be successively a one paragraph description, an example, and finally a fully detailed illustration. See Appendix B for an example of semantic expansion.

The specification part includes a level specifier and a series of words separated by commas. The level specifier is a series of level numbers separated by commas and enclosed in parentheses. The level numbers refer to the level of expansion desired for each word listed. If the level specifier is omitted, the desired level is assumed to be one.

DEFINE (1,2) RADAR

This command will return the first two levels of semantic expansion of the word 'RADAR' to the user.

2.3.4 Relation Mode

The commands of the Relation Mode make use of binary relations stored in the Relation Mode File. These relations may be generic-specific, noun-modifier, whole-part, word-words of its definition. Only the generic-specific is used in this system.

RELATION Command

The specification part contains a series of words separated by commas preceded by a relation specifier which is a series of numbers each separated by a comma and enclosed in parentheses. The numbers of the relation specifier indicate the particular relations defined for each word listed.

Example:

RELATION (8) AUTOMOBILE

Assuming 8 indicates the relation 'is generic to', the execution of this command would give the user all terms generic to automobile. The absence of the relation specifier would indicate that all relations be applied to the list of words.

2.4 System-Directed Dialogue

It is to be recalled that the execution of the NUMBER and COMBINE commands of the search mode result in an internal list of accession numbers being formed and the user being informed of the number of documents so listed. If these commands are not

followed by a bibliographic command, the system initiates a computer-directed search to ascertain which sections of these documents is of concern to the user. Note that the user, with Real English, has the choice of bypassing this dialogue by specifying the desired sections in his request. This dialogue makes the user aware of the various sections of the documents. It also influences the grammar of the system since it also has the purpose of limiting the user's messages by eliminating a call to the EXPLAIN mode in the event the user would not know what to do after he got the number of references satisfying his initial NUMBER or COMBINE command.

CHAPTER 3

SYNTACTICAL ANALYSIS

The syntactical decomposition of the user's message is performed by means of string analysis. This analysis characterizes the sentences of a language as consisting of one elementary sentence (its center), plus zero or more elementary adjuncts, i.e. word-sequences of particular structures which are not themselves sentences and which are adjoined immediately to the right or left of an elementary sentence or adjunct, or of a stated segment of an elementary sentence or adjunct, or of any one of these with adjuncts adjoined to it [6]. An elementary sentence or adjunct is a string of words, the words (or particular sequences of them) being its successive segments.

The particular syntactical analysis performed in the Real English system is based upon one that was initially developed at the University of Pennsylvania under the direction of Professor Z. Harris and continued at New York University by Dr. N. Sager. This grammatical analysis consists of three separate programs [12]:

a - the program to generate the word dictionary records

b - the program to generate the grammar records

c - the program which analyzes sentences using the word dictionary, grammar, and input sentence as inputs

The grammar is composed of strings and restrictions. A grammar string consists of definitions of the grammatical entity which the string represents; a restriction consists of tests to

determine whether or not a particular definition of the string is valid for the sentence being analyzed. The word dictionary supplies a category list for each word in the dictionary. This list consists of all the categories of the word: e.g. noun, verb, etc. The sentence analyzer parses a sentence by constructing a tree (from the string definitions) whose terminal nodes correspond to the categories of the successive words of the sentence.

In the context used here the syntactical analysis is not an end in itself as it is at N.Y.U. under Dr. Sager, but only a means to an end; i.e. the tree produced to represent the parse of the sentence is used by the system to determine the necessary system commands to be executed to fulfill the user's request, and to form the proper syntax for each such command.

Because of the different environment in which the parser is to be used in the Real English system, several changes in the N.Y.U. grammar were carried out. The N.Y.U. grammar is highly sophisticated as to the type of sentences it can properly analyze. This degree of sophistication was unwarranted in the present application because the linguistic structures likely to be encountered were restricted by the environment in which the grammar is used. In addition, certain sentence structures that are likely to occur in the Real English system (e.g. the conversationally-dependent sentences discussed in Section 4.2.4) are excluded in the original grammar. Because the Real English grammar is written so as to produce a parse that would lend itself to relatively easy interpretation, the restrictions used

in the treatment of conjunctions, adjuncts, and objects have been changed. In addition, restrictions were added to aid in the proper alignment of index terms with their associated grammar string.

The pragmatic interpreter which tests that certain conditions prevail with respect to the tree, has been incorporated into the system as grammar restrictions. The logic involved in the pragmatic interpreter involves procedures to scan the tree, grammar, or sentence list. These procedures have already been, for the most part, established for use by the syntactical analyzer, so that coding the pragmatic interpreter in the form of a restriction list which will call upon these established routines will result in magnetic core memory efficiency. In addition, the grammarian may easily change this logic without altering any of the established programs.

3.1 Word Dictionary

The words are arranged in the dictionary as follows: 1) a word is placed into one of fourteen groups, i.e. Group 1 contains words of one or two characters, Group 2 contains words of three or four characters, etc., 2) within each group the words are put into lexigraphical order according to their numerical representation as individual characters. A word W has a set of category assignments C_1 or C_2 or ... or C_n, where each C_i corresponds to a part of speech, a particular word, or to a special condition. C_i is the same as the name of an atomic string in the grammar. The C_i's form a category list C. Each category may have a set of subcategories. Therefore C consists of L_1C_1 or

L_2C_2 or ... or L_nC_n where L_i is C_{i1} and C_{i2} and ... and C_{im}. L_i is a set of properties of W for the C_i category.

3.2 Grammar Strings

A grammar string S is defined to be S_1 or S_2 or ... or S_n. The S_i's are called the options of S. The S_i's are defined to be S_{i1} and S_{i2} and ... and S_{im}. The S_{ij}'s are the elements of the i^{th} option and are themselves grammar strings like S. Therefore, we have a system of strings composed of other strings. However, the system is not infinite. The process ends with atomic strings. An atomic string is a symbol which is not composed of any other string; it corresponds to a word category whereas the non-atomic string represents a broader grammatical entity.

To allow for a more refined and compact grammar, restrictions were added to the options of a string. The complete definition of S is actually R_1S_1 or R_2S_2 or ... or R_nC_n, where R_i is a series of tests to be performed upon the sentence or tree. If R_i fails, then S_i is not a proper choice for S.

The strings are represented in the machine as lists. The first word in the list of a string S is the head of S which contains its code name S and certain properties of S. The second word points to R_1S_1, the first option of S and its restriction. In general the $(n+1)^{st}$ word of S points to R_nS_n, the n^{th} option and its restriction. The options are also lists. The list of each option S_i is similar to the list of a string except that it has no head. The first word of S_i points to the head of S_{i1}, the first element of the i^{th} option of string S. In general the n^{th}

word of S_i points to the head of S_{in}, the n^{th} element of S_i. There is a slight difference in atomic strings. The head of the atomic string has an atomic signal and the string is composed of only a head.

3.3 The Tree

The tree is constructed by the analyzer program from the grammar strings. It is actually a record of the options and their elements which the program has chosen from the definitions of the strings.

Each unit of the tree is called a node and corresponds to an element of an option. The node N takes up eight half-words in memory. N consists mainly of pointers: an "UP" or "LEFT" pointer, a "DOWN" pointer and a "RIGHT" pointer which point respectively to the node above or to the left of N, to the node below N and to the node to the right of N. N also has a "GRAMMAR" pointer whose function will be explained later.

Suppose the program has just attached a node NS corresponding to a string S and must now look at the definition of S and choose an option from it. Suppose the option S_i (which consists of elements $S_{i1}, S_{i2}, \ldots, S_{im}$) is the correct choice for S. Nodes $NS_{i1}, NS_{i2}, \ldots, NS_{im}$ will be attached to the tree in the following manner:

1) NS will have a "DOWN" pointer to NS_{i1}.
2) NS_{i1} will have an "UP" pointer to NS and a "RIGHT" pointer to NS_{i2}.
3) NS_{i2} will have a "LEFT" pointer to NS_{i1} and a "RIGHT" pointer to NS_{i3}.

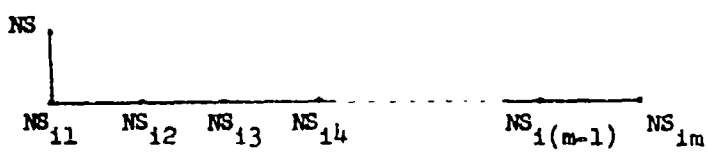
4) NS_{im} will have a "LEFT" pointer to $NS_{i(m-1)}$ but no "RIGHT" pointer.

5) Each node will have a "GRAMMAR" pointer to set up the correspondence between NS_{ij} and:

   a) S_{ij}, i.e. the name of the corresponding string

   b) its place in the definition of S, i.e. it occupies the ij^{th} position in the definition of the parent node of S.

If we draw the tree starting from S, it will look like:



The part of the tree under NS is called the substructure of NS. NS_{i1} through NS_{im} are all one level below NS; hence, NS is their parent(node). NS_{i1}, however, is the only node directly below NS. The node structure via the position and grammar pointers shows the particular option of S chosen during the analysis. The context in which S was chosen can be ascertained by looking at the parent node of S. If in the course of building the tree it were found that the particular option used for S is incorrect, the tree would point (via the grammar pointer) to the place in the grammar where the choice was made. This would enable another option for S to be chosen and a different substructure for NS to be constructed.

Since each NS_{ij} corresponds to a string, it will in general have a substructure similar to that of NS shown above. However, if S_{ij} is atomic it cannot have such a substructure since it is a symbol and does not consist of strings. It corresponds to a part of speech, and the current word W must have a category matching S_{ij}. If W does, then NS_{ij} is complete, and it corresponds to the analysis of W. If there is no match, the choice was incorrect and a different substructure would have to be built for S, the parent node. When the tree is complete for a sentence, it means all the branches are complete and all the words of a sentence correspond to some atomic node of the tree via the matching process. Thus, the tree represents a parse of the sentence.

3.4 The Restrictions

A restriction is a series of routines with their arguments which operate in the tree or any of the lists (grammar, word dictionary, or sentence lists). The restrictions are part of the grammar and therefore determined by the grammarians. By means of these routines the tree or list structure may be examined for different properties, e.g. wellformedness of substructures. A restriction is itself represented in the machine as a list. Each routine in the restriction is executed in order; if any routine in the list fails the restriction fails.

To give some idea of the versatility of the routines which make up the restrictions, they are listed below according to their group identification. For more details about any particular routine, refer to Appendix C.

1) Major Routines - A restriction list composed of these routines is the restriction R_1 found on the option S_1 of string S.

   WELLF, SUBJR, VERBR, MARK, DSQLF,

   SPECF, CHECF

2) Logical Routines - These routines perform logical tests or operations upon other restriction lists.

   TRUE, AND, OR, IMPLY, CANDO, NOT, COMMN,

   ORPTH, ITER, EXPNT

3) Climbing Routines - These routines move around the tree or a list.

   UPONE, UPTRN, UPTO, LEFT, RIGHT, DOWNl,

   DOWN, DWNTO, DNTRN, DNRIT, NEXTL, PREVL,

   ATTRB, INTOL

4) Property Routines - These routines test certain properties of the tree, list or sentence.

   NOATM, EMPTY, ISIT, ATTRB, FIND, PARSE,

   WORDL, SENTL, RARE, REP

5) Tree to List Routines - These routines use the tree to get to a list.

   FRSTL, LASTL, INTOL, EXEC, ATTRB, NEXTL

6) List Producing Routines - These routines work on the tree or a list to produce another list.

   GENER, EDIT, SPCFY, SCOPE

7) Register Routines - These routines use certain "registers" to save and restore nodes or list words.

   STORE, LOOKT

8) Pragmatic Routines - These routines have been added to the original analyzer system to aid in the pragmatic interpretation of the sentence.

BITIN, BITT, FETCH, FILLIN, NEXTAT, PLACE

SETT, TSET

3.5 Cross-Reference Table

Since each grammar record is independent of the others the strings must be linked by a common linkage table, called the cross-reference table. Each string must have its name in the common cross-reference table so that it can be referenced by other strings. It is also advantageous to have independent records for restrictions or lists that are used frequently. If the name of a restriction or list is placed in the cross-reference table, a separate grammar record may be set up for it and may be referenced by any other record.

3.6 Examples of Grammar Record, Word Record and Parse

As an illustration of a grammar record consider that of COB shown in Figure 2. Each entry in the record is made up of three distinct fields: location, operation, and variable field, respectively. The DEFIN in the operation field indicates that the options for the grammatical entity named in the location field are expressed in the variable field. The PATH entries indicate that the location field names a restriction list made up of the routines with their arguments that appear in the variable field. The various options are enclosed in parenthesis. Within each option, the elements are separated by commas. Therefore there are

four options for the center string, COB. These are, respectively, C1 - the declarative string, COD - the question strings, C3 - the imperative string, and C4 - the conversationally dependent strings. Each of the first three options has a restriction. If the function of each routine of each restriction list were gotten from Appendix C, it will be seen that restriction list CO.10 checks that in order to try the declarative option (C1) there must be no question mark in the sentence; restriction list CO.3 checks that in order to try the question strings (COD) there must be a question mark in the sentence; and finally restriction list CO.20 checks that in order to try the imperative string (C3) there must not be a question mark in the sentence and furthermore there must be an untensed verb mark (A30) on the main category list of some word of the sentence.

For a correspondence between the symbolism expressed in Section 3.2 and the above example, note:

S_1 = (C1) $\qquad$ R_1 = CO.10

S_2 = (COD) $\qquad$ R_2 = CO.3

S_3 = (C3) $\qquad$ R_3 = CO.20

S_4 = (C4)

also,

$$S_{11} = C1,\ S_{21} = COD,\ S_{31} = C3,\ S_{41} = C4$$

The word dictionary record of <u>written</u> will serve to illustrate the discussion of Section 3.1. Considering Figure 3, the main category list of <u>written</u> contains one element, A32 - past participle (or Ven category). The sublist of A32 contains subcategory

```
*       COB     CENTER STRINGS
COB     DEFIN   ((CO.10,(C1),CO.3,(COD),CO.20,(C3),(C4)),T
CO.10   PATH    ((DSQLF((CO102,$X1))))
CO101   PATH    ((WORDL)(ITER(CO104,RNEXTL)))
CO102   PATH    ((NOT(CO101)))
CO.3    PATH    ((DSQLF((CO101,$X2))))
CO.20   PATH    ((DSQLF((CO102,$X1)(CO201,$X3))))
CO201   PATH    ((WORDL)(ITER(CO202,RNEXTL)))
CO202   PATH    ((INTOL(CLSSL(A30)))
CO104   PATH    ((INTOL(CLSSL(A49)))
        END     COB
```

Figure 2

```
        WORD            WRITTEN
        LISTIS          (.1,A32)
.1      LISTIS          (.2,BO,.3,PO,BVC)
.2      DEFOBJ          ((B2))
.3      DEFOBJ          ((A60))
        END             WRITTEN
```

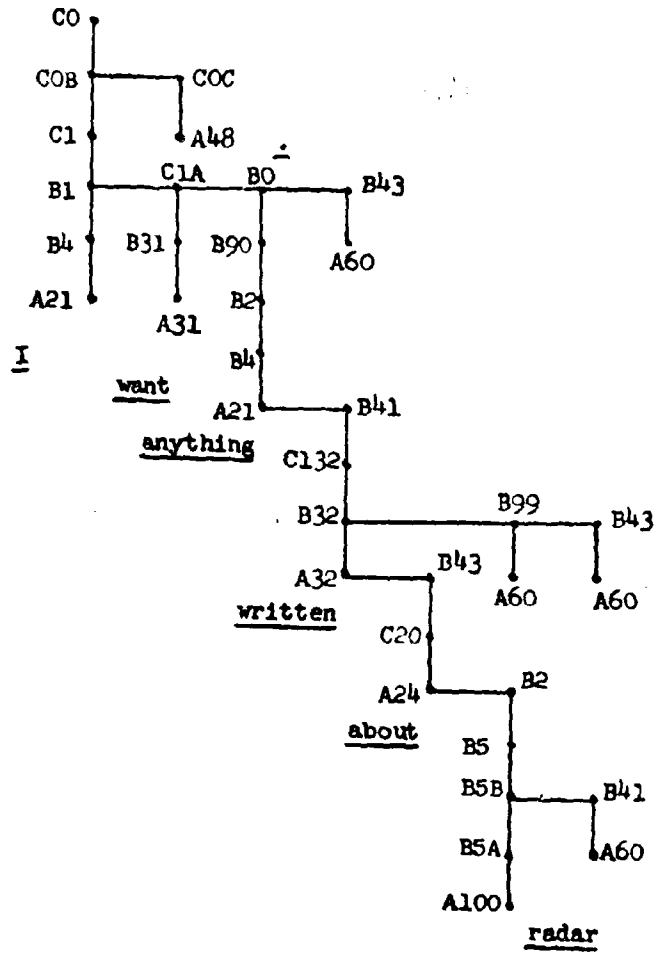
Figure 3

information about the word as a past participle, e.g. the active objects (sublist of PO) are expressed through the (B2) option, the passive objects (sublist of PO) are expressed through the (A60) option, and in addition the BVC symbol indicates that written is a bibliographic verb, i.e. the verb can be used to indicate an index term's bibliographic status. Note that since no verb can take all the possible object strings in the English language, those object strings that are permissible are listed in the verb's word dictionary record so as to lower the parse time.

Consider the parse shown in Figure 4. The nodes represent grammatical entities as described in Figure 5. It can be seen that the particular option chosen for any string may be obtained by looking at the nodes one level below the node in question. For example, the option chosen for C1 is (B21,B1,B21,C1A,B21,BO, B43,B21).

From the parse it can be seen that:

1) the user's message was a declarative statement as the COB (center string) string has the C1 option. The C1 as seen from Figure 5 is the declarative string.
2) the subject of the sentence (the B1 node) is a pronoun as is seen by the A21.
3) the verb of the sentence (the C1A node) is a tensed verb.
4) the object of the verb is the pronoun string: 'anything written about radar'.



Parse of: 'I want anything written about radar.'

