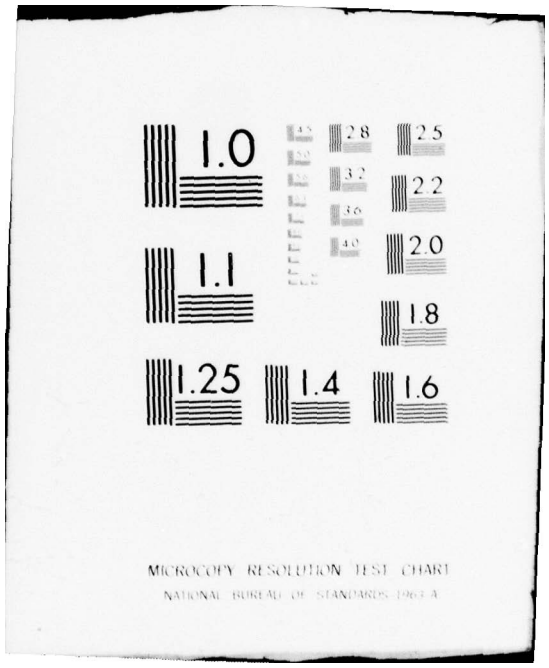
AD-A069 899 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA F/G 9/2
A NATURAL LANGUAGE QUERY SYSTEM TO AID IN NAVY COMMAND AND CONT--ETC(U)
JAN 79 C L BLAIS

UNCLASSIFIED NOSC/TR-374 NL

| OF |
AD A069 899

END
DATE FILMED
7-79
DDC


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL II (12)

NOSC

NOSC TR 374

(14) NOSC/TR-374

Technical Report 374

(6) **A NATURAL LANGUAGE QUERY SYSTEM TO AID IN NAVY COMMAND AND CONTROL.**

Concepts for modifying LADDER (language access to distributed data with error recovery) for the retrieval of information in an operational environment.

(10) C.L. Blais

(12) 28 p.

(11) 9 January 1979

(9) Final Report for Period: October 1977 — September 1978

(16) F21211

Prepared for
Naval Electronic Systems Command (ELEX 330)
Washington DC 20360

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

393 159 LB

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI, CAPT, USN
Commander

HL BLOOD
Technical Director

ADMINISTRATIVE INFORMATION

Work was performed by the Naval Ocean Systems Center ACCAT Program Office (Code 832), under program element 62721N, project F21211, NOSC work unit CF01, for Naval Electronic Systems Command (ELEX 330). This report covers work from October 1977 to September 1978, and was approved for publication 9 January 1979.

Released by
CDR DF Lesemann, Head
ACCAT Program Office

Under authority of
JS Campbell, Head
Command, Control, Communications
and Intelligence Systems Department

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NOSC Technical Report 374 (TR 374)

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: A NATURAL LANGUAGE QUERY SYSTEM TO AID IN NAVY COMMAND AND CONTROL
Concepts for modifying LADDER (language access to distributed data with error recovery) for the retrieval of information in an operational environment

5. TYPE OF REPORT & PERIOD COVERED: Final Report for Period October 1977–September 1978

6. PERFORMING ORG. REPORT NUMBER:

7. Author: CL Blais

8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Naval Ocean Systems Center
San Diego CA 92152

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
62721N
F21211
CF01

11. CONTROLLING OFFICE NAME AND ADDRESS:
Naval Electronic Systems Command
Code 330
Washington DC 20360

12. REPORT DATE: 9 January 1979

13. NUMBER OF PAGES: 25

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:

15. SECURITY CLASS. *(of this report)*: Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*:
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:

18. SUPPLEMENTARY NOTES:

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:
Command and control systems
Information retrieval
Data management
Programming languages – natural language

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*:
The LADDER natural language query system was evaluated in the Advanced Command and Control Architectural Testbed (ACCAT) facility at NOSC San Diego. The operational utility of LADDER with respect to command and control applications was assessed in an experiment using Navy officers as subjects. A second effort, the subject of this report, considered the technological capabilities demonstrated by LADDER and provided insights into advanced concepts contributing to the enhancement of query systems in general.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

OBJECTIVE

Investigate the technological capabilities demonstrated by LADDER. Provide insights into advanced concepts contributing to the enhancement of query systems in general.

RESULTS

1. Maintaining large lists of proper names in a grammar is unnecessary and wasteful because of the specificity of operational contexts. It is more logical to base the placement of identifiers in the grammar on user-system interactions, supporting particular contexts as required.