Figure 4

SYMBOL	STRING NAME
A21	Pronoun
A24	Preposition
A31	Tensed verb
A32	Past participle form of verb
A48	Period
A60	Empty string. Not all the elements of an option are required to correspond to some words of the sentence. When a sentence occurs in which these elements do not have any correspondence with words of the sentence, they are satisfied by the empty string.
A100	Index item
B0	Object string
B1	Subject string
B2	Noun strings as object
B4	Pronoun string
B5	Index term sequence with right adjuncts
B21	Sentence adjuncts
B31	Tensed verb with adjuncts
B32	Past participle with adjuncts
B41	Right adjunct of noun phrase
B42	Right adjunct of auxiliary words (e.g. may, can, would)
B43	Right adjunct of verb

Figure 5

Figure 5 (Con't.):

SYMBOL	STRING NAME
A63	Left adjunct of verb
B66	Left adjunct of preposition
B69	Left adjunct of pronoun
B90	Active object strings
B99	Passive object strings
C0	Superstring - the root of every tree
C0A	Introducer
C0B	Center string
C0C	End mark
C1	Declarative
C2	Yes-no question string
C3	Imperative
C4	Conversationally-dependent sentences
C20	Prepositional phrases
C1A	Verb
C132	Ven plus passive object
B5A	Index item
B5B	Index term

Note the choice that was made as to the placement of 'about radar'. It appears as a prepositional phrase (C20) acting as a right adjunct (B43) of the verb 'written'. It might be argued that this prepositional phrase should be the passive object of 'written' or placed with the post-object B43. The decision is based upon interpretation ease and will be further explained later.

3.7 Grammatical Considerations

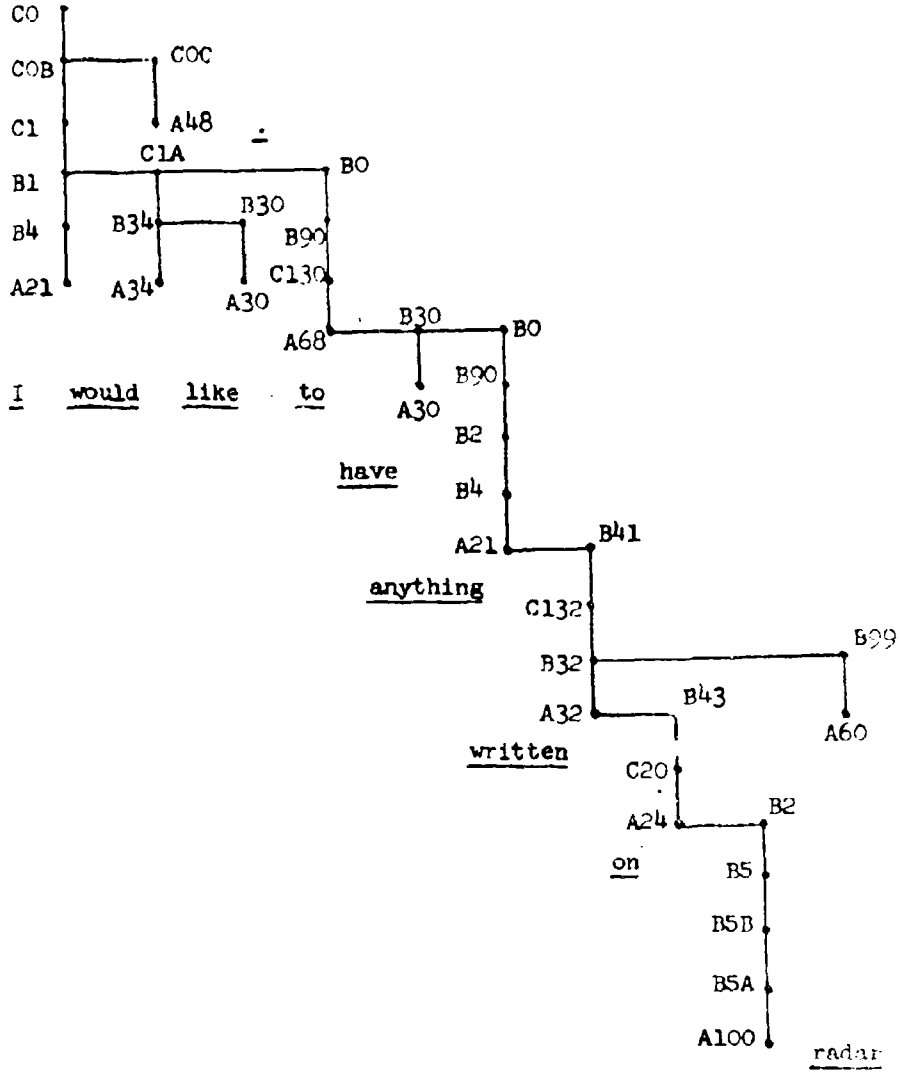
In the Real English system the parse produced by the syntax analyzer is not the end result but only the starting point in the pragmatic interpretation of the user's message. The grammar used by the parser is written with this in mind. When a choice can be made concerning, for example, the placement of adjuncts, the decision is based upon the resulting ease of interpretation and consistency with the remainder of the system. As will be seen in Chapter 4, wherever practical the pragmatic interpretor treats the user's message as if it were of the form "I want ... ". The missing part is filled in with what is called the ultimate object of the sentence. This ultimate object may be viewed as the starting point in the sentence of the informative material. In making this transformation, care must be taken so that the ultimate object appears in the parse as an object.

The discussion to follow, which deals with various considerations in developing the grammar, finds most of its applicability in the search mode of operation as the examples will illustrate. It is to be noted, at this time, that search mode queries may be looked upon as a series of clauses each of which references a

bibliographic piece of data. The grammar will cause the parser to produce a tree that clearly shows these inherent references. Each such clause will contain index terms of some kind together with a distinguishing verb to determine the type of index term (i.e. author, title, etc.).

3.7.1 'Aspectual' Verbs

Aspectuals, for example, like, want, wish, desire, have the property of taking another verbal phrase as object and then having this verb contain an object that could have been the object of the aspectual verb. In other words, the aspectuals act as meta-verbs of the lower level verb that contains an object common to both verbs. If this second verb is itself an aspectual, the process could repeat. The final object of this sequence could be substituted into the message "I want ... ", and then treated insofar as the interpretation is concerned, as the original message. This final object referred to above is the 'ultimate object' of the sentence. Note that the ultimate object has brought into focus that part of the sentence containing the informative material as far as command recognition is concerned, and that indeed the ultimate object is an object string. Each aspectual has a list (ultimate object list) of object strings that could cause it to behave aspectually. The verb comprising this object string must have the proper subcategory necessary to accept the string as an intervening object string.



Parse of: 'I would like to have anything writt[illegible] on radar.'

Figure 6

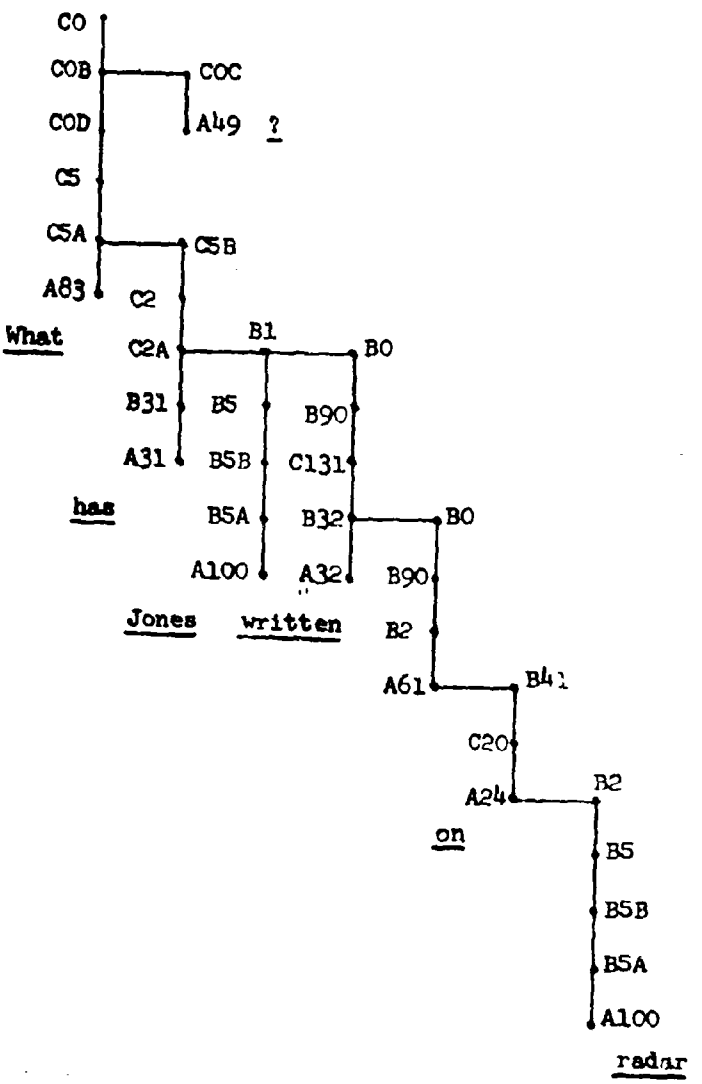
As an example, consider "I would like to have anything written on radar". The parse of this sentence shown in Figure 6 reveals that the object of like is the C130 string 'to have ... '. It happens that indeed C130 is on the ultimate object list of like and also that have is an acceptable verb for C130 to act as an intervening object string. The object of have, B2, is not on the ultimate object list of have and as such is the ultimate object of the sentence.

3.7.2 Omission

Consider: "What has Jones written on radar?" This message is syntactically divided into the word what and the C2 (yes-no question) string with a missing (or omitted) noun as indicated by the A61 node as the object of written (see Figure 7). This inner C2 string with omission may be derived from 'Has Jones written $\bar{N}$ on radar?', where what replaces $\bar{N}$.* In any event, a transformation on this C2 string will transform the given sentence into the desired form of 'I want (ultimate object)'. In the case of the original sentence, the ultimate object is '(omission) on radar'. Therefore the pragmatic interpretor would consider the given sentence as 'I want $\bar{N}$ on radar'. The apparently lost information expressed by 'has Jones written' is acknowledged by recognizing Jones as an author (this recognition process will be explained in Chapter 4) and storing this information prior to the start of the

---

* The N represents a noun or pronoun phrase.



Parse of: 'What has Jones written on radar?'

Figure 7

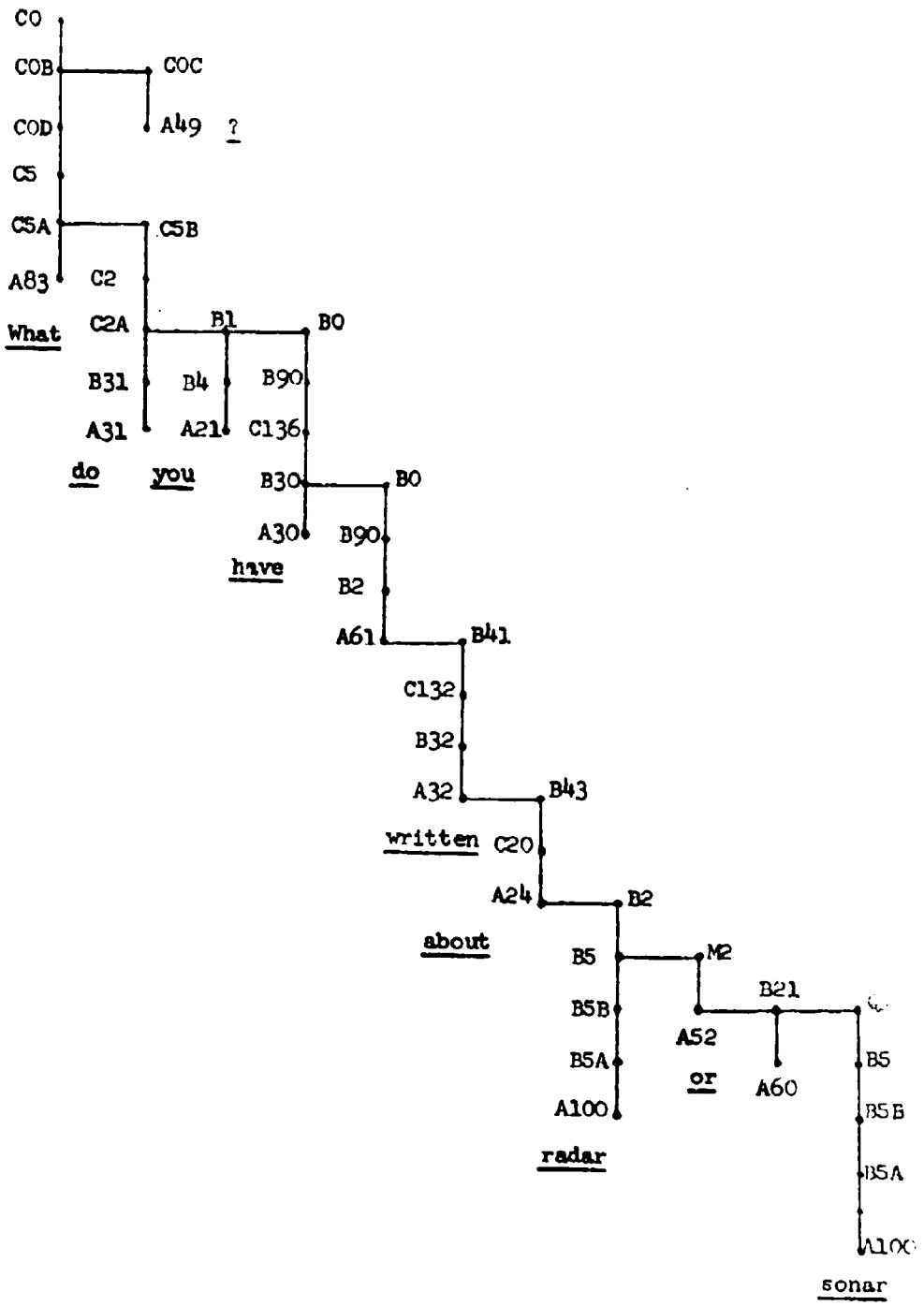
pragmatic interpretation based upon 'I want $\bar{N}$ on radar'.*

Note that the parse did separate the two informative clauses, namely, 'has Jones written' and 'on radar' instead of treating the prepositional phrase 'on radar' as a right adjunct of written as was done in Figure 4. The problem of establishing the correct division among informational clauses of a sentence and having the proper node used as the ultimate object is handled by the joint grammar definitions and restrictions of the right adjuncts of verbs, (i.e., B43), and the omission option of the object noun strings (B2). A few examples will help illustrate the principles. Refer to Figures 8-12 for the respective parse of sentences 1-5 below.

1) What do you have written on radar or sonar?
2) Give me something that has been written by Jones on radar or sonar.
3) Give me something that Jones has written on radar or sonar.
4) I want all the papers you have on radar or sonar.
5) What have you on radar or sonar?

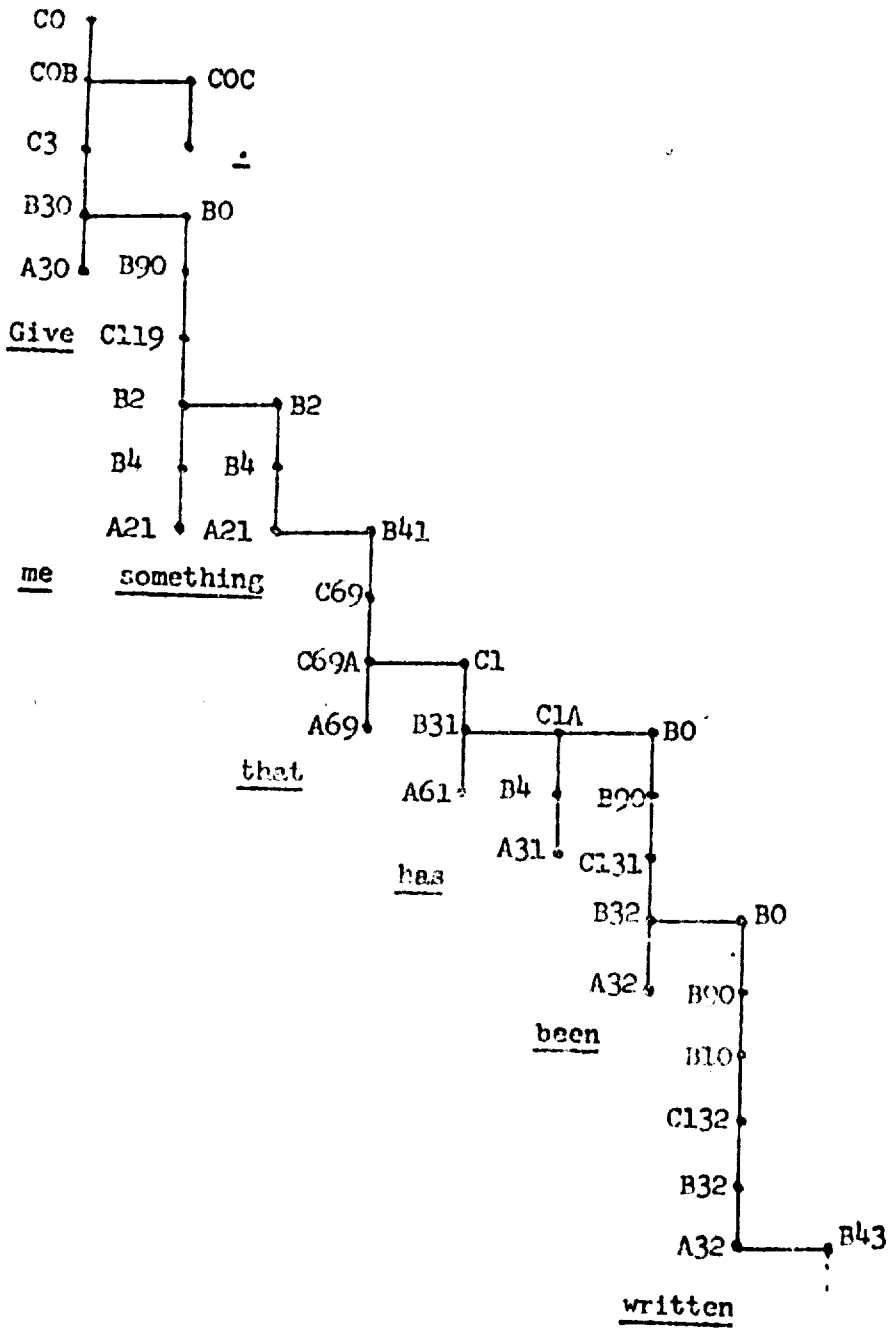
Sentence 1 is derived from 'Do you have anything written on radar or sonar?' and as such have should have an omitted object whose right adjunct is 'written on radar or sonar'. Comparing sentences 2 and 3, one notes that both have two informational clauses the second of which is identical. The first informational

* The pragmatic interpretor considers the noun of $\bar{N}$ as indicative of no particular system file. Therefore $\bar{N}$ may be replaced by anything or something.



Parse of: 'What do you have written about radar or sonar?'

Figure 8

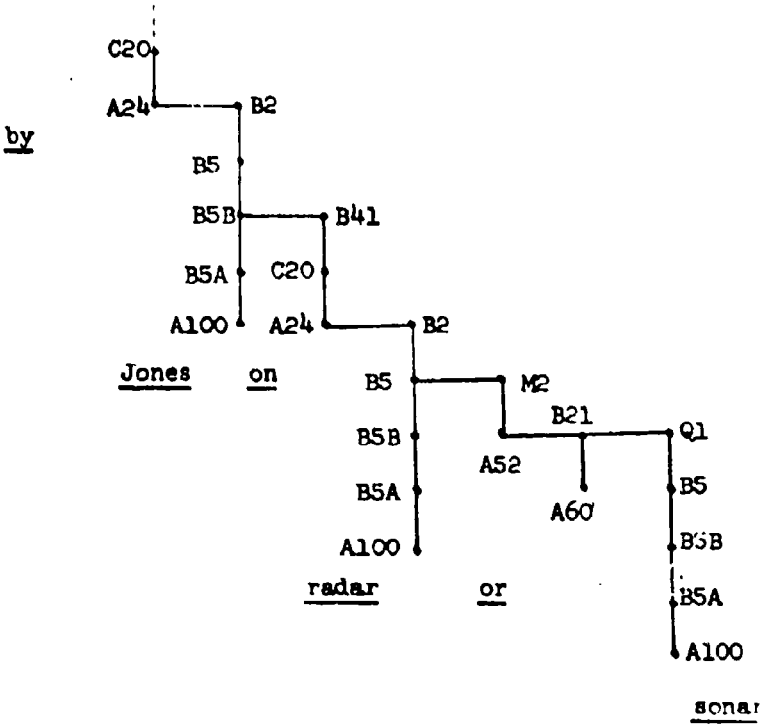


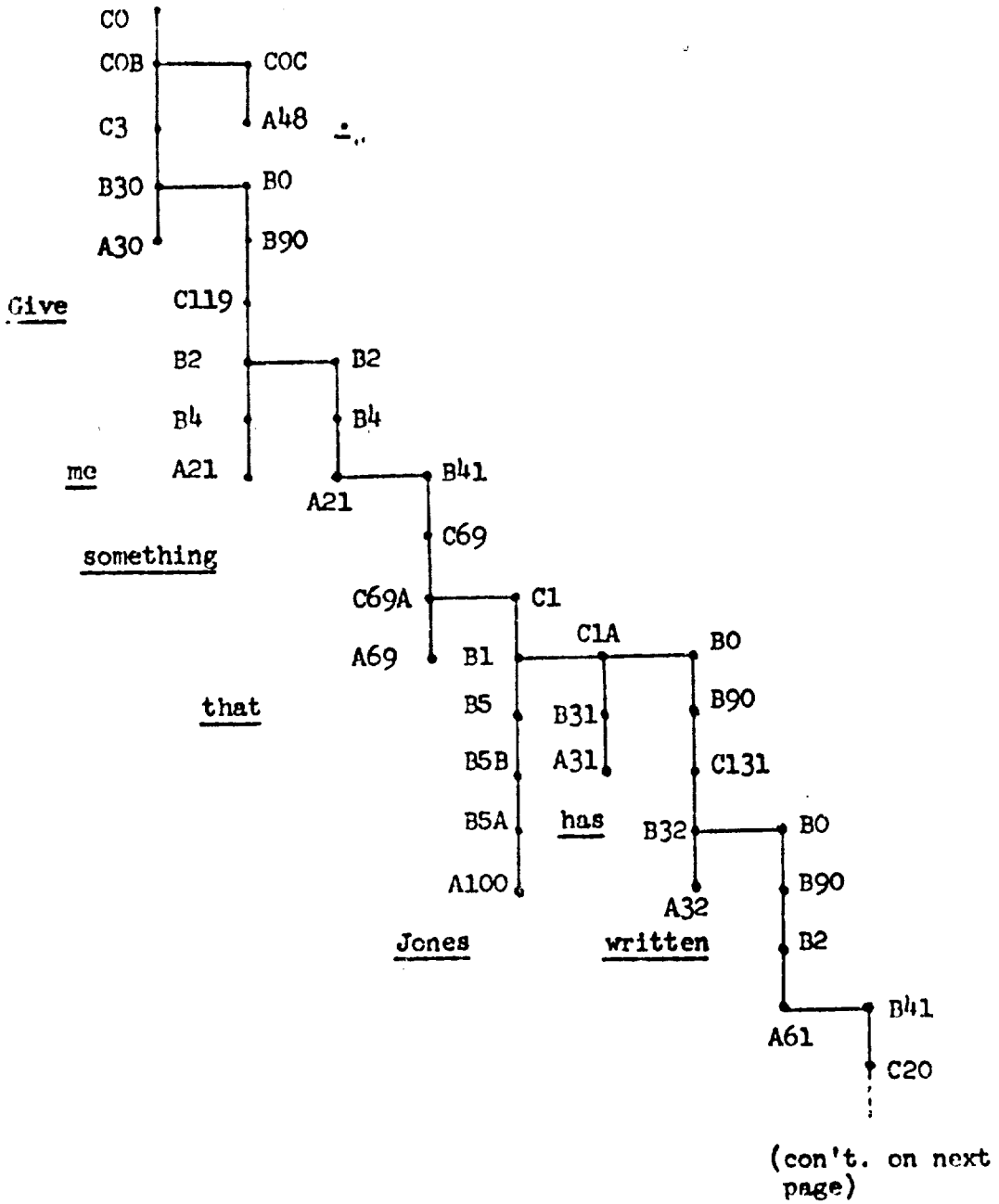
(con't. on next page)

Parse of: 'Give me something that has been written by Jones on radar or sonar.'

Figure 9

Figure 9 (con't.):



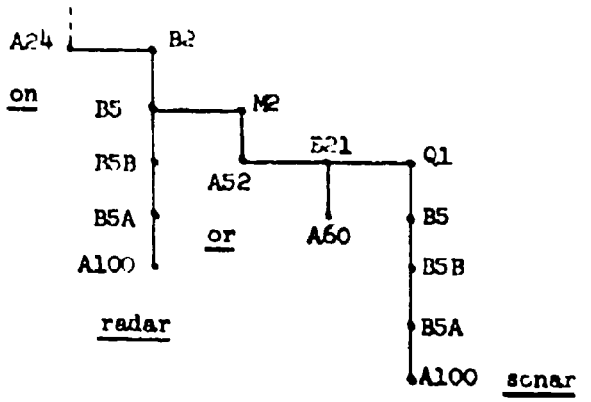


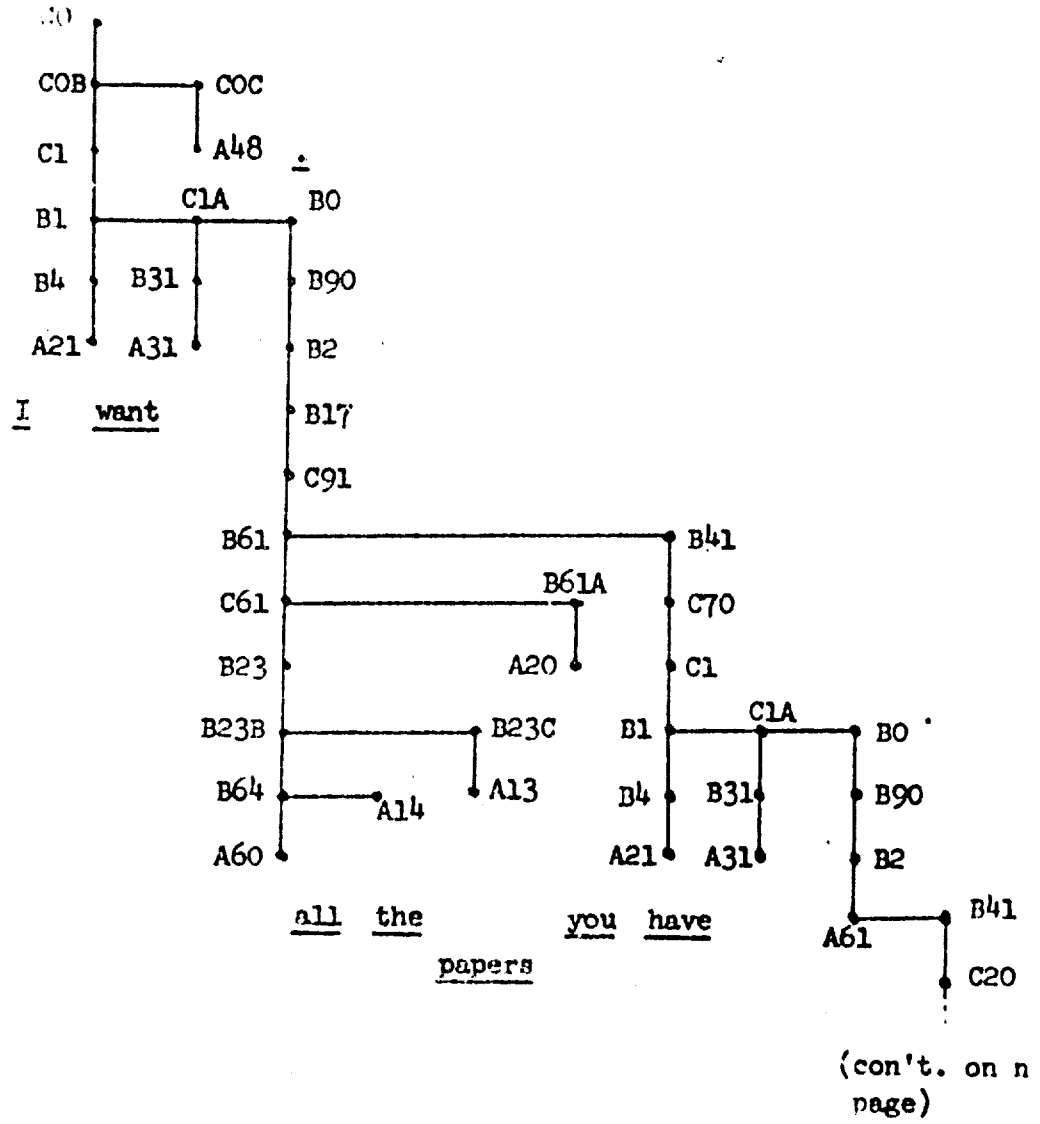
Parse of:

'Give me something that Jones has written on radar or sonar.'

Figure 10

Figure 10 (con't.):

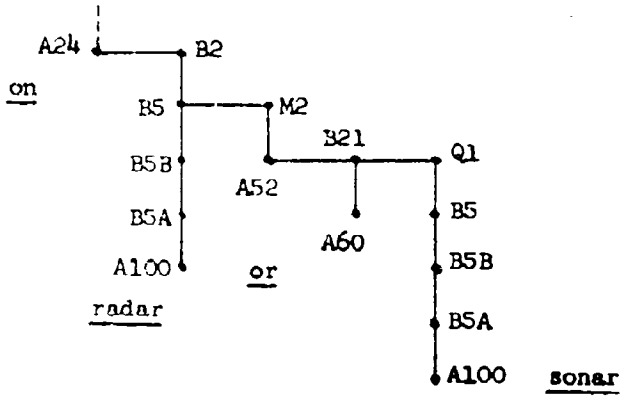


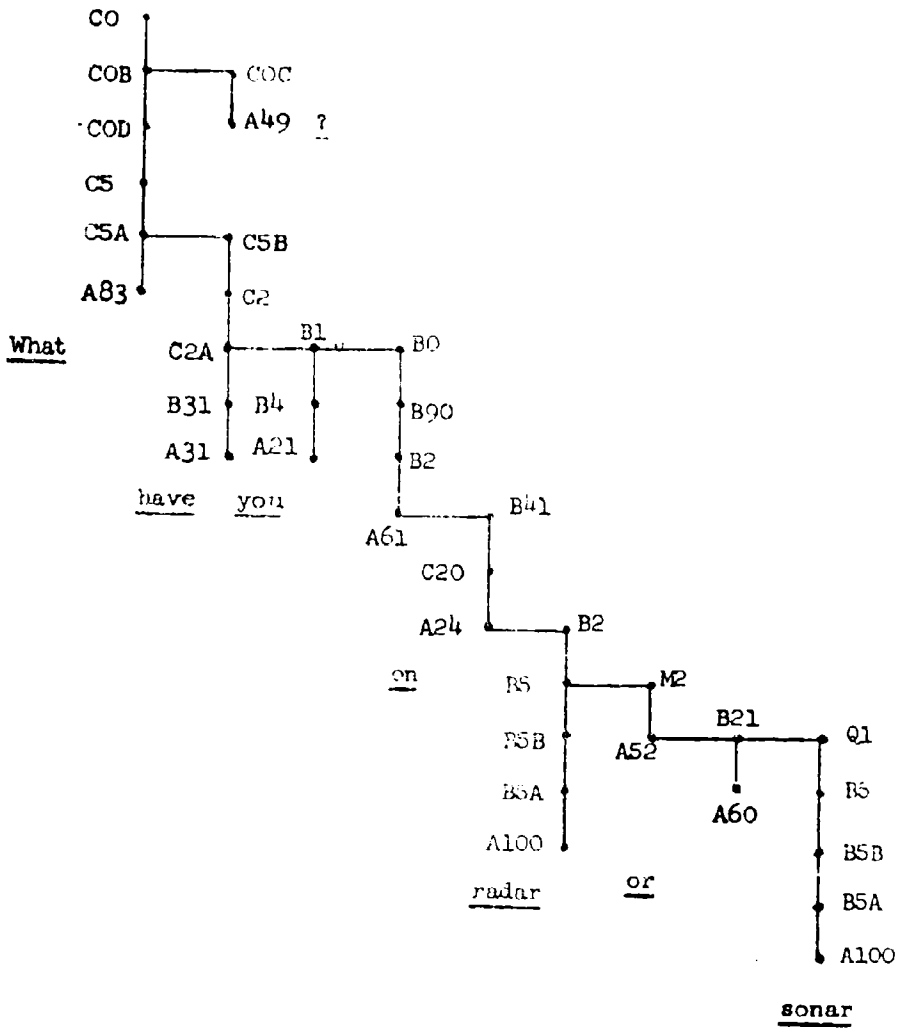


Parse of: 'I want all the papers you have on radar or sonar.'

Figure 11

Figure 11 (con't.):





Parse of "What have you on radar or sonar?"

Figure 12

clause of each sentence is a C69 (i.e. THAT + C1 with an omitted noun). In sentence 2, the verbal construction is passive and as such the index term follows the verb. Therefore the prepositional phrase (C20), 'by Jones', is associated with 'written'. In sentence 3, the verbal construction is active and as such the index terms, if any, precede the verb. Therefore the prepositional phrase 'on radar' is not associated with written but instead forms its own informational clause. Sentence 4 has a C70 (C1 with an omitted noun) as a right adjunct of papers. This C70 is derived from the complete C1: 'you have something on radar or sonar'. Therefore the object of have is again the omitted noun with 'on radar or sonar' as its adjunct. Sentence 5 is similar to the first sentence in that the object of have is the omitted noun.

Listed below are the restrictions on B43 and the omitted option of B2 that accomplish the above divisions.

B43 Restrictions

1) Only bibliographic verbs in the past participle form (i.e. BVC on the sublist of the A32 category) may have non-zero right adjuncts. A zero adjunct is A60. These verbs include: written, authored, edited, published, dated, etc.

2) If a Ven which is also a BVC occurs in the C131 (Ven as an active object) then B43 is zero.

B2 Restrictions for the (A61,B41) Option

1) If B2 occurs as an object (BO) then either

   a) the BO is an element of C1 which is an element of C69 or C70, or

b) the BO is an element of C131 or C136 (the untensed verb, V, with its adjuncts) which is an element of C1 or C2; or

c) the BO is an element of C2 whose verb is a form of have and whose subject is a pronoun.

3.7.3 Adjuncts

The B2 option mentioned above is (A61, B41) and not just (A61) because the former yields the advantage of early attachment of the right adjuncts of the omitted word directly to the omission mark. Other situations exist in which an adjunct string (usually B41 - right adjunct of nouns and pronouns) is placed as an element of a parent string so as to cause a more rapid parse. One common example applies to the index term sequence, B5.

Consider the elementary index term sequence: B5

```
B5     DEFIN  ((B5B,B41)),T
B5B    DEFIN  ((A100),(B5B,A100))
```

As can be seen, B5B is a recursive string in that if the first option is tried and fails, the second option will cause another B5B node to be attached below the original B5B node. Again the first option will be tried and again it will fail and the process would continue indefinitely except for the fact that the parser recognizes recursive strings and requires that a recursive string S starting with word count W be successful (n-1) times before S^n (the string S for the n-th recursion) is allowed to be attached with a word count of W.

Now assume a message as follows:

'Give me anything on radar written by E. J. Smith.'

Strictly speaking, both of the informational clauses: on radar and written by E. J. Smith are right adjuncts of the pronoun anything, i.e. they are both right adjuncts or B41 strings. The B41 string is repetitive which means that its last option is (B41,B41). i.e. it consists of two elements each of which is the string itself. The purpose of such an option is to allow for two or more successive occurrences of the string. Again an infinite loop could be set up if a repetitive string S is attached to the tree and all of its options (except the last) fail. To prevent such a situation the parser requires that in order to try the last option of a repetitive string with word count W, one of the previous (n-1) options must have been successful in analyzing at least one word of the sentence starting at the W-th word.