2. Adaptation of grammatical constructs and internal views of data base structure can be developed to reflect user-oriented contexts. Interactive and automatic generation of profiles for information retrieval tasks can assist the user by anticipating his information requirements.

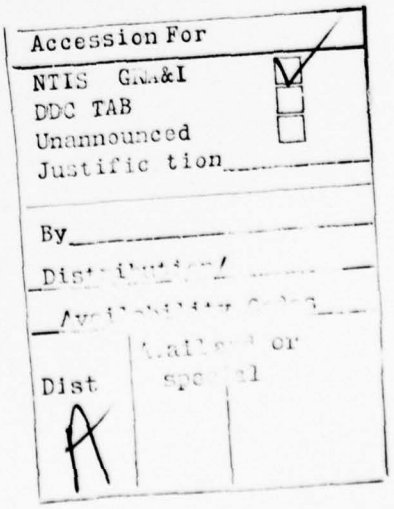
3. Development of software to profile an individual's capabilities and approaches for solving problems in specific tasks can further assist the user. Such software can provide a "user friendly" environment – one that is tailored to each individual – in which to perform those tasks.

4. Taken together, these conclusions form the basis for conceptualization of "intelligent" query systems or "intelligent assistants" that can delineate the scope of an information retrieval task, can perform routine, preliminary footwork to establish situation contexts for the user, and can provide assistance to user interactions and query processing through intelligent guidance and organization of information.

RECOMMENDATIONS

1. Use LADDER as a tool to investigate these concepts in experiments with actual users.

2. Conduct such investigations to validate the theoretical foundation for these concepts and to examine resulting effects on performance of information retrieval tasks.



CONTENTS

INTRODUCTION

During FY 78, the LADDER (language access to distributed data with error recovery) natural language query system was evaluated in the Advanced Command and Control Architectural Testbed (ACCAT) facility at NOSC San Diego. The operational utility of LADDER with respect to command and control applications was assessed in an experiment using Navy officers as subjects (ref 1). A second effort, the subject of this report, considered the technological capabilities demonstrated by LADDER and provided insights into advanced concepts contributing to the enhancement of query systems in general.

The use of natural language for querying data bases has been a widely researched issue for a number of years (see natural language bibliography). Although several experimental systems have resulted from this research (ref 2-4), few have been applied to the operational needs of specific users. One exception is the LADDER (language access to distributed data with error recovery) system, which was developed as a management aid to Navy decision makers by SRI International (ref 5).

Underlying capabilities of LADDER, however, exhibit a range of application much broader than it was designed for. LADDER is therefore useful for the evaluation of query system features in general as well as natural language systems.

NOSC, San Diego, performed rather extensive test and evaluation of the LADDER system during FY 78 as part of the ACCAT project, jointly funded by DARPA and NAVELEX. The evaluation progressed along two mutually supportive assessments: (1) the operational utility of LADDER with respect to its interactions with Navy users, and (2) current and advanced capabilities of LADDER deemed appropriate to Navy command control systems. The first assessment (ref 1) provided valuable insights into the utility of a natural language system and indicated unexpected eccentricities in the man-computer dialogue. This report focuses on the second assessment. It addresses a number of issues that could ultimately enhance the operational use of LADDER. In fact, many features discussed herein are not restricted to LADDER, or even to natural-language-based systems. Concepts of particular interest deal with user interactions, certain grammatical constructs, and data access techniques. These concepts imply a query system design which emphasizes user supportive responsibilities of the query system and of the data base management system.

---

1. NOSC Code 8321 ltr rpt, Performance of a Natural Language Query System in a Simulated Command Control Environment, by HG Miller, RL Hershman, and RT Kelly, May 1978.

2. REQUEST: A Natural Language Question-Answering System, by WJ Plath; IBM Journal of Research and Development, vol 20, July 1976, p 326-335.

3. An English Language Question Answering System for a Large Relational Database, by DL Waltz; Communications of the ACM, vol 21 no 7, July 1978, p 526-539.

4. BBN Report 2378, The Lunar Sciences Natural Language Information System: Final Report, by WA Woods, RA Kaplan, and B Nash-Webber; Bolt, Beranek and Newman, Inc, Cambridge MA, 1972.

5. Final Technical Report, Project 4763, Mechanical Intelligence: Research and Applications, by ED Sacerdoti (ed); SRI International, Menlo Park CA, December 1977.