Consider the events of parsing if B5 did not have the B41 as an element of its option. The prepositional phrase would be attached to the tree as right adjunct of the pronoun anything. Since written by E. J. Smith would then fail to fit into any of the remaining strings to be placed on the tree the parser would backtrack to the B41 string attempting to use its last option. Because it was already successful starting from the word on, the parser would allow the last option to be tried. Thus, 'on radar' would again be attached to the tree under the first B41 of the option (B41,B41) and 'written by E. J. Smith' would be attached under the second B41. Therefore the parser was required to construct the

substructure for 'on radar' twice. If this substructure had been longer and more complicated, this repetition in construction would add unnecessary delay to the parsing mechanism. Having the B41 in the B5 option permits the parser to immediately attach the second informational clause (i.e. written by E. J. Smith) to the tree. In the interpretation phase of the system, this is taken into account in analyzing the B41 element of a B5 string.

3.7.4 Conjunctions

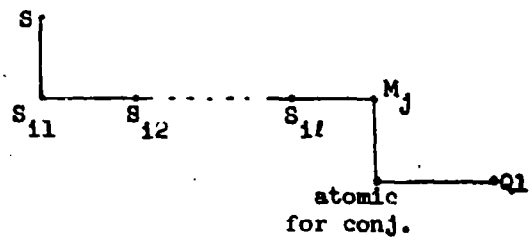
The grammar does not contain coordinate conjunctional strings in the main body of strings. If conjunctions (and other "special" words of the language) were accounted for explicitly in the strings, the grammar would attain very large proportions. To avoid this, a "special process" mechanism exists in the parser which allows for the insertion of an element in an already defined string of the grammar upon the appearance of a conjunction (or other "special" word) in the sentence.

Suppose word W has just been analyzed and word W+1 has become the current word. If W+1 has a special process mark M on its category list, the sublist of M is obtained. It is a list T, defined to be R_1T_1 or R_2T_2 or ... or R_nT_n. R_k is a series of tests and T_k is an option of T. If NS_{ij} (node representing the string S_{ij}) is the current node, a node NT_{kl} is attached (if R_k permits) to the right of it. Thus when (and if) NS is complete, the nodes $NS_{il},\ldots,NS_{ij},NT_{kl},\ldots,NS_{im}$ appear below it, as if S_i were defined to be S_{il} and ... and S_{ij} and T_{kl} and ... and S_{im}. Each element T_{kl} is the grammar string which contains the definition of the

special process word. Thus if W is 'and' its T list has one option T_1 and T_1 has one element T_{11} which is the grammar string representing conjunctions. However, words marked special may appear in a sentence in one of their non-special uses. Therefore a word with a special mark M is first treated specially, and if no analyses can be produced the special process marker M must be ignored at this point - NS_{ij} - in the analysis and the regular procedure followed until the analysis of NS is complete.

In the case of simple coordinate conjunctions the special process node is M1 for 'and', M2 for 'or', M3 for 'but', M4 for 'nor', and M5 for 'as well as'.

The last element of the one option for each of the above Mj (j = 1 to 5) strings is the conjunctional string, Q1, which produces its own set of options. Given that Mj has been inserted following NS_{ij} of string S, the tree would look like:



Q1 produces the following set of options:

(S_{i1})

(S_{i1-1},S_{i1})

$(S_{i1-2},S_{i1-1},S_{i1})$

.
.
.

$(S_{i1},S_{i2},S_{i3},\ldots,S_{i1})$.

That is, it carries out the process of structural repetition on the elements that were current when the conjunction appeared, namely, S_{i1}, S_{i2}, ..., S_{il}.

According to the procedure outlined above, a M-node will be attached at each succeedingly higher level of the tree until it is accepted as such or until the top of the tree is reached in which case backtracking will take place. At each such level the Q1 string will generate a set of options representing structural repetition at that level. From the experiments performed concerning the written syntactical structures most likely to occur, it has been found that it is necessary to permit such M-node attachment to the right of only a limited number of nodes. These nodes are listed below:

1) B5 (index term sequence) to permit a logical combination of index terms.
2) B24 (quantifier strings) to permit a logical combination of quantifier sequences as used in the Combine command. For example:

   'Give me the papers indexed by at least two but not more than four of the following terms: radar, sonar, laser, maser, pacer.

3) B0 (object strings) to permit multiple requests. A multiple request is a message indicating that the system is to respond to more than one request. For example: 'I want the author of document 110 and the title of anything on radar.' In such a case, the parse is

accomplished and the system proceeds to analyze only the first request contained in the message.

4) B32 (Ven with adjuncts) to permit a sequence of bibliographic verbs. For example:

'What has been written, edited, or published by Smith?'

5) C91 (noun phrases) to permit a sequence of related nouns to be joined. For example:

'What papers or books have been written on game theory?'

6) B41 (right adjunct of noun or pronoun) to permit conjoining information clauses of various kinds. For example:

'How about anything on radar and written by Smith.'

A comma is treated as a simple coordinate conjunction unless it is followed by a different coordinate conjunction. That is, in the sequence: 'Give me any stuff concerning radar, sonar, and laser.', the first comma is treated as a conjunction whereas the second comma is treated as pure punctuation. If this were not the case, the following might occur. Considering Figure 13, note that the second Q1 (i.e. $Q1_2$) is about to generate the option (A51) which cannot fit the sequence. The point is that the combination ', and ' is one conjunction and is treated as stated above. As a result the tree would be as shown in Figure 14. The special process string for the conjunction comma, M10, is similarly limited as to its left neighbors on the tree. The M10 to the

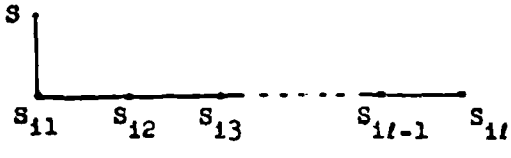
right of A100 is allowed since, in this usage, the comma is not a conjunction since there is no Q1 string as an element of M10.

In the case of correlative (or scope-marked) conjunctions, the special process node is M11 for 'both...and...', M12 for 'either...or...', M13 for 'neither...nor...', and M14 for 'not only...but also...'.

In terms of the operation of structural repetition by which conjunctional strings are obtained, the scope-marker (C') words either, neither, both, and not only can be seen to mark the point in the host string beyond which elements cannot be 'repeated' in the conjunctional string. That is, a scope-marker-and-conjunction pair C'...C marks off a structure X (string or string-segment) which also appears following C in the sentence: C'XCX. Since X is the expected structure when C' occurs, it is possible to define a special process, initiated by C', which inserts the string C'XC at the interrupt point where X is the specified string or string-element; when this string is satisfied the program reverts to its normal operation and finds X as it expected before the interruption.

The element in the one option for any scope-marked special process string is the scope-marked conjunctional string Q2 which produces all of the possible options for X from the remaining string elements to be attached to the tree at the present level.

Consider the string $S = (S_{i1}, S_{i2}, \ldots, S_{in})$ with the following tree:



$(l < n)$

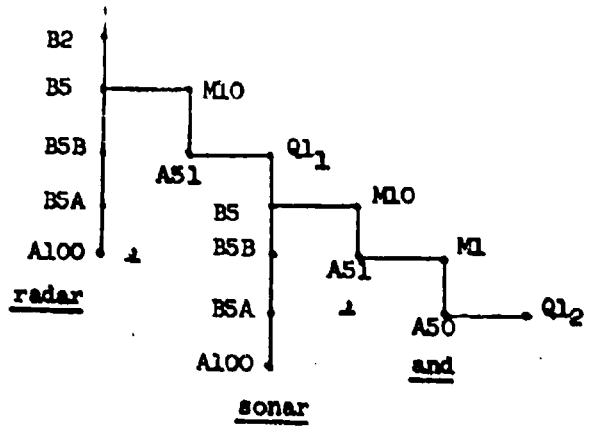


Figure 13

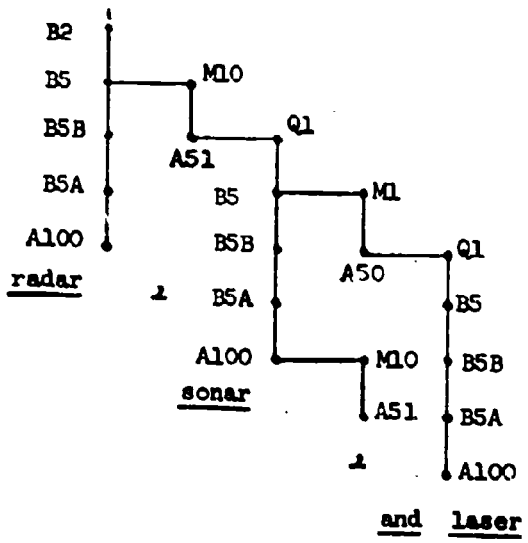
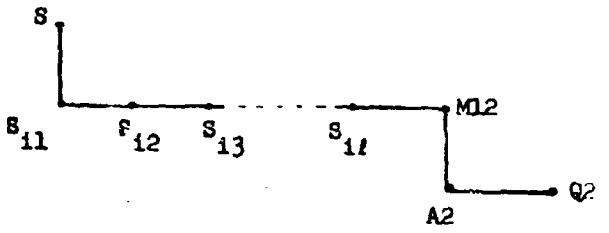


Figure 14

Upon occurrence of a scope-marked conjunction, say M12, the tree is expanded to:



The options produced by Q2 are:

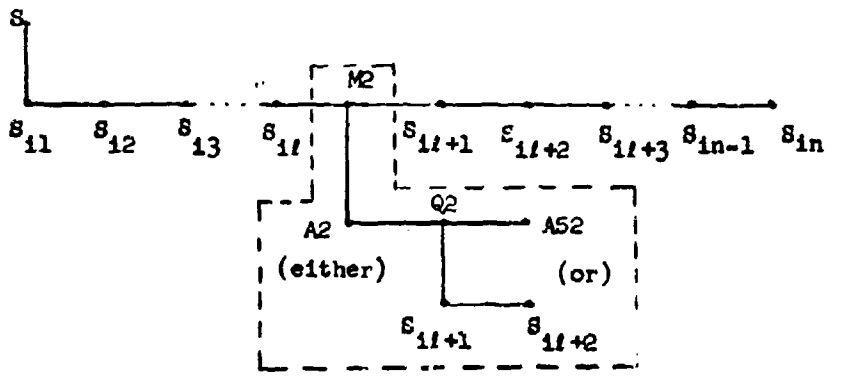
(S_{il+1})

(S_{il+1},S_{il+2})

.
.
.

$(S_{il+1},S_{il+2},\ldots,S_{in-1},S_{in})$.

Assuming that the second option, (S_{il+1},S_{il+2}), were actually correct, the tree would appear as:



The dotted enclosure is the inserted structure C'XC.

Based once again on the results of the language study experiments discussed before, it was discovered that the scope-marked conjunctions also have a restricted set of left attaching nodes.

These correlative conjunctions find their most wide applicability in structures involving the logical construction of index terms a . . as such may be attached to and, , , or, but and the prepositions that usually signal the oncoming index term sequence.

CHAPTER 4

PRAGMATIC INTERPRETER

4.1 Introduction

After the construction of the tree, the system may be thought of as existing in one of four states. The pragmatic interpreter is concerned directly with states 1 and 2 and will be discussed in the present chapter*. The third and fourth states which are, respectively, the specification filler and the organizer will be discussed in Chapter 5.

A description of the system states follows:

State 1: System is engaged in the Ultimate Object Analysis which determines the starting node to be used in the interpretation of the message.

State 2: Starting from the ultimate object or other node selected by the Ultimate Object Analysis, the command-set (not necessarily with the syntax required for each command) is determined.

State 3: This state is involved if the command-set includes either the NUMBER or COMBINE command. The syntax for these commands is formed during this state. The process, to be discussed in Chapter 5, depends on the command.

With the NUMBER command, the system will:

a) associate the proper sector (i.e. author, title, etc.) with each index term, and

* The term "pragmatic interpreter" implies that the system is interested in the 'intended' meaning of the user's message.

2) C5 String - Question with wh- with noun-omission.

The question (or wh-) word may be what, who.

Examples: Who wrote documents 1296, 301 and 2?

What is generic to automobile?

What do you have on sonar, and either laser or maser?

3) C6 String - Question with wh- with or without noun-omission.

The wh- word may be when, how, where.

Examples: How is radar defined?

Where was document 16 published?

When was accession number 412 written?

4) C9 String - Question with wh + noun with noun-omission.

The wh- word may be how many, what, which.

Examples: What words do you have starting with st?

How many papers on radar are there in the file?

Which words are synonymic to procedure?

The philosophy behind the analysis of question strings is to 1) transform the given message into an equivalent declarative, 2) compare the two parses to determine the first node of a common information clause, and 3) if other informational clauses seem to be ignored by an analysis of the equivalent declarative starting from this point, analyze these clauses "according to the clues contained therein".

Such clues mentioned above are stored in the word records and are brought out by several system packages each of which is

called upon in various situations. Some system packages with examples of their use in the general philosophy follow.

1. The Set Command Package

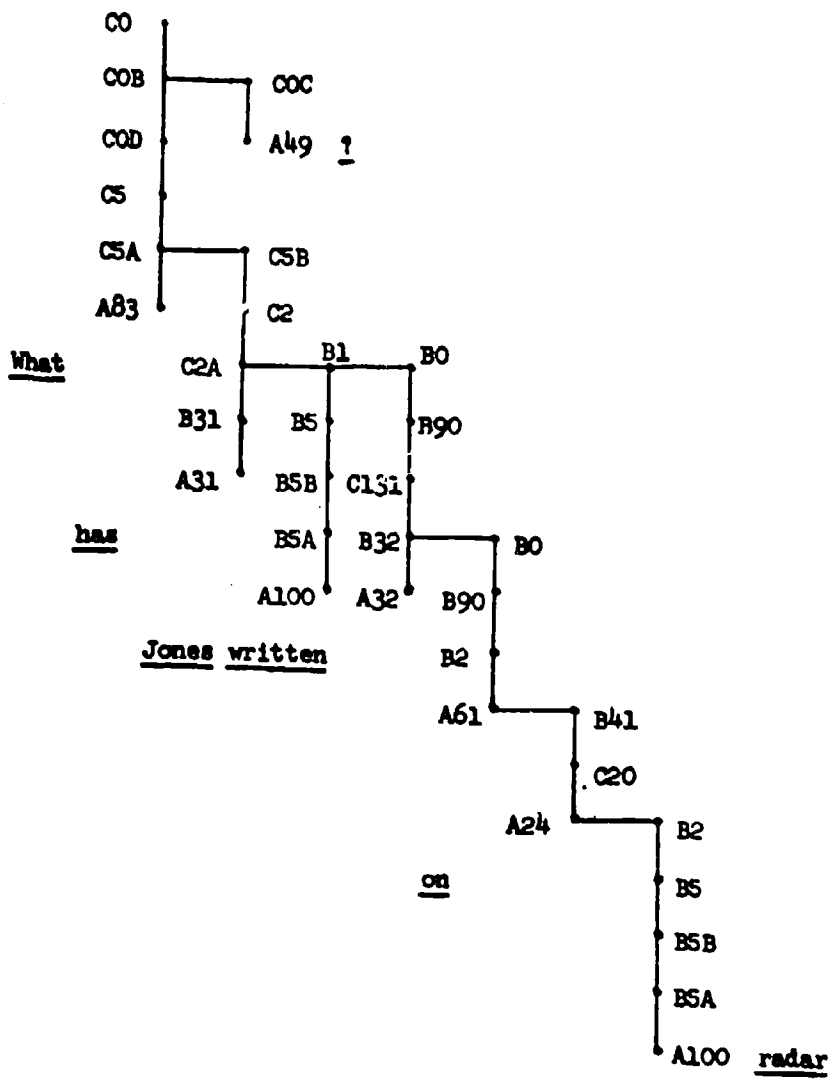
A Ven (past particle form of verb) or A30 (untensed verb) in a C131 (Ven + Ω active) or C136 (V + Ω) string respectively may be associated with a group of index terms. The word records of this verb contain information indicating the command to be set. (It is to be noted that only the NUMBER and COMBINE commands require elaborate syntax formation. The other commands need at most a listing of the index terms.) In the event that the NUMBER or COMBINE is set, the word record also supplies the data necessary to decide the index term's sector designation (i.e. author, title, abstract, etc.). A code representing this sector together with the index terms are put into the 'ARGUMENT' buffer as a partial syntax formation of part of the user's message.

This package finds its application in the ultimate analysis of various question strings. As an example, consider

'What has Jones written on radar?'

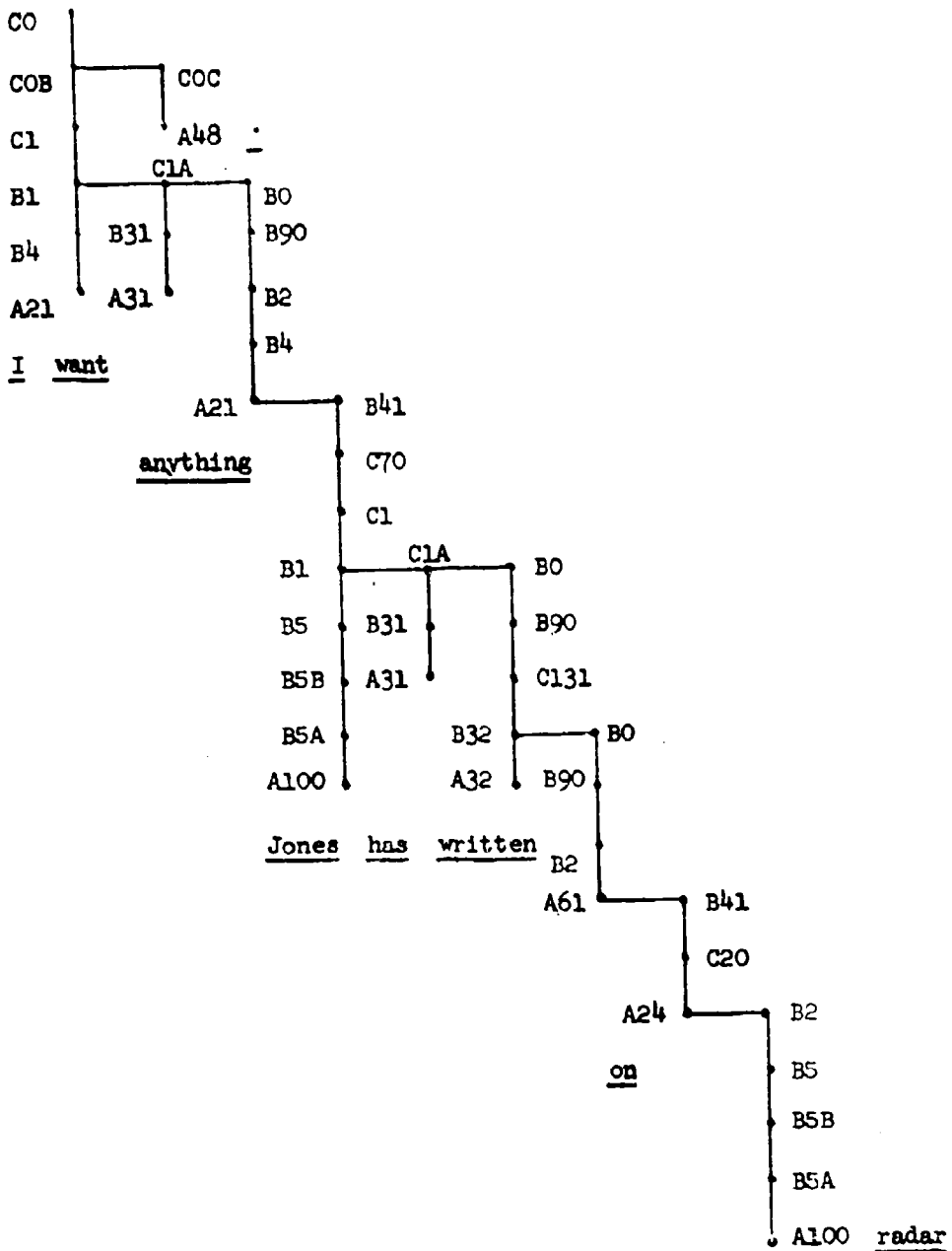
The parse of this C5 string is shown in Figure 15. Considering the given sentence as it might appear in an equivalent declarative sentence, the parse of 'I want anything Jones has written on radar.' shown in Figure 16 will suffice.