This report is conceptual and is not to be construed as a preliminary statement of functional specifications for a Navy query system. However, a fundamental goal of ACCAT is to "shape and influence the research and development of appropriate information processing technology such that it is potentially transferable to the Navy for command control applications" (ref 6). Opinions of the operational community need to be shared with researchers throughout the evaluation process. This report is one attempt to fill that need.

LADDER has been established as an extremely flexible tool for the investigation of query system features. It will be used in future experiments with operational personnel to test alternatives to a purely natural language front end. Nearly all of the features and enhancements described herein are considered to be within the state of the art in artificial intelligence and information processing and could conceivably be demonstrated by using the LADDER system. (As will be explained, some of these features have already been demonstrated.) Much of LADDER's evolutionary growth has been in response to experimental results and observations from NOSC analysts, illustrating that interaction between research and operational communities in the early stages of system development can prove beneficial.

INTELLIGENT INTERPRETATION OF USER INPUTS

A natural language system such as LADDER faces a much greater challenge in correctly interpreting user inputs than do the menu select, function key or structured English approaches, in which the range of input is generally well-defined and can be readily anticipated by the system. The inherent flexibility of natural language prohibits the system from knowing what to expect when a user initially interacts with the system or as he proceeds from one query to the next. Even though LADDER is concerned with a specific conceptual domain – Navy command and control – and thereby limits query context somewhat, the structure of user inputs is seemingly endless, varying from simplistic "unnatural" queries ("WHAT NATION PECOS") to more complicated structures ("WHAT SHIPS WITHIN 500 MILES OF THE PECOS HAVE A DOCTOR ABOARD") (ref 1). LADDER developers attempted to overcome this problem by providing (1) a reasonable coverage of user syntax with respect to the given context (semantically-oriented grammar (ref 5)) and (2) the capability for grammar expansion, to incorporate specific user-idiosyncratic constructs. These two provisions together provide the user with total syntactic coverage, although for complex structures he must have considerable knowledge of the generative methodology for the grammar. (In such cases personnel responsible for system maintenance generally are called upon for assistance.) Maximum syntactic coverage can best be provided through extensive system testing at operational command and control centers, where users interact with the system in their own jargon and task-related terms. Grammatical modifications would then reflect user-specific requirements. Although it would be desirable for such testing to be done in simulated command and control centers in the NOSC ACCAT facility, there is no way to duplicate the variety of situations and information retrieval requests that occur in the actual environment.

One implementation issue facing LADDER as well as other query systems (particularly those providing some form of automatic validation of user input) is the inclusion of proper names (ships, ports, aircraft types, surface tracks, etc) or identification codes in the grammar itself. Specification of unique identifiers serves a number of useful purposes. Examples:

Incorrect spellings often can be recognized and corrected.

---

6. NOSC Code 8321 ltr rpt, Advanced Command and Control Architectural Testbed Concept of Operations Plan, vol 1, July 1977.

The context of the query can be deduced by recognizing the classification or category to which the identifier belongs.

Implicit assignment of context based on the interpretation of identifiers enhances data retrieval by supporting the translation of user input to commands understood by the data base management system. The context designates what files are to be accessed and what attributes or fields are to be retrieved. For example, although the attribute, MAXIMUM SPEED, is applicable to both ships and aircraft, each file might have its own name (eg MAXSPEED in the aircraft file, MCSF in the ship file). Recognition of the object provides the appropriate context from which the data base query can be generated. Without this information, the query system would be forced to search several files for the specified identifier. If the grammar is to recognize identifiers, however, it must maintain lists containing potentially thousands of such items. This places a tremendous burden on the system developers and the computing resources. Moreover, provision must be made for the updating of dynamic elements (flights, surface tracks, air tracks, etc) either through direct interaction with the data base management system, in response to the presence of certain "alert" criteria, or through operator action based on other sources of information or on operator knowledge of values stored in the data base. The latter method places considerable burden on the user and increases the likelihood that inconsistencies could arise between the data base and the grammar's knowledge.