From Figure 16 it may be seen that the two informational clauses 'Jones has written', and 'on radar' are right adjuncts of anything (considered to be analogous to the omission mark of Figure 15 since the word anything implies no bias to any



Parse of: 'What has Jones written on radar?'

Figure 15



Parse of: 'I want anything Jones has written on radar.'

Figure 16

particular file as would papers or documents). Comparing Figures 15 and 16, it is seen that their first node of common structure is the B41 node of the (A61,B41) option of B2 which becomes the ultimate object. In order to proceed from this point as a declarative, however, the other informational clause must first be analyzed. To do so, the Set Command Package interrogates the word record of written for a SC code on the sublist of the category used in the parse, i.e. the sublist of A32.

	WORD	WRITTEN
	LISTIS	(.1,A32)
.1	LISTIS	(.2,BO,.3,BO,.4,SC,BVC)
.2	DEFOBJ	((B2))
.3	DEFOBJ	((A60))
.4	LISTIS	(.41,Z1)
.41	LISTIS	(Z2)
	END	WRITTEN

Figure 17

The sublist of SC contains the bit (Z1 is the first bit, Z2 is the second, etc.) to be set in the flag word representing the command to be used in the execution of the request. According to Figure 17, written causes the NUMBER command to be set since, as shown below, the first bit represents the NUMBER command. The analysis of the informational clause containing written is treated as occurring in the declarative of Figure 16 (i.e. Jones has written). The SEM Sublist Value Package discussed

Bit Position	Command	Bit Position	Command
1	NUMBER	7	THES/BF
2	COMBINE	8	THES/AF
3	SYNONYM	9	THES/X
4	DEFINE	10	THES/BT
5	RELATION	11	THES/AR
		12	FORM

below completes the analysis of this clause. Because the command-set has been determined, the ultimate object analyzer will ready the system for State 3 execution by pointing to the B41 node mentioned previously. State 3 will continue the analysis by considering the analogous declarative sentence.

Notice that for this sentence, the ultimate object analyzer performed the following:

1) The command-set was determined by the SET COMMAND PACKAGE.
2) One informational clause was partially analyzed.
3) The node representing the remaining informational clause was found (viz: the B41 of the (A61,B41) option of B2).
4) Readied the system for State 3 execution.

Notice that sentences like those below are similarly analyzed.

Has Carter or Wilson written on game theory?

Did Greene edit a book on network analysis?

2. The Index Term Lister Package

The syntax of many of the system commands requires a list of index terms (an index term is composed of one or more index items) separated by commas. When the command-set is one requiring such a format, the system will execute the Index Term Lister Package. The substructure of the B5 node contains an atomic A100 for each index item of the index term represented by the parent B5. This package gathers all the index items from the appropriate substructure of the tree and places their EBCDIC representation into the 'ARGUMENT' buffer separating the index terms by commas.

Consider:

'What is reentrant code and time sharing?

Recognizing that the message is a C5 whose main verb is a form of BE, subject is an index term sequence and object is empty, causes the DEFINE command to be set. The Index Term Lister Package is executed starting at the subject node B1. This will cause the index items reentrant, code, time and sharing to be placed in the 'ARGUMENT' buffer with a comma separating reentrant code from time sharing.

3. The SEM Sublist Value Package

In a previous example, viz: 'What has Jones written on radar?', it was stated that together with the sector codes gotten from the word record of written the index term Jones was put into

the 'ARGUMENT' buffer. This is not entirely true. Consider the message: 'What have Jones and Wilson written on radar?'. This message differs from the previous request in that the subject string is satisfied by a logical construction of index terms. A call for the Index Term Lister Package would cause Jones and Wilson, separated by a comma, to be placed in the 'ARGUMENT' buffer. Because the NUMBER command requires &, + or ↑ between index terms of the same sector designator and not commas, problems would arise. Therefore to keep the logical structure of the message, the SEM Sublist Value Package is executed.

The SEM Sublist Value Package will be briefly introduced below and more fully developed in Chapter 5. Every significant word occurring in an informational clause of a NUMBER command has a SEM code on the sublist of the category used for that word occurrence. Most such words have only one value on its SEM sublist. This value may or may not have its own sublist. In such cases that a word has more than one value, the context of its usage as indicated by the parse dictates which value is used. Therefore the function of this package is to place the SEM value of every word in the sentence in the indicated substructure under consideration into the 'ARGUMENT' buffer. Index terms have their EBCDIC representation together with their SEM value placed in the buffer. The purpose of having such a package is to make for uniform analysis of the various syntactical structures that could constitute an informational clause. This will be brought out more fully in Chapter 5.

Returning to the example: 'What have Jones and Wilson written on radar?' after executing this package, the 'ARGUMENT' buffer will contain the following:

200	Jones	52	200	Wilson	101

Index term Value of SEM	SEM Value for and	SEM Value of written holding key to sector designation

4.2.4 Conversationally-Dependent Sentences

Conversationally-dependent sentences are of two types: 1) they are responses to a system reply to a previous request and as such are abbreviated, or 2) they are requests in which the user has used that part of a declarative corresponding to the ultimate object. Examples of each type are:

Type 1: How about Jones.
And Smith.
Allen.

Type 2: Documents by Greene.
Synonymic to instruction.
Words generic to motor vehicle.

The type 2 conversationally-dependent sentence is treated as if it were preceded by 'I want ...' which is equivalent to saying that the sentence is the ultimate object. Type 1 requests are entirely different as they are based upon previous dialogue.

Consider the following two cases:

	CASE I	CASE II
user:	I want the definition of radar.	I want everything about radar.
system:	I don't know.	I don't have anything.
user:	How about sonar?	How about sonar?

The second user response in both cases is identical, yet their pragmatic interpretation must be different since in Case I a definition is requested whereas in Case II a document search is requested on the index term 'sonar'. In Case II, the system uses the first sector designator of the original request as the designator of 'sonar'. Therefore after each request, a record is kept of:

1) the command-set
2) the sector code of the first index term used in a NUMBER or COMBINE if such a command were the previous command.

To summarize, then, in the interpretation of type 1 conversationally-dependent sentences the system will use the previous command together with the newly supplied index terms.

4.2.5 Examples

Below are listed user messages along with the corresponding transformed message used in the analysis. Also shown, when applicable, is the 'ARGUMENT' buffer, STATE of system to be entered and ultimate object.

1) I want some papers written by Carter.

   a. Ultimate Object is: 'some papers ...'

   b. To enter State 2.

   c. Transformed message is same as original.

2) Give me the papers written by Carter.

   a. Ultimate Object is: 'me the papers ...'

   b. Enter State 2.

   c. Transformed message is: I want the papers written by Carter.

3) What has Carter written?

   a. Enter State 3.

   b. 'Argument': 200 Carter 101

4) Have you anything written by Carter?

   a. Ultimate Object is: 'anything ...'

   b. Enter State 2.

   c. Transformed message is: I want anything written by Carter.

5) What books do you have which were written by Carter?

   a. Ultimate Object is: 'books which were written by Carter.'

   b. Enter State 2.

   c. Transformed message is (in two steps):

      1. What do you have which was written by Carter?*

      2. I want books which were written by Carter.

---

* A note is made of books.

6) Do you have any papers that Carter wrote?

   a. Ultimate Object is: 'any papers that Carter wrote'.

   b. Enter State 2.

   c. Transformed message is: 'I want any papers that Carter wrote'.

7) How are radar and sonar defined?

   a. Enter State 4.

   b. 'Argument': radar, sonar.

4.3 Command-Set Generator

The Command-Set Generator, or State 2, is called upon if the Ultimate Object Analysis failed to determine the complete command-set necessary for the proper execution of the user's request. State 2 starts its analysis at the ultimate object node of the transformed message. In all but exceptional cases the ultimate object is a noun (or pronoun) phrase (the noun of which is called the core) whose adjuncts are informational clauses. The pragmatic content of these nouns is coded and placed with its word dictionary record. Words like information, data, material, stuff offer no clue as to the desired mode of operation, whereas words like papers, words, definition, author carry definitive pragmatic information. In fact the nouns appearing as the core of an ultimate object may be classified into one of three groups: 1) no specific mode: anything, something, 2) non-search mode: words, definition, phrases, and 3) search mode: documents, papers, books. The right adjuncts of the core noun are then analyzed, one

by one in order of appearance in the sentence, until the command-set is established. Each individual right adjunct carries with it its own decoding scheme based upon the atomics in its sub-structure and the subject-verb-object or verb-object relationship of the adjunct. Examples follow.

4.3.1 The Ven Phrase and Pure Prepositional Phrase

The mechanism involved in the decoding of prepositional phrases is embedded into that of the Ven phrase (i.e. past participle + Ω passive). The past participles (Ven) encountered in such environments are classified as search mode oriented (BVC in word record) or relation mode oriented (RVC). Within the search mode, this Ven may signal the execution of either the NUMBER, COMBINE, or FORM command. The ambiguity is resolved by the prepositional phrase associated with the Ven as either a right adjunct of this verb, B43, or as the passive object of this verb, B99. Notice that at this point, the prepositional phrase may be separately analyzed as such as long as the presence of the associated Ven is taken into account. The processing of the prepositional phrase takes into account 1) the preposition itself, 2) the associated verb, if any, which may be Ven, Ving, tV, 3) the object of the preposition - it may be an index term which may or may not indicate a date, or it may be another noun phrase, 4) the presence anywhere in the sentence of a phrase which would specifically direct the system to a particular mode of operation, such as 'in the thesaurus' in 'Give me everything in the thesaurus about radar.', and 5) the group into which the core of the ultimate

object is classified.

As an example consider the word record, shown in Figure 18, of about.

	WORD	ABOUT
	LISTIS	(.15,A24)
.15	LISTIS	(.1,TYP1,.2,TYP2,.3,TYP3)
.1	LISTIS	(.11,VOID,.11,BVC,.14,TVC)
.11	LISTIS	((A5),TD1,(A1),TD3,(A1),TD4)
.14	LISTIS	((A11),TD1,(A11),TD3,(A11),TD4)
.2	LISTIS	(.21,VOID,.21,BVC,.14,TVC)
.21	LISTIS	((A5),TD1,(A5),TD3,(A5),TD4)
.3	LISTIS	(.31,VOID,.31,BVC)
.31	LISTIS	((A1),TD3,(A1),TD4)
	END	ABOUT

Figure 18

The group into which the core of the ultimate object is classified is recorded as follows:

The OBTYP (object type) variable has the values:

TYP1 - no specific mode

TYP2 - non-search mode

TYP3 - search mode

The associated verb is indicated by the value of the PREVB (previous verb) variable as follows:

VOID - no associated verb

BVC - verb associated with bibliographic data
(e.g. written)

RVC - verb associated with relational data
(e.g. related)

TVC - verb associated with thesaurus or lexicon data
(e.g. beginning)

The presence of particular phrases directing the system to a particular mode of operation is indicated by the variable TDICT (thesaurus-dictionary) as follows:

TD1 - indicates THESAURUS mode

TD2 - indicates DEFINE mode

TD3 - indicates no specific mode and the index term is not a date

TD4 - indicates no specific mode and the index term is a date

Now, consider the following sentences.

1) I want anything about radar.

2) I want any papers about radar.

3) I want anything about radar in the thesaurus.

In the analysis of these three sentences the ultimate object would be respectively: 'anything about radar', 'any papers about radar', and 'anything about radar in the thesaurus'. The respective values of 1) OBTYP are TYP1, TYP3, and TYP1, 2) PREVB are VOID, VOID and VOID, and 3) TDICT are TD3, TD3, TD1. In each case, the Command-Set Generator will analyze the prepositional phrase 'about radar'. All the necessary information for this

analysis is stored in the word dictionary record of the preposition in tree-like fashion.

The process is as follows:

1) Start at the sublist of the preposition (A24).
   In this case .15 (refer to Figure 18).
2) Go to the sublist of the symbol that is the value of the variable, OBTYP.
3) Go to the sublist of the symbol that is the value of the variable, PREVB.
4) Go to the sublist of the symbol that is the value of the variable, TDICT.
5) This sublist contains a symbol Ax, where x is a number from 1 to 11. The value of x corresponds to the x^{th} bit position of the CMAND1 variable, which is one of two variables (the other is CMAND2) used to specify the command-set. Each bit corresponds to a different command, as shown below:

CMAND1	
Bit Number	Command
1	NUMBER
2	COMBINE
3	SYNONYM
4	DEFINE
5	RELATION
6	NOT USED
7	THES/BF

8	THES/AF
9	THES/x
10	THES/BT
11	THES/AR
12	FORM

The bit positions of CMAND2 correspond to the commands: AUTHOR, DATE, TITLE, EDITOR, PUBLISHER, JOURNAL, SPECIFIC, GENERIC, AUTHORITY LIST ENTRY, ABSTRACT, DESCRIPTORS, DESC/ALL, DESC/BIBLIO.

Using this scheme, sentences 1 and 2 will set the NUMBER command, sentence 3 will set the RELATION command. Although sentences 2 and 3 are interpreted correctly, it may be argued that sentence 1 could be requesting information concerning 'radar' from any of the mode files. In this sense, sentence 1 is ambiguous. Experience will help decide the eventual course to take in such cases. The choice selected here is based on the experiences of the author. It may be that 1) a dialogue between user and machine should be initiated to resolve the ambiguity, or 2) a record of past performance of the user might solve the ambiguity, or 3) the choice selected above is used in the vast majority of cases so as not to warrant the time-consuming (and in some cases, annoying) dialogue mentioned above.

If the object of the preposition is itself a noun phrase, then the possibility exists of setting the COMBINE command. Sentences translatable into the COMBINE command have a quantifier

sequence (B24) modifying a zero noun* (A62), as in:

Give me the papers indexed by not more than 3
of the terms: A,B,C,D,G.

The sequence 'not more than 3' modifies a zero noun whose right adjunct (B41) is the prepositional phrase 'of the terms'.

4.3.2 Adjectival Phrases

Certain adjectives indicate the desired mode of operation. In such cases, the adjective's word record carries the information in the sublist of COM symbol which occurs on the sublist of the category, adjective (A15). Consider the record of 'synonymous' below.

	WORD	SYNONYMOUS
	LISTIS	(.1,A15)
.1	LISTIS	(.2,COM)
.2	LISTIS	(Y)

The Y indicates that the synonym command is to be set. The adjective phrase holding the adjective under consideration will also contain the words involved. In such a case, the Index Term Lister Package will then place the index terms into the 'ARGUMENT' buffer.

---

* A zero noun indicates that a noun that does not necessarily have to occur at a certain point in the sentence, did not occur as in 'Those two were not there.' The noun which 'those two' modifies is said to be zeroed.

4.3.3 The Relative Clause - That + C1 with Noun Omission (C69)

The diversity in the subject-verb-object relationship applicable to the C69 string makes this string capable of appearing in requests involving all the various modes of operation. The analysis may be divided into three sections depending upon the subject of the C1 string.

a) Subject is 'you' or 'there'

Examples include:

Give me all that you have on radar.

I want anything that there is concerning the field of optics.

In such cases, the object is the omitted string (A61). Its right adjuncts may then be treated as if they had occurred alone.

b) Subject contains an index term

Examples include:

I want anything that Jones is the author of.

I want anything that radar is generic to.

Do you have anything that Jones has written dealing with radar?

The object of the verb in this adjunct contains the key to its interpretation. The noun author (which has the subcategory BN, bibliographic noun, in its word record) indicates the NUMBER command. Generic in the second sentence makes the analysis similar to that explained in Section 4.3.2 except that the opposite relation is required here. That is, the sentence:

'Give me all the words generic to radar.'

requires the inverse relation of that necessary for the sentence:

'Give me all the words that radar is generic to?'

Referring to the third example, written with its BVC subcategory indicates the NUMBER command is involved.

c) Subject is omitted

Examples include:

Give me everything that is generic to radar.

Give me anything that has been written describing radar.

What do you have that has Jones as the author?

What words are there that begin with the letters ST?

In these cases, either the verb (as in the last sentence) or its object (as in the other sentences) carries the distinguishing information.

4.3.4 Summary

It is to be noted, that in the entire Command-Set Generator analysis those words indicative of the system commands and syntactical structures carry the clues to the interpretation. The analyzer uses the parse generated by the syntax analyzer to determine the environment in which these words are used. Based upon the environment found and the subcategories stored in the words' dictionary records, the command-set is formed. In the commands associated with the CMAND1 variable, all but NUMBER and COMBINE would require the execution of the Index Term Lister Package in order to fill the 'ARGUMENT' buffer with the appropriate index terms. Recognition of a NUMBER or COMBINE command would cause

the system to enter State 3 which will form the specification part of the command as will be explained in Chapter 5.

The above discussion dealt with the adjuncts of core nouns indicative of no specific mode of operation. However, there are nouns which do indicate the entire command-set or only part of it.

The N48 nouns (indicating DEFINE mode) and the N49 nouns (indicating SYNONYM mode) yield a quick analysis when they occur as the core noun as in:

What is the meaning of radar?

Give me some synonyms of radar.

I want the definition of the following words:

radar, sonar, and laser.

The bibliographic nouns (BN) cause a CMAND2 command to be set. The BN subcategory of the word (e.g. author, editor) carries its own sublist indicating the bit to be set in the CMAND2 variable. Once the command(s) corresponding to these BN nouns have been set, the analysis continues as above to find other commands that might be required.

Consider:

Give me the author of any papers dealing with radar.

The BN noun, author, causes the AUTHOR command to be set. Analysis would continue interpreting 'any papers dealing with radar' as if it were part of the sentence 'I want any papers dealing with radar'.

The various word categories used in the analysis are shown in Appendix D.

CHAPTER 5

SPECIFICATION FILLER AND ORGANIZER

5.1 Introduction

If the command-set generated by either the Ultimate Object Analysis or the Pragmatic Analysis includes the NUMBER or COMBINE command, then the Specification Filler must be executed to establish their specification part before the final output commands can be formed. This specification part which includes the association of a sector designator with each index term and the formation of the implied logical construction of the request by the proper placement of parenthesis for grouping and of the logical symbols &, +, †, is performed by the system in State 3 or the Specification Filler State. After the completion of State 3, the Organizer (State 4) forms the various commands together with their specification part in the output buffer in the proper sequence.

5.2 Specification Filler

Before execution of State 3, the previously active state has determined the proper starting node (command-formatter node) for the specification analysis. In some cases, part of the analysis has been made (as the example on page 77), and the results placed in the 'ARGUMENT' buffer.

The Specification Filler analysis includes:

1) formation of a sequence of codes representing the significant words of the informational clauses of the request starting from the command-formatter node.

2) the manipulation of this sequence in order to associate each index term with a code representing the appropriate sector designator. Also, as is required by the syntax rules of the commands, all index terms must sequentially follow its associated sector designation code. In addition, multi-word conjunction (e.g. and either) codes are replaced by one representing the collective action of the conjunction.

3) the logical construction implied by the original request is maintained. Any ambiguity inherent in the user's message is resolved on the basis of a hierarchy scheme for conjunctions. All codes representing parenthesis, logical symbols, and sector designators are replaced by their actual representation as required by the command's syntax.

5.2.1 SEM-Value Extractor

Code numbers for all the significant words that occur in the informational clauses of a NUMBER or COMBINE command are stored in that word's dictionary record in the sublist of the SEM category which is itself found on the sublist of the category chosen for the word. Some words (e.g. written) have more than one code number (or SEM-value) indicating that the proper value to be used depends upon the context in which this word is used. Also some words (e.g. the) have no SEM sublist at all indicating that their presence in the string (although necessary for syntactical purposes), reveals no information useful for command formation.

Words indicative of a sector designator that can occur both before and after its associated index terms have a multi-valued SEM sublist. The word 'written' which may occur before an index term as in 'written on radar' or after an index term as in 'that Jones has written' has a SEM value associated with each case. It is the purpose of the SEM-Value Extractor to resolve all ambiguities through the tree produced by the syntax analyzer.

At the conclusion of the SEM-Value Extractor, the 'ARGUMENT' buffer contains the SEM-value of all words occurring in the informational clauses of the request.

5.2.2 Associating Mechanism

The Associating Mechanism associates the various index terms, as represented in the 'ARGUMENT' buffer, with the proper sector designator code and does so ensuring that the index terms follow their sector code in the 'ARGUMENT' buffer.

Some examples follow to help bring out the methods used. Throughout these examples, the following SEM values were used.

WORD	SEM VALUE
WRITTEN	1
EDITED	2
[illegible] LISHED	3
SMITH	200 followed by Smith
JONES	200 followed by Jones
,	51
OR	53

BY	22
EITHER	55
1967	201 followed by 1967
THAT	82
PERIOD	99
IN	20

Example 1: Give me anything written, edited or published by either Smith or Jones.

The Ultimate Object Analysis establishes the ultimate object as being 'anything ...'. The Pragmatic Analysis causes the NUMBER command to be set by virtue of the respective values of OBTYP, PREVB, TDICT and the fact that the index terms do not represent a date, as explained in Section 4.3.1. As a result of the SEM-Value Extractor, the 'ARGUMENT' buffer contains the sequence:

1 51 2 53 3 22 55 200 Smith
53 200 Jones

Because 'written' occurs preceding its index terms (a.i.b. 1)* there must be an associated preposition. The next element in the sequence being a conjunction† instead of the preposition indicates that a sequence of BVC words is present. The system will now associate the preposition 'by' (a.i.b. 22) and its following index

---

* a.i.b. is an abbreviation for 'as indicated by the'.

† All codes 51-70 indicate a conjunction and 20-49 indicate a preposition.

term sequence - 55 200 Smith 53 200 Jones - with each of the BVC words written, edited (a.i.b. 2) and published (a.i.b. 3). The code representing the sector designator for 'written by' is gotten from the sublist of the SEM value of written. Referring to the word record of written (Figure 19), this sublist contains the code for the preposition involved, viz, 22*. This 22 has a 2 on its sublist. The 2 represents the sector designator (in this case, AUTHOR). If the 22 had no sublist, as is the case of 20 (in), the system must make a further study of the sequence to determine the sector designator.

```
          WORD      WRITTEN
          LISTIS    (.1,A32)
.1        LISTIS    (.4,SC,.2,PO,.3,BO,.15,SEM,BVC,(A1),BV)
.15       LISTIS    (.31,N1,(N2),N101)
.31       LISTIS    ((N2),N22,N20,N21,(N14),N24,(N14),N25,
          LISTIS    N26,(N12),N27,(N11),N28,N29,N30,
          LISTIS    (N11),N31,(N12),N32,(N15),N33)
            .
            .
            .
          END       WRITTEN
```

Figure 19

Therefore the resulting 'ARGUMENT' buffer is:

---

* N22 actually appears. The N is necessary for program considerations, but the actual list will have 22.

2 55 Smith 53 Jones 51 4 55 Smith 53
Jones 53 5 55 Smith 53 Jones

The 200 code indicating a non-date index term has been eliminated. The numbers (1-15) above indicate sector designators and are no longer SEM values. The above sequence is used to form the proper logical constructions by the placement of parenthesis and then the proper syntactical symbols replace all codes as will be discussed in Section 5.2.3.

Example 2: What has Jones written that was published in 1967?

As a result of the analysis carried out in States 1 and 2, the command-formatter node is 'that ...', and the 'ARGUMENT' buffer before execution of State 3 contains:

200 JONES 101

As a result of the SEM-Value Extractor, the 'ARGUMENT' buffer contains:

200 JONES 101 82 3 20 201 1967

Because written occurs following its index term (a.i.b. 101), the sector designation code must be placed into a position preceding the index term. The sublist of written's SEM value indicates the sector designation code. As seen from Figure 19, its value is 2. Therefore at the conclusion of the analysis of the first informational clause, the 'ARGUMENT' buffer contains:

2 Jones 82 3 20 201 1967

The 82 (representing that) is used to analyze informational clauses similar to: 'that has Jones as the author', i.e. in cases where the sector designator is represented by a noun instead of a verb as in this case. Therefore in this case the 82 is ignored.

The 3 (representing published) is treated similarly to that of written in Example 1, i.e. the code for the following preposition is looked up in the sublist of the 3 in the word record of published (see Figure 20).

```
        WORD      PUBLISHED
        LISTIS    (.1,A32)
.1      LISTIS    (.15,SEM,BVC,.5,SC,(A5),BV,.2,BO,.3,PO)
.15     LISTIS    (.31,N3,(N5),N103)
.31     LISTIS    (N20,N21,(N5),N22,(N14),N24,(N14),N25,
        LISTIS    N26,(N12),N27,(N11),N28,N29,N30,(N11),
        LISTIS    N31,(N12),N32,(N15),N33)
          .
          .
          .
        END       PUBLISHED
```

Figure 20

The sublist in question is the .31 list. The preposition to be found is 20 which can be seen to have no sublist. This indicates that further analysis is needed to determine the sector designator code. The system uses the remaining portion of the 'ARGUMENT' to distinguish between structures like:

1) published in 1967

2) published in the ACM

3) published in the period 1957 to 1961

4) published in 1957-63

5) published in the 1950's

The 'ARGUMENT' buffer in each of these cases would be, respectively:

1') 3 20 201 1967

2') 3 20 200 ACM

3') 3 20 99 201 1957 34 201 1961

4') 3 20 204 1957-63

5') 3 20 202 1950's

It can be seen that each case has its own distinguishing features which are used to determine the appropriate sector designation code. These five clauses represent respectively papers published in the single year 1967, papers appearing in the ACM publication, papers published in any year between 1957 and 1961, papers published in any year between 1957 and 1963 but expressed as an hyphenated date, and papers published in the decade starting at 1950. The sector designator codes applicable in these cases are respectively 9 (indicating exact date), 8 (indicating journal publication), 14 (indicating interval of dates given the two end points), 10 (indicating hyphenated dates), 13 (indicating a decade of dates).

Therefore returning to the example at hand, the 'ARGUMENT' buffer would be:

2 JONES 9 1967

Every informational clause has been reduced to its index term sequence preceded by the appropriate sector code designator. A complete list of the sector code designators follows:

CODE	SECTOR DESIGNATOR
2	AUTHOR
3	TITLE
4	EDITOR
5	PUBLISHER
6	DESCRIPTOR/ABSTRACT
8	JOURNAL OCCURRENCE
9	DATE - EXACT
10	DATE - HYPHENATED
11	DATE - MINIMUM
12	DATE - MAXIMUM
13	DATE - DECADE
14	DATE - INTERVAL

The complete list of SEM values appears in Appendix E.

5.2.3 Logical Maintenance

This step in the analysis performs the following functions:

1) lists all the required dates explicitly in the 'ARGUMENT' buffer in the cases in which the sector designator code is between 10 and 14 inclusive.

2) places parenthesis codes in the buffer to maintain the implied logical construction.

As an example, consider 'I want anything written after 1966 dealing with either nylon, rayon and dacron or wool, but not published by Stevens McGill.' Following the procedures outlined previously, the 'ARGUMENT' buffer as a result of the SEM-Value Mechanism would be:

1	28	201	1966	71	36	55	200	NYLON
51	200	RAYON		52	200	DACRON	53	200
WOOL	51	54	60	3	22	200	STEVENS	MCGILL

As a result of the Associating Mechanism, the 'ARGUMENT' buffer would be.

11	1966	6	55	NYLON	51	RAYON	52
DACRON	53	WOOL	65	5	STEVENS	MCGILL	

The 11 indicates that 1966 is the minimum year desired so that 1967, 1968, and 1969 will be put into the final command along with 1966 all joined by the logical or (+). The 6 indicates that the following index term sequence refers to descriptors, and the 5 indicates that Stevens McGill is a publisher.

All the logical and control symbols used at the stage in the analysis are represented by codes as follows:

(- 110	+ - 114
) - 111	& - 115
[- 112	¬ - 116
] - 113	/ - 117

9	1966	114	1967	114	1968	114	1969	6	55
NYLON	51	RAYON		52	DACRON	53	WOOL	65	5
STEVENS	MCGILL								

The placement of logical symbols must be made to maintain the implied logic both between informational clauses and within any given clause. The conjunctions are divided into two groups - those conjunctions preceded by a comma and those not preceded by a comma are respectively Group 1 and Group 2. Any Group 1 conjunction has a higher priority (i.e will be considered first in this analysis) than any Group 2 conjunction. Within any group the order of decreasing priority is:

as well as

both ... and ...

not only ... but also

but neither / and neither

and either

or either

either

and also

but not / and not

but

not

nor

or

and

If two connectives have the same priority they are operated upon as they appear in the sentence reading from left to right.

Under this scheme, the connective ', but not' is taken first. As with every connective, it must be determined whether the connective joins two informational clauses or is within a single informational clause. In this case, the former is true. Therefore, the 'ARGUMENT' buffer contains:

9 1966 114 1967 114 1968 114 1969 6 55
NYLON 51 RAYON 52 DACRON 53 WOOL 111 116
110 5 STEVENS MCGILL

The matching parentheses are placed adjacent to a previously placed parenthesis, if any exists, or else at the beginning and end of the sequence. Therefore, the buffer contains:

110 9 1966 114 1967 114 1968 114 1969 6 55
NYLON 51 RAYON 52 DACRON 53 WOOL 111 116
110 5 STEVENS MCGILL 111

The comma which is by itself is the next connective to be considered. Since its environment implies an <u>and</u> construction, this comma is treated as such resulting in:

110 9 1966 114 1967 114 1968 114 1969 6 55
NYLON 115 RAYON 52 DACRON 53 WOOL 111 116
110 5 STEVENS MCGILL 111

The next connective, 'either ... or', is within an informational clause and as such uses brackets for purposes of grouping resulting in:

110 9 1966 114 1967 114 1968 114 1969 6 112
NYLON 115 RAYON 52 DACRON 113 114 112 WOOL 113 111 116
110 5 STEVENS MCGILL 111

After the connectives are operated upon, the 'ARGUMENT' buffer contains the following [the codes representing the logical and control symbols are still present but their actual symbol is used below]:

(9 1966 + 1967 + 1968 + 1969 6 [[
NYLON & RAYON] & [DACRON]] + [WOOL]
) ↑ (5 STEVENS MCGILL)

It should be noticed that the informational clauses 'after 1966' and 'dealing with ...' are not joined by a connective in violation of the syntax rules. Therefore such a situation will be treated as an 'and' connective between informational clauses, resulting in:

(9 1966 + 1967 + 1968 + 1969) & (6 [[
NYLON & RAYON] & [DACRON]] + [WOOL]
) ↑ (5 STEVENS MCGILL)

The matching parenthesis of the just treated 'and' are placed adjacent to the closest left and right parenthesis from this 'and' as seen below. Note that if there were no distinction between parenthesis and brackets, confusion would result. The final 'ARGUMENT' buffer is:

((9 1966 + 1967 + 1968 + 1969) & (6 [[NYLON & RAYON] & [DACRON]] + [WOOL])) ↑ (5 STEVENS MCGILL)

At this point, all logical and control codes and sector designation codes are replaced by their actual representation yielding a final specification part:

((DATE 1966 + 1967 + 1968 + 1969) & (DESC ((NYLON & RAYON) & (DACRON)) + (WOOL))) ↑ (PUBL STEVENS MCGILL)

It should be noted that the above connective mechanism is limited as to the occurrence of a higher priority connective within the scope of a lower priority scope-marked connective. That is, in the above example if it had been: 'I want anything written after 1965 dealing with either nylon, rayon, and dacron or wool, but not published by Stevens McGill.', the comma following 'rayon' would cause the connective ', and' to be executed before 'either...or' thereby causing an incorrect grouping.

5.3 Organizer

The function of the Organizer is to form the output buffer with the selected commands and their associated specification parts. The commands are placed in the order necessary to perform the request. For example, if the request is:

'Give me the author of anything on optical scanning.'

then the output buffer would contain:

```
                                                          E
NUMBER   DESC   OPTICAL   SCANNING   **   AUTH   **      O
                                                          M
```

The NUMBER command will form a list of document numbers each of which has been indexed by 'optical scanning' as subject matter. The AUTH command will then give the user the author of each document in the list formed by the NUMBER command.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH SUGGESTIONS

6.1 General Conclusions

Real English has been designed for use in the information retrieval system of the Moore School Information Systems Laboratory. It is programmed on the RCA Spectra 70/46 entirely in the FORTRAN IV language. At present, it is a stand alone package and as such inputs its user messages through a card reader and outputs the translated Symbolic Command Language commands to the line printer. Described below are sample dialogues based upon the indicated messages to illustrate the various capabilities of the system. In the accompanying figures, the system responses are indicated by an asterisk (*) at the left margin†. The translated Symbolic Command Language commands are shown in each case and are indicated by a slash (/) at the left margin.

Incorporation of the Real English package into an on-line information retrieval system which has taken into account the linguistic style of its users and the complete set of system commands will enhance this system's natural language man-machine conversational capabilities. The user will be free to use messages whose pragmatic interpretation is dependent upon the previous dialogues. Furthermore, the system will be able to recognize messages that are not, strictly speaking, sentences.

---

† The system responses are actually taken from the present MSISL information retrieval system for which Real English was designed. They are included to produce typical dialogues based upon the supplied user messages.

In addition, multi-mode access affords the user a wider range of search strategy. Illustrations of these features are presented in Figures 21-23.

The search strategy portrayed in Figure 21 reveals that the user wishes to do a combinatorial search based upon more than two of his index terms. The Document List formed as a result of the execution of the translated COMBINE command is used to extract the titles requested in the first message and the authors requested in the second message. Because the initial request explicitly referred to the information sectors desired, no computer initiated and directed dialogue occurred. The last message which may be classified as conversationally dependent uses information derived from the previous message in order to associate Heilman with the sector designator AUTH or author.

The first message of Figure 22 illustrates the system's ability to associate the preposition by with each of the past participles and the index term Johns with each informational clause. Since no particular informational sectors were indicated in the request the system enters into a system initiated and directed dialogue to solicit this information. Note that before the user returns to his initial line of questioning he has made reference to both the DEFINE and RELATION files based upon information revealed to him in the SEARCH mode of operation. This multi-mode operation is a key feature of Real English.

The dialogue dependency feature of Real English is further illustrated in Figure 23. In this case, users A and B have

PLEASE GIVE ME THE TITLE OF DOCUMENTS ON MORE THAN TWO OF THE FOLLOWING AREAS HARMONIC ANALYSIS, NONLINEAR CONTROL, FEEDBACK, DYNAMIC COUPLING.