An apparent dichotomy exists: the grammar should be smart enough to recognize and understand (ie determine the context for) proper names of static and dynamic objects, but should not have to maintain gigantic lists of such names, that is, to remember what the objects are and what they are called. A partial resolution of this problem is suggested by considering the user environment in which LADDER or some other query system might be employed. Very simply, no operator will require information on *all* objects in the data base at any one time (if ever). Instead, each command and control center is concerned with a specific area of interest and with a limited subset of information contained in the data base. Suppose, then, that the query system has no a priori knowledge of object identifiers but will learn them through interaction with the user. For example, an operator may ask, "WHERE IS THE CONSTELLATION?" In LADDER, the query cannot be parsed unless the name is in the grammar. More appropriately, the query system could respond as follows:

THE IDENTIFIER, CONSTELLATION, IS NOT
RECOGNIZED BY THE SYSTEM.
DOES IT DEFINE A

A. SHIP?
B. PORT?
C. AIR TRACK?
D. SURFACE TRACK?

.
.
.

The user then specifies the intended interpretation, whereupon the system can properly categorize the name and formulate data retrieval commands for processing by the data base management system. The identifier would become a permanent element in the grammar (at least for the duration of the current transaction), available for use in subsequent queries without further specification.

Query responses also contain information relating to the user's context. If an operator asks "WHAT SHIPS ARE WITHIN 500 MILES OF NAPLES?" he is obviously indicating some interest in the ship names which will be returned. This list of names can be intercepted briefly by the query system, to incorporate these names into the grammar, then passed on to the user. By these two methods – interacting with a user to provide interpretation of unknown object names and presuming that response information reflects the user's scope of interest – a contextual framework is generated within the grammer for that particular transaction. This modified grammar will be referred to as a "contextual" grammar. Instead of the grammar's having to be so general as to require no human assistance in understanding queries, the system becomes adaptive, responding to the needs of specific users or classes of users. Once defined, a contextual grammar can be preserved for later reference. A command and control center could conceivably possess a library of such grammars, each appropriate to specific tasks of individual users (eg routine reports or briefings) or to overall missions assigned to a staff of users (eg transit planning, search and rescue operations, etc). Contextual grammars are fundamental to the concept of mission profiles, to be discussed later in this section.

Some problems are anticipated in implementing an adaptive grammatical scheme as introduced above. First, the specified name in a query (eg "CONSTELLATION") might not be a valid name in the data base. To illustrate this situation, suppose a user inquires about the carrier JOHN F. KENNEDY, although the name stored in the data base is really "KENNEDY JF" or even "JFK." The data base management system would be unable to find any data for the "JOHN F. KENNEDY." A possible solution is to create a list of translations from "synonyms" to proper data base values. Although this list of associated names could conceivably become quite large – in contrast to the grammar it may be desirable to create just one such file for all names – its relative stability would permit efficient structuring of the data for rapid access of any record. Uncommon, user-specific synonyms such as "MYBOAT" for "CONSTELLATION" could be defined in the grammar itself (LADDER has this capability), so that an initial translation to a common name can occur when the query is parsed. Morever, following interaction with the data base management system, the actual data base value could be substituted for the common name, providing a one-step translation to a value consistent with the data. This procedure would preserve the stability of the file of associated names, would save query processing time for future references to that name, and would contribute to the concept of grammatical adaptation to the user's specific needs.

A second problem involves the definition of appropriate query syntaxes that would incorporate unknown or indefinite patterns. When the input is parsed, the presence of these patterns would cause the system to initiate interaction with the user to determine the proper interpretation of unknown identifiers. Investigations at NOSC/ACCAT have demonstrated the feasibility of this approach through definition of a top-level pattern in LADDER:

⟨WHAT⟩ ⟨BE⟩ ⟨ATTRIBUTE⟩ ⟨OF⟩ ⟨INDEF.NAME⟩

where INDEF.NAME can be matched by a variety of words or phrases, such as

(1) a known (ie already listed in the grammar) ship, port, track, flight, etc

(2) VESSEL name
FLIGHT number
TRACK NUMBER identifier
(Here, the name or identifier is qualified by an explicit descriptor. If the name is not known to the grammar, it is automatically added to the appropriate list.)

(3) an arbitrary word or phrase

In item (1) no interaction with the user is necessary; the object name has been defined previously. In item (2) the same is true if the object name is known to the grammar; otherwise, the system may want to ask for confirmation from the user (in case of spelling error) before placing the name into its proper category. To handle item (3), the system interacts with the user in a manner similar to that shown earlier. Grammatical modifications used to perform these actions are given in appendix A.