COMBINE (G2) DESC HARMONIC ANALYSIS / NONLINEAR CONTROL / FEEDBACK / DYNAMIC COUPLING **
TITL **

25 DOCUMENTS HAVE BEEN SELECTED.

19 DOCUMENTS INDEXED BY EXACTLY 2 OF THE INDEX TERMS
7 DOCUMENTS INDEXED BY EXACTLY 3 OF THE INDEX TERMS
0 DOCUMENTS INDEXED BY EXACTLY 4 OF THE INDEX TERMS

ACCESSION NO. 112
TITL: HARMONIC OSCILLATIONS OF NONLINEAR SYSTEMS

ACCESSION NO. 253
TITL: NONLINEAR FEEDBACK

ACCESSION NO. 1713
TITL: HARMONIC ANALYSIS OF STRINGS

WHO WROTE THEM?
AUTH **

Figure 21 - Dialogue 1

Figure 21 (Con't.):

```
*       ACCESSION NO. 112
*       AUTH:  M. KLOPTER, P. HEILMAN
*
*       ACCESSION NO. 253
*       AUTH:  J. ARNOLDSEN
*
*       ACCESSION NO. 1713
*       AUTH:  A. B. STEVENS

        WHAT ELSE DID KLOPTER WRITE ABOUT HARMONIC ANALYSIS?
/       NUMBER AUTH KLOPTER & DESC HARMONIC ANALYSIS **
*
*       NO REFERENCES HAVE BEEN FOUND.

        HOW ABOUT HEILMAN OR STEVENS ON FEEDBACK NETWORKS?

/       NUMBER AUTH ( HEILMAN + STEVENS ) & DESC FEEDBACK
/       NETWORKS **
*
*       NO REFERENCES HAVE BEEN FOUND.
```

```
        DO YOU HAVE ANYTHING ABOUT READING WRITTEN, EDITED
        OR PUBLISHED BY JOHNS?
/       NUMBER ( DESC READING ) & ( AUTH JOHNS + EDIT JOHNS
/       + ISSR JOHNS ) **
*
*       26 DOCUMENTS HAVE BEEN SELECTED
*       DO YOU WANT TO SEE ANY?

        YES
*       INDICATE DESIRED INFORMATION (AUTHOR, TITLE, ETC.)

        TITLE
*       ACCESSION NO. 111
*       TITL:  LANGUAGE EXPERIENCE IN THE TEACHING OF READING
*
*       ACCESSION NO. 473
*       TITL:  READING THROUGH PHONICS
*
*       ACCESSION NO. 506
*       TITL:  READING DIFFICULTIES DUE TO EYE DEFECTS

        WHAT IS THE LANGUAGE EXPERIENCE?
/       DEFINE LANGUAGE EXPERIENCE **
*       LANGUAGE EXPERIENCE -
*               AN APPROACH TO READING THAT IS BASED UPON
*               PUPIL-MADE MATERIAL **
```

Figure 22 - Dialogue 2

Figure 22 (Con't.):

```
        GIVE ME SOME TERMS THAT ARE SPECIFIC TO EYE DEFECTS.
/       RELATION (7) EYE DEFECTS
*
*       EYE DEFECTS -
*               SPECIFIC TERMS:  MYOPIA, HYPEROPIA, ASTIGMATISM,
*               CATARACTS

        WHAT DOES HYPEROPIA MEAN?
/       DEFINE HYPEROPIA **
*               A CONDITION IN WHICH VISUAL IMAGES COME TO A
*               FOCUS BEHIND THE RETINA OF THE EYE.

        WHAT DOCUMENTS DO YOU HAVE IN THE FIELD OF READING
        DISABILITIES WRITTEN IN EITHER 1967 OR 1968?
/       NUMBER ( DESC READING DISABILITIES ) & ( DATE 1967 +
/       1968 ) **
```

User A

WHAT DOCUMENTS DO YOU HAVE CONCERNING REFLECTION?

/ NUMBER DESC REFLECTION **

HOW ABOUT REFRACTION?

/ NUMBER DESC REFRACTION **

DIFFRACTION.

/ NUMBER DESC DIFFRACTION **

User B

GIVE ME THE DEFINITION OF REFLECTION.

/ DEFINE REFLECTION **

HOW ABOUT REFRACTION?

/ DEFINE REFRACTION **

DIFFRACTION

/ DEFINE DIFFRACTION **

Figure 23 - Dialogue 3

identical follow-up requests. However their pragmatic interpretations are dependent upon the previous dialogue and thus user A continues to receive information based upon the NUMBER command whereas user B receives information based upon the DEFINE command.

The strong linguistic basis inherent in Real English is due to its syntactical grammar. This grammar is powerful enough to accept a wide range of syntactical structures and yet flexible to be changed according to future developments. An example of the logical construction permitted by Real English is illustrated in Figure 24. Figure 25 shows several different messages which would be translated into the same command to extract the documents written by Jones and at the same time whose subject area is radar. Such diversity in the structure of user messages demonstrates the versatility of the Real English system.

For additional translated user messages, refer to Appendix F.

6.2 Future Research Goals

Future research associated with this dissertation should encompass the areas of grammar evaluation, influence of a computer-initiated and directed dialogue on system performance, and pragmatic ambiguity resolvers.

To have a truly useful information retrieval system with a natural language man-machine interface, the grammar comprising the acceptable syntactical structures must be shown to handle the linguistic style of its users. Using an actual information retrieval system, experiments should be conducted along these lines. Messages which cannot be properly parsed should be

I WANT THE AUTHOR, DATE AND TITLE OF ALL MATERIAL DEALING WITH THE AREA OF COSMIC RADIATION WRITTEN BY SCHWARTZ OR ALLEN BUT NOT ROBSEN AFTER 1966.

```
NUMBER  ( DESC COSMIC RADIATION ) & (( AUTH ( SCHWARTZ +
        ALLEN ) ↑ ( ROBSEN )) & ( DATE 1966 + 1967 + 1968
        + 1969 )) ** AUTH ** DATE ** TITL **
```

GIVE ME ANYTHING WRITTEN IN THE 1950's ON BOOLEAN ALGEBRA.

```
NUMBER ( DATE 1950 + 1951 + 1952 + 1953 + 1954 + 1955 +
       1956 + 1957 + 1958 + 1959 ) & ( DESC BOOLEAN ALGEBRA ) **
```

Figure 24 - Logical Complexity

WHAT DID JONES WRITE ABOUT RADAR?

WHAT HAS BEEN WRITTEN BY JONES ABOUT RADAR?

GIVE ME SOMETHING IN THE AREA OF RADAR BY JONES.

DO YOU HAVE ANYTHING ON RADAR AUTHORED BY JONES?

WHAT HAS JONES WRITTEN ABOUT RADAR?

PAPERS BY JONES ON RADAR.

AUTHORED BY JONES ABOUT RADAR.

I WOULD LIKE MATERIAL ON RADAR BY JONES.

LIST THE PAPERS BY JONES THAT DEAL WITH RADAR.

WHAT DO YOU HAVE ON RADAR WRITTEN BY JONES?

COULD I HAVE MATERIAL ABOUT RADAR THAT WAS WRITTEN BY JONES?

Figure 25

Diversity of Syntactic Structures

collected and decisions affecting their inclusion into the grammar should be made based upon their frequency of occurrence and relative importance to the total retrieval service.

The computer initiated and directed dialogue may have two major effects on system operation: 1) it may affect the linguistic style of the users, and 2) it may have psychological effects that may be detrimental to the mental attitude of the user. This dialogue may be useful in overall system performance by leading the user to his next request and thereby detouring him from a line of questioning which the pragmatic interpretor is not yet prepared to handle. For example, if the user has received the number of documents satisfying his initial SEARCH mode query and is unaware of the particular information available to him, his second request may be something like - 'What do I do now?' or 'What next?'. A dialogue initiated immediately after informing him of the number of documents might lead him to discovering system capabilities and also avoid the above response which the system may not be able to handle. On the other hand, constant interruptions by the system might annoy the user and so act in a detrimental manner toward system performance. Experiments should be performed to achieve the proper balance.

With the addition of more and more modes of operation, the possibility of pragmatic ambiguity increases. Consider a one mode system having the SEARCH mode. A query might be: What do you have related to *radar*? which would be translated into the number of documents whose subject matter is radar. Also consider a one mode system whose mode is the RELATION mode. The same

query would be translated into a series of words or phrases associated with radar through one or more relationships. If a system had both of these modes, an ambiguity would occur which might be resolved by a man-machine dialogue or by a profile chart of the user. Such a chart might include, for example, the user's propensity to use certain words or phrases when referring to particular system modes. In this way the system could use past experiences of the user to resolve ambiguities.

APPENDIX A

SENTENCES FROM THE ORAL AND WRITTEN EXPERIMENTS

APPENDIX A

SENTENCES FROM THE ORAL AND WRITTEN EXPERIMENTS

I would like to have a list of magazine articles on thin films.

I would like a list of references on the following subjects: computer memory elements, computer stores, thin films, computer memory design.

Give me anything else that J. F. Brown wrote.

Give me anything Williams has written.

List all the books which contain information about computer memories written by D. Simon.

Supply list of books on computer design.

Do you have anything on thin film reliability?

What are the other books written by Williams?

Did Jenkins write any other books?

Are there any other books by Simon?

What about thin film physics?

Can you find anything on thin film manufacturing techniques?

Give me a list of references by the following authors: D. Simon, W. Evans, S. T. Jenkins, L. Williams.

Supply list of books by D. Simon.

Let me see the references for computer memory design.

Is there anything under memory design logic?

I would like the titles of books written by R. Gray.

APPENDIX B

SEMANTIC EXPANSION ILLUSTRATION

APPENDIX

SAMPLE EXPANSION ELABORATION

The term 'interpretive program' is expanded below:

A. First-level response:

"An INTERPRETIVE PROGRAM is a computer program that combines translation and execution."

B. Second-level response:

"An INTERPRETIVE PROGRAM is a computer program which receives a sequence of commands in a source language, examines each command, determines a translation to replace it in the object language, and executes it if possible. The major characteristic of an INTERPRETIVE PROGRAM is that the translation of an instruction is performed each time the instruction is to be obeyed."

C. Third-level response

"An INTERPRETIVE PROGRAM carries out the instructions of a program written in one language by translating each instruction of that source language into a sequence of computer instructions in the language of the computer being used, and by allowing these computer instructions to be obeyed before translating the next instruction. This step-by-step translation and execution becomes significant when the execution of one instruction causes a change in the meaning of that instruction or another one. A new translation of the changed instruction will

oc necessary then before it can be correctly executed."

D. Fourth-level response:

"Consider the following sequence of instructions:

Position	Instruction
1	FETCH 5
2	ADD 6
3	STORE 5
4	GO TO 1
5	...
6	...

An INTERPRETIVE PROGRAM might first translate FETCH 5 into 24005 ("bring into the accumulator the contents of memory position 5") and execute the instruction. Then it might translate STORE 5 into 02005 ("store the contents of the accumulator in memory position 5") and execute that instruction. Finally, it might translate GO TO 1 into 32001 ("go to the instruction located in memory position 1 and execute it"). The instruction located at memory position 1 is FETCH 5. Because the INTERPRETIVE PROGRAM has carried out all instructions immediately after translating them, memory position 5 now contains a new value which will be incorporated into all further instructions involving it. If translation of all instructions had been completed before any of them had been executed, such a change would have been ignored. This demonstrates the major

characteristic of an INTERPRETIVE PROGRAM -- that translation of an instruction is performed each time the instruction is to be obeyed.

APPENDIX C

GRAMMAR ROUTINES

APPENDIX C

GRAMMAR ROUTINES

A restriction is a series of routines with their arguments which operate in the tree or any of the lists (grammar, word dictionary, sentence lists)[5]. The restrictions are part of the grammar and therefore determined by the grammarians. However, the function of the routines in the analyzer program will be described below. By means of these routines the tree or list structure may be examined for different properties, e.g. well-formedness of substructures. A restriction is itself represented in the machine as a list. Each routine in the restriction is executed in order; if any routine in the list fails, the restriction fails. When the restriction is encountered, the machine is 'looking at' either a node or a word in a list. If a restriction fails, it always returns to its starting point (node or list word); if it succeeds, it remains just where the last routine exited. If a routine fails, it also returns to its starting point or leaves the machine 'looking at' a different place depending on its function. Some routines must start at nodes and others at list words. Some can differentiate between the two structures and those may start at either place.

Some routines (e.g. AND) have the property of recursiveness, i.e. during the execution of this routine, a call is again made to the routine. Since FORTRAN does not support recursive subroutines the recursive routines are put into one subroutine and

a pushdown stack is used to store and restore the appropriate 'locations' needed for the proper operation of these routines. In this case, the locations are labelled FORTRAN statements.

A detailed alphabetical description of the routines will follow. In order to avoid unnecessary repetition the routines will be represented as functions and certain symbols will be used to describe various details.

Namely:

1) F(Z) = A routine that returns to its starting point only if it fails.
2) T(Z) = A routine that always returns to its starting point.
3) The following variables will describe the type of argument of the routine:
   a) a = a symbol
   b) y = a restriction list
   c) A = a list of symbols $a_1, a_2, \ldots, a_n$
   d) Y = a list of restriction lists $y_1, y_2, \ldots, y_n$
   e) 0 = no argument
   f) X = grammar register (special location available to the grammarian. It is used in a restriction for storing and retrieving nodes or list words.)
4) The subscripts V_1-V_2 attached to a T or F indicate where the machine must start (V_1 position) and where it will end (V_2 position).
   a) V = N represents a node

b) V = L represents a list word

c) V = 0 means that the routine's functioning is independent of the starting (V_1) or stopping (V_2) point.

Name of Routine		Type	Operation[1]
AND		$T_{O-O}(Y)$	Test that all y_i's exit +.
ATTRB	1.	$F_{N-L}(O)$	The current node NS must be atomic. Therefore, it corresponds to category S of the word which matches NS. Go to the sublist of category S.
	2.	$F_{L-L}(O)$	Go to the sublist of the current list word.
	3.	$F_{N-L}(A)$	Perform 1, then go down the sublist until an a_i is reached.
	4.	$F_{L-L}(A)$	Perform 2, then go down the list, until an a_i is reached.

[1] If the operation can be performed, the routine is successful and exits +; otherwise, the routine fails and exits -. If every routine in a restriction y exits +, then y itself exits +; if any routine in y exits -, y exits -.

Name of Routine		Type	Operation
BITIN*	1.	$T_{o-o}(a_1,a_2,a_3)$	Set (if $a_1 = 1$) or Reset (if $a_1 \neq 1$) the a_3^{th} bit of the halfword represented by the a_2 symbol.
	2.	$T_{L-L}(a_1,a_2)$	The computer is looking at a grammar list. This list contains numbers representing the bits of the a_2 halfword to be set ($a_1 = 1$) or reset ($a_1 \neq 1$).
BITT*	1.	$T_{o-o}(a_1a_2)$	Test that the a_2^{th} bit of the a_1^{th} halfword is set.
	2.	$T_{L-L}(a_1)$	The computer is looking at a grammar list. This list contains numbers representing the bits of the a_1 halfword that must set if routine is to pass.
CANDO		$T_{o-o}(Y)$	Test that y can be executed successfully.

---

* In BITIN, BITT, SETT, TSET the symbols used are such as to yield a number one hundred larger than desired. The numbers referred to in the description are the numbers after 100 is subtracted.

Name of Routine		Type	Operation
CHECF		$T_{o-o}(A)$	Test whether the current word has an a_1 on its category list.
CLSSL		$F_{L-L}(A)$	Go to the place in the present list that has an a_1 category.
COMEN	1.	$T_{o-o}(A_1A_2)$	Test that a symbol in A_1 matches a symbol in A_2.
	2.	$T_{o-o}(yA_2)$	Test that the following be done: Execute y successfully. If y leads to a node NS form a list A_1 consisting of the symbol S. If y leads to a list set the list equal to A_1. Go to 1.
	3.	$T_{o-o}(A_1y)$	Do step 2 for A_2.
	4.	$T_{o-o}(y_1y_2)$	Do step 2, then 3.
DNRIT		$F_{N-N}(0)$	Go to the rightmost node, one level below the current node.

Name of Routine	Type	Operation
DNTRN	$F_{N-N}(A_1A_2A_3)$	Descend to a node Na_{11} below the current node. Descent in the following manner: (1) Set m = 1; (2) Descend to an Na_{11} m levels below the current node, scanning from left to right. If there are none, set m = m+1 and go to (2). During the descent, if any node in level m is an Na_{31} or is nontransparent do not go below it, unless it is an Na_{21}. Exit "-" when a further descent is no longer possible, either because there are no more transparent nodes or because the lowest node of the tree has been reached.
DOWN	$F_{N-N}(0)$	Go to the node directly below the current node.

Name of Routine	Type	Operation
DOWN1	$F_{N-N}(0)$	Go to the node directly below the current node. The node below must have no node to its right (except for special process nodes).
DSQLF	$T_{N-N}((y_1 a_1) \ldots (y_n a_n))$	Test whether all y_i's can be executed successfully: 1. Set i = 1. 2. Empty all grammar registers. 3. Execute y_i successfully and return to the starting point. 4. Set i = i+1, go to 2.
DWNTO	1. $F_{N-N}(A)$	Go to a node Na_i below the current node using the same manner of descent as DNTRN except that all nodes are treated as if they were transparent.
	2. $F_{L-L}(A)$	Go to the place in the present list that has an a_i category.

Name of Routine	Type	Operation
EDIT	$F_{L-L}(y)$	The machine must be 'looking at' the first option of a list L of options. Generate a list L' of options from L in the following manner: 1. Set i = 1. 2. Look at the ith word of L (which points to L_i, the ith option of L). 3. Test whether y can be executed successfully. If so, put the ith word of L in L' and go to 4; if not, go to 4. 4. If there is another option in L set i = i+1 and go to (2); if not, go to 5. 5. Look at L'. It must have at least one option.

Name of Routine		Type	Operation
EMPTY		$T_{N-N}(0)$	Test whether the current node is empty. I.e. that no node in its substructure corresponds to a word of the sentence.
EXEC	1.	$F_{N-L}(0)$	NS is the current node; NS_{11} is below it. Look at the (i+1)-st word of the string S.
	2.	$F_{N-O}(RT)$	Perform 1, then get the restriction R_i and find the routine RT on R_i. If RT is found, execute its argument; if RT is not found, find the routine OLIST and execute its argument.
EXPNT		$F_{O-O}(y)$	Execute y.
FETCH		$F_{O-N}(0)$	Used in conjunction with FILLIN. Go to the node on top of the STAC pushdown produced by FILLIN. If STAC is empty, FETCH fails.