The same principle applies to an indefinite name that occurs in the interior of a query (eg "HOW FAR IS XYZ FROM HONOLULU?"). In LADDER, the parser would be unable to continue past the word "XYZ," assuming it is not part of the grammar. At that point, it may be possible to present the query tail "XYZ FROM HONOLULU" to the operator, to ask him to enter the word (or phrase) he intended to use as the object name, to ask for definition (categorization) of the object name, to enter the name into the appropriate category, to remove the word(s) from the left end of the tail, then to proceed with the parsing. Although the programming details are not trivial, this approach appears feasible. Top-level syntaxes (at least in a LADDER-like system) would become more general, with context-specific patterns (eg ⟨SHIP⟩ and ⟨PORT⟩) replaced by ⟨INDEF.NAME⟩ productions. In this respect, the grammar may become easier to modify and maintain.

In the above discussion, it was contended that recognition of context facilitates translation of user input into the language of the data base management system. In LADDER, user queries are actually translated to an intermediate language, called the IDA (Intelligent Data Access) query language, which is then used by the IDA subsystem to generate Datalanguage (ref 5, 7, 8). Unless the context is recognized, the query "WHAT IS THE SPEED OF XYZ?" is difficult to translate. If XYZ is a known ship, for example, the query should be translated to the IDA query

(? PTS) (NAM EQ 'XYZ').

To obtain the answer, the resulting Datalanguage would calculate the join of the SHIP and TRACKHIST relations by using the common attribute of UIC (Unit Identification Code) or VCN (Vessel Control Number). If XYZ is an air track identifier, on the other hand, a secondary translation of attribute PTS (the only grammatical translation of the word "SPEED") is necessary to access attribute SPEED of the AIRTRACK relation. Thus, although IDA may receive the query

(? PTS) (ATRACK EQ 'XYZ'),

it (currently) will not generate correct Datalanguage, even though IDA knows that SPEED is the synonym for PTS in the AIRTRACK relation. If the front-end grammar were to become more aware of the context, the problem could be avoided completely by issuing the proper query

(? SPEED) (ATRACK EQ 'XYZ')

to IDA, explicitly indicating what file to access to obtain the response.

Ambiguity resolution clearly represents one of the most difficult problem areas in a natural language system. Given a query satisfying more than one interpretation, LADDER does have the capability of displaying possible interpretations to the user, who can thereupon select the correct one for processing. LADDER does not normally operate in this mode, however, since determination of all possible interpretations is time-consuming. Moreover, it

---

7. Datacomputer Version 5 User Manual; Computer Corporation of America, Cambridge MA, July 1978.

8. The Datacomputer – A Network Data Utility, by T Marill and D Stern; AFIPS Conference Proceedings, vol 44, May 1975, p 389-395.

is assumed that the incorrect response will encourage the user to rephrase his question less ambiguously. By and large, the first interpretation generated by the system is the best anyway; presumably, little would be gained by the extra processing time of the alternate mode. This trade-off is unavoidable, at least until the natural language query system has better understanding of context and nuances of meaning. Since this discussion is primarily concerned with modifications to LADDER that involve moderate effort but are hoped to result in substantial improvements in operational utility, ambiguity resolution remains an open issue.

Although a natural language system cannot in general anticipate user queries, there are circumstances for which it can. For example, an operator may ask "WHAT SHIPS ARE WITHIN 500 MILES OF THE PECOS?" To determine the response, the system finds first the position of the PECOS, then the positions of all other ships. For each ship, the distance to the PECOS is calculated. Those within 500 miles are stored in the response list. Since, in this case, the data base management system has to access ship positions in the TRACKHIST file, there is little cost (ie time) involved in also retrieving auxiliary information such as course and speed from that file. Furthermore, the SHIP relation was also accessed to determine ship names from the positional information. (TRACKHIST uses ship UIC and VCN, not the ship name per se.) Therefore data from that file could be added to the auxiliary buffer (such as ship class, nationality and type). The extra attributes are part of the tuples already located for the essential information; no extra searching time is necessary to find these values. When the user is presented with the requested data, such as simply a list of ship names in the above example, the system could ask "DO YOU CARE TO SEE MORE INFORMATION?" If the user answers "yes," the system would display the amplifying information immediately without having to pass a new IDA query to the data base management system for processing. For general queries, the system would have to be much more selective to ensure (1) that the amplifying information does not repeat data contained in the query response (except for identification of the objects in question) and (2) that the additional data relate to the correct objects (those contained in the response) rather than to intermediate values used to calculate or qualify the result. For example, if the user asks "WHAT SHIPS ARE FASTER THAN THE FASTEST US SUB?", no amplifying information should be retrieved for the "FASTEST US SUB."