Name of Routine	Type	Operation
FILLIN	$T_{N-N}(a_1)$	For each a_1 in the substructure of the current node, place its location in the STAC pushdown. If there are no such nodes, FILLIN fails.
FIND	1. $F_{L-L}(RT)$	The present list should be a restriction list. Go to the place in the present list that has the routine RT. Look at the argument of RT.
	2. $F_{N-L}(RT)$	NS is the current node. NS_{11} is the node below it. Look at the restriction list R_i on option S_i. Go to 1.
	3. $F_{L-L}(RT(A))$	a) Perform 1. If it is successful, go to b; otherwise, exit -. b) Check whether there is an a_i on the argument list of RT. If there is, exit +; otherwise, go to 3a) to

Name of Routine	Type	Operation
FIND (con't.)		find the next RT on the restriction list.
	4. $F_{N-L}(RT(A))$	Perform 2. Then go to 3b).
FRSTL	$F_{N-L}(0)$	Go to the place in the sentence list corresponding to the word that was current just before the current node was constructed.
GENER	$F_{N-L}(0)$	The current node must be a special process node. At least one node NS_{ik} must be to the left of it. Generate an option list T, so that: $T_1 = \underline{S_{ik}}$ $T_2 = \underline{S_{ik-1} \text{ and } S_{ik}}$ $T_k = \underline{S_{i1} \text{ and } S_{i2} \text{ and } \ldots S_{ik}}$ Look at T.

Name of Routine		Type	Operation
IMPLY		$T_{o-o}(y_1y_2)$	If y_1 exits -, IMPLY exits +; if y_1 exits +, then IMPLY succeeds only if y_2 exits +.
INTOL	1.	$F_{N-L}(0)$	NS_{ij} is the current node. Look at the jth word of the option S_i.
	2.	$F_{L-L}(0)$	The current list word is pointing to another list. Go to that list.
ISIT	1.	$T_{N-N}(A)$	Is the current node an Na_i?
	2.	$T_{L-L}(A)$	a) If the current list word is an option: is the first element in the option an a_i? b) If the current list word is an element of an option: is the element an a_i? c) If the current list word is a symbol: is the symbol an a_i?

Name of Routine		Type	Operation
ITER	1.	$F_{o-o}(y)$	Execute y successfully at least once. Then keep executing y until it fails.
	2.	$F_{o-o}(y_1y_2)$	Execute y_1 successfully in the following manner: a) Execute y_1. If it is successful, ITER exits +. If it is not successful, go to b). b) Execute y_2. If it is successful, go to a). If it is not, ITER fails.
LASTL		$F_{N-L}(0)$	NS is the current node. Go to the place in the sentence list corresponding to the word that was current when NS was completed.
LEFT		$F_{N-N}(0)$	Go left until the first node which is not a special process node is reached.

Name of Routine	Type	Operation
LOOKT	$F_{o-o}(X)$	Go to whatever is stored in X. If X is empty, exit -.
MARK	$V_{o-o}(A)$	MARK must be retrieved and its argument A looked at by FIND. This enables the options of a string definition to be assigned properties which may be tested by other parts of the grammar. There may be several MARK routines (for different types of property lists) in one restriction. In that case A must contain a special symbol to identify which type of list it is. For example, if the symbol P identifies a list of prepositions that list may be obtained by using the following command: FIND(MARK(P)).

Name of Routine		Type	Operation
NEXTAT	1.	$F_{N-N}(a_1)$	Starting at atomic node. Go to next atomic node on tree.
	2.	$F_{N-N}(0)$	Starting at non-atomic node. Go to next atomic node on tree.
NEXTL	1.	$F_{L-L}(0)$	Go to the next list word.
	2.	$F_{N-L}(0)$	NS_{ij} is the current node. Go to the (j+1)-st word of option S_i.
NOATM		$T_{N-N}(0)$	Is the current node non-atomic?
NOT		$T_{o-o}(y)$	If y exits +, NOT exits -; if y exits -, NOT exits +.
NOTEL		$F_{o-L}(A)$	Look at list A.
ORR		$T_{o-o}(y)$	The y_i's are executed successively starting with y_1. As soon as any y_i exits +, ORR exits +. If no y_i exits +, ORR exits -.

Name of Routine		Type	Operation
ORPTH		$F_{o-o}(Y)$	The execution is identical to ORR. On completion, ORPTH remains where the successful y_i has brought it.
PARSE		$T_{o-o}(0)$	Was a parse obtained for this sentence?
PLACE*	1.	$T_{N-N}(0)$	Place EBCDIC representation of index term for current node into ARGMT buffer.
	2.	$T_{o-o}(a_1)$	The symbol a_1 is not the 500th element of symbol table. Place this number into ARGMT buffer.
	3.	$T_{L-L}(a_1)$	The symbol a_1 is the 500th element of symbol table. Place the value of the current list word into ARGMT buffer.

---

* The symbols a_1, a_2 are such that have the symbol table location of the desired number.

Name of Routine		Type	Operation
PLACE (con't.)	4.	$T(a_1,a_2)$	The symbol a_1 is the 500th element of symbol table. Place the value of the a_2 halfword into ARGMT.
PREVL		$F_{L-L}(0)$	Go to the previous list word.
RIGHT		$F_{N-N}(0)$	Go right until the first node which is not a special process node is reached.
SCOPE		$F_{N-L}(0)$	The current node must be a special process node. The node NS_{ik} would have been attached if the special process mechanism had not been interrupted. Generate an option list T so that: $T_1 = \underline{S_{ik}}$ $T_2 = \underline{S_{ik} \text{ and } S_{i(k+1)}}$ $T_{n-k} = \underline{S_{ik} \text{ and } S_{i(k+1)} \text{ and } \ldots \text{ and } S_{in}}$ Look at T.

Name of Routine		Type	Operation
SENTL		$F_{0-L}(0)$	Go to the place in the sentence list corresponding to the first word of the sentence.
SETT*	1.	$T_{0-0}(a_1,a_2)$	Set the a_1 halfword to the value of a_2.
	2.	$T_{L-L}(a_1)$	Set the a_1 halfword to the value of the current list word.
SPCFY		$F_{0-L}(S)$	Set up a list T (of options) composed of one option T_1. T_1 is composed of one element S. Look at T.
SPECF		$F_{N-L}((y_1a_1)\ldots(y_na_n))$	This routine must be on the restriction R_1 on S_1 and it always exits +. It will either find a substitute set of options for S, or leave S unchanged.

* In BITIN, BITT, SETT, TSET, the symbols used are such as to produce a number one hundred larger than desired. The numbers referred to in the description are the numbers after 100 is subtracted.

Name of Routine	Type	Operation
SPECT (con't.)		1. Set i = 1. 2. Empty all grammar registers. 3. Execute y_i. 4. If y_i is successful, the machine is now 'looking at' a substitute set of options. Ignore the remaining routines on R_1 and return to PARSER with a 'substitution' signal. 5. If y_i is not successful, set i = i+1 and go to 2.
STORE	$T_{o-o}(X)$	Store the address of the current node or list word in X.
SUBJR	$V_{N-O}(y)$	SUBJR must be retrieved and y executed by EXEC. y should be the path to the subject.
TRUE	$T_{o-o}(0)$	Always exit +.

Name of Routine	Type	Operation
TSET*	$T_{o-o}(a_1,a_2)$	Test that the a_1 halfword has the value a_2.
UPONE	$F_{N-N}(0)$	Go to the parent node of the current node.
UPTO	$F_{N-N}(A)$	Go to an Na_i above the current node.
UPTRN	$F_{N-N}(A_1A_2A_3)$	Go to an Na_{1i} above the current node; however, do not go above an Na_{3i} or a non-transparent node unless it is an Na_{2i}.
VERBR	$V_{o-o}(y)$	This is a non-executable routine. It must be found and y executed by EXEC. y is usually the path to the verb.
WELLF	$V_{N-O}((y_1a_1)$ $...(y_na_n))$	WELLF must be retrieved and its argument y executed by PARSER after NS is complete. Its

* In BITIN, BITT, SETT, TSET, the symbols used are such as to produce a number one hundred larger than desired. The numbers referred to in the description are the numbers after 100 is subtracted.

Name of Routine	Type	Operation
WELLF (con't.)		argument is executed in the same manner as that of DSQLF.
WORDL	$F_{O-L}(O)$	Go to the place in the sentence list corresponding to the current word W.

APPENDIX D

PRAGMATIC CATEGORIES

APPENDIX D

PRAGMATIC CATEGORIES

N41A - indicates general information

data, stuff, material, information

N41B - indicates definite article-type

article(s), book(s), work(s), number(s), paper(s), document(s), publication(s)

N42 - indicates bibliographic noun (used to set bibliographic command)

date(s), author(s), year(s), editor(s), issuer(s), title(s), writer(s), abstract(s), co-author(s), publisher(s), description

N43 - indicates word sequence type

word(s), phrase(s)

N45 - indicates expansion noun

example(s), illustration(s)

N46 - indicates system branch noun

N46A - file(s), system(s), library

N46B - thesaurus, lexicon

N46C - dictionary

N47 - subclass of N41 that may have accession number as a right adjunct

article(s), number(s), document(s), paper(s)

N48 - indicates definition

definition, meaning

N49 - indicates synonym

synonym(s)

N53 - indicates a possessive index term

Smith's, Jones'

BN - indicates sector designator noun

date(s), author(s), year(s), editor(s), issuer(s),

title(s), writer(s), co-author(s), publisher(s)

BVC - bibliographic verbs

authored, co-authored, appeared, deal, dated, deals,

dealt, dealing, edited, issued, listed, appearing,

dealing, produced, written, pertaining, published

RVC - relational verbs

relate, related, relates, relating

TVC - thesaurus verbs

begin(s), beginning, start(s), starting

SC - set-command verbs

written, authored, co-authored, published, issued,

edited, produced, write, edit, produce, author, issue,

publish, mean

BV - sets bibliographic command verbs

wrote, edited, produced, published, issued, written

APPENDIX E

SEM-VALUES

APPENDIX E

SEM-VALUES

Value	Word
1	written, authored, co-authored
2	edited
3	published, issued
4	dated
5	concerned
6	entitled
9	characterized, indexed
20	in
21	on
22	by
23	
24	from
25	between
26	during
27	before
28	after
29	under
30	about
31	since
32	earlier
33	around

Value	Word
34	to
35	of
38	as
40	prior
51	,
52	and
53	or
54	but
55	either
56	neither
57	both
58	as
59	well
60	not
61	only
62	also
63	nor
71	dealing, pertaining
72	appearing
73	concerning, regarding, describing, covering
74	having

Value	Word
81	file, library, system
82	that
83	teletype, printer
91	author(s), co-author(s), writer(s)
92	year(s), date(s)
93	title(s)
94	editor(s)
95	issuer(s), publisher(s)
96	field(s), area(s), topic(s), subject(s)
97	word(s)
98	publications, journal(s)
99	interval, period
101	written, wrote, write, authored, co-authored
102	edited, edit
103	published, produced, issued, publish issue, produce, co-author

APPENDIX F

SAMPLE QUERIES

APPENDIX F

SAMPLE QUERIES

1. Give me everything written, edited, or published by Jones.

   NUMBER (((AUTH JONES) + (EDIT JONES)) + (ISSR JONES)) **

2. What is generic to radar?

   RELATION (8) RADAR **

3. Do you have something about radar?

   NUMBER DESC RADAR **

4. Give me some synonyms of automobile.

   SYN AUTOMOBILE **

5. What does radar mean?

   DEFINE RADAR **

6a. I want anything by Jones.

   NUMBER AUTH JONES **

6b. How about Allen.

   NUMBER AUTH ALLEN **

7. I want something related to radar.

   RELATION RADAR **

8. What is radar?

   DEFINE RADAR **

9. Give me anything in the thesaurus starting with ABS.

   THES/x ABS **

10. Could I have data concerning the theory of salt with sugar?

   NUMBER DESC THEORY SALT SUGAR **

11. Give me anything on radar and anything on sonar.

    NUMBER DESC RADAR **

12. Give me everything on either sonar or laser.

    NUMBER DESC (SONAR) + (LASER) **

13. What books do you have on radar?

    NUMBER DESC RADAR **

14. What do you have on radar?

    NUMBER DESC RADAR **

15. Who has written anything on radar?

    NUMBER DESC RADAR ** AUTH **

16. Define radar.

    DEFINE RADAR **

17. Look up radar in the dictionary.

    DEFINE RADAR **

18. I want radar defined.

    DEFINE RADAR **

19. What has Jones written on radar?

    NUMBER ((AUTH JONES)) & (DESC RADAR) **

20. Give me the author of document 110.

    FORM 110 ** AUTH **

21. What have Jones and Allen written about radar?

    NUMBER ((AUTH (JONES) & (ALLEN))) & (DESC
    RADAR) **

22. What has Jones written?

    NUMBER (AUTH JONES) **

23. What do you have written by Jones?

NUMBER AUTH JONES **

24. I want anything related to wave propagation and time dependent transforms.

RELATION WAVE PROPAGATION, TIME DEPENDENT TRANSFORMS **

25. I want the author and date of publication of documents 110, 120, 130.

FORM 110 , 120 , 130 ** AUTH ** DATE **

26. I want the author and date of documents 110, 120, 130.

FORM 110 , 120 , 130 ** AUTH ** DATE **

27. Synonyms of radar.

SYN RADAR **

28a. Documents by Jones.

NUMBER AUTH JONES **

28b. And Allen.

NUMBER AUTH ALLEN **

28c. Smith

NUMBER AUTH SMITH **

29. Give me the author, title and issuer of something pertaining to radar.

NUMBER DESC RADAR ** AUTH ** TITL ** ISSR **

30. I want the bibliographic information of document 130.

FORM 130 ** DESC/BIBLIO **

31. What could I have written by Jones?

NUMBER AUTH JONES **

32. Please give me all of the documents written after 1950 about radar.

NUMBER (DATE 1950 + 1951 + 1952 + 1953 + 1954 + 1955 + 1956 + 1957 + 1958 + 1959 + 1960 + 1961 + 1962 + 1963 + 1964 + 1965 + 1966 + 1967 + 1968 + 1969) & (DESC RADAR) **

33. Give me everything between AB and AZ in the thesaurus.

THES/BW AB, AZ **

34. Give me anything written in 1950 by either Alan or Smithe.

NUMBER (DATE 1950) & (AUTH (ALAN) + (SMITHE)) **

35. I want anything by two of the following authors: Greene, Molden, Allen, Wills.

COMBINE (2) AUTH GREENE / MOLDEN / ALLEN / WILLS **

36. I want all the stuff Jones has written.

NUMBER (AUTH JONES) **

37. Anything by more than two but less than four of the following terms: radar, sonar, laser, maser, pacer.

COMBINE (G2AL4) DESC RADAR / SONAR / LASER / MASER / PACER **

38. What is the definition of radar, sonar, and laser?

DEFINE RADAR, SONAR, LASER **

39. What has Jones written, edited or published about radar?

NUMBER ((AUTH JONES) + ((EDIT JONES) + ((ISSR JONES)))) & (DESC RADAR) **

40. Give me any word around ST in the thesaurus.

THES/AR ST **

41. I want all that you have on radar, sonar and laser.

    NUMBER DESC (RADAR & SONAR) & (LASER) **

42. Give me anything radar is generic to.

    RELATION (7) RADAR **

43. What did Jones edit about radar?

    NUMBER ((EDIT JONES)) & (DESC RADAR) **

44. By not less than two of the following authors: Hopsy, Wilson, Pett, Robbin, Cyde.

    COMBINE (GE2) AUTH HOPSY / WILSON / PETT / ROBBIN / CYDE **

45. Indexed by more than two of the following terms: radar, sonar, laser, pacer.

    COMBINE (G2) DESC RADAR, SONAR, LASER, PACER **

46. By two or three of the following: AB, CD, EF.

    COMBINE (203) DESC AB / CD / EF **

47. Generic to radar, sonar and laser.

    RELATION (8) RADAR, SONAR, LASER **

48. I want all the words that radar, sonar and laser are generic to.

    RELATION (7) RADAR, SONAR, LASER **

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)
The Moore School of Electrical Engineering
Philadelphia, Pennsylvania

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
REAL ENGLISH: A TRANSLATOR TO ENABLE NATURAL LANGUAGE MAN-MACHINE CONVERSATION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Scientific Interim

5. AUTHOR(S) (First name, middle initial, last name)
Harvey Cautin

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May 1969	151	0

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 49(638)-1421	
b. PROJECT NO. 9769-01	
c. 61102F	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d. 681304	AFOSR 69-1481TR

10. DISTRIBUTION STATEMENT 1. This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Research Directorate of Information Sciences(SRI) Arlington, Virginia 22209

13. ABSTRACT

This dissertation presents a pragmatic interpreter/translator called Real English to serve as a natural language man-machine communication interface in a multi-mode on-line information retrieval system. This multi-mode feature affords the user a library-like searching tool by giving him access to a dictionary, lexicon, thesaurus, synonym table, and classification tables expressing binary relations as well as the document file representing the field of discourse. The user is thereby allowed a greater freedom in search strategy. Real English will 1) syntactically analyze the user's message by means of a strong analysis grammar to produce a tree representing the interrelationships of the grammatical entities comprising the message, 2) use this tree together with a pragmatic grammar to establish the set of commands necessary to fulfill the request, and 3) form the proper syntax for each command. The strong linguistic foundation of the syntax analyzer endows the system with the power of flexibility. As experience shows that certain new structures occur and should therefore be a part of the system, they may be incorporated into the system by the grammarian without a major overhaul of the procedures to date. The user is permitted to phrase his requests in any convenient form (i.e. declarative, imperative, interrogative, or fragmented sentence referred to as contextually dependent sentences). Thus instead of placing the user in the difficult position of learning a new language, the system is given the responsibility of responding in and to the user's language, i.e. the man-machine conversation in carried out in a natural language.

DD FORM 1 NOV 65 1473

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Cybernetics						
Information Science						
Artificial Language						
Computer Interface						
Computer Software						

UNCLASSIFIED

Security Classification