LADDER has the capability to refer to a previous query to resolve pronominal references and elliptical, or incomplete, queries. Additionally, a previous query, numbered by the INTERLISP system, can be reentered and processed by using the REDO command, specifying the query number or range of query numbers to be "redone." (See ref 9 for a description of the REDO command.) For this capability to be truly useful, the operator would need to preserve earlier portions of his transaction on hard copy to refer to those numbers. Frequently used queries or a complicated series of questions can be defined as a single query by using the macro-paraphrase feature of LADDER. For example, the following questions might commonly be asked in a search and rescue operation in which PECOS is assumed to be the name of a distressed vessel:

WHAT SHIPS ARE WITHIN 500 MILES OF THE PECOS?
WHAT IS THEIR READINESS?
WHICH OF THESE SHIPS HAVE A DOCTOR ABOARD?
HOW LONG WOULD IT TAKE THESE SHIPS TO REACH THE PECOS?

The simple statement "SARSITUATION AROUND PECOS" can be defined as a paraphrase of the given series of questions. Moreover, the question is not restricted to the PECOS; if

---

9. INTERLISP Reference Manual, by W Teitelman; Xerox Palo Alto Research Center, Palo Alto CA, December 1975.

any other ship name is used, LADDER will make the appropriate substitutions. This capability is analogous to the stored query capability of the operational NWSS (Naval WWMCCS Software Standardization) Query Module (ref 10). In that system, queries composed in fixed format can be placed into a Stored Query File for future use. When recalled, the queries may be modified and/or passed to the Query Module for processing. In each of the above requests, LADDER recomputes the distances from all ships to the PECOS to satisfy the pronominal references to "SHIPS WITHIN 500 MILES OF THE PECOS." It would be more efficient for the query system to place the actual ship names satisfying the given qualification into its memory and to use that list to determine the responses to subsequent references.

The features just discussed comprise the ability of the query system to remember prior interactions and to provide recall so as to simplify operator composition of new or repeated queries. Elliptical and pronominal references represent an immediate recall of the user's inputs; macro-expansions (ie stored queries in the Query Module context) are a more permanent recollection of the user's information retrieval needs. When elliptical and pronominal references are used in conjunction with the grammatical modifications discussed previously, a query system can be envisioned that would no longer be a passive recipient of user requests but an active, intelligent assistant that aids the user in the performance of his duties. Based on a given task, an appropriate grammar would be selected, either automatically or by the user, to profile the user's information requests. Such a mission profile would be defined by the contextual grammar and by stored macro-expansions. The latter would allow the user to easily obtain general, task-oriented information typically of import to the given mission. Additionally, amplifying information would be stored for the user's attention should he desire to view it. *The grammar itself* would further adapt to the relevant context throughout the interaction by determining what ships or aircraft are involved, what sensor data are necessary, what tracks are pertinent, etc. In short, the query system would provide the environment, or mission profile, by which the user could easily and efficiently accomplish his information retrieval tasks. The user would retain the option of asking ad hoc queries in order to obtain further insight into the given situation, to retrieve data unrelated to the current context, or to establish a new context altogether. With the traits of adaptability, contextual awareness, and flexibility thus incorporated, the query system becomes responsive to the user's needs, thereby providing the intelligent interpretation of his requests essential to effective utilization of the system.

INTELLIGENT DATA ACCESS

Beyond having the "intelligence" to comprehend user inputs, a query system can exercise some degree of judgment in the actual access and retrieval of data. Data discrimination (providing only what is asked for) and optional response amplification, as introduced in the preceding section, are two examples of such judgment. The IDA subsystems of LADDER have been found to be adequate for a number of applications, but somewhat difficult to use when applied to the Navy data base resident in the NOSC/ACCAT system. This became apparent during efforts at NOSC/ACCAT to expand both the front-end grammar and the IDA schema (IDA's notional model of the data base structure) from the limited view supported by the Bluefile data base (ref 11) to a more complete Navy context

10. NAVCOSSACT Document 07A001A UM-04, Navy WWMCCS Software Standardization: Book 3 Query Module User Manual; Naval Command Systems Support Activity, May 1976.

11. NOSC Code 8321 ltr rpt, The Relational Model for the Bluefile Data Base (Revised), November 1976.

represented by an at-sea command and control center data base (ref 12). The problems encountered during those efforts will be discussed in this section. Furthermore, possible approaches to data access to solve these problems will be suggested.

Currently, IDA contains a very straightforward algorithm for determining what files (ie relations) are to be used to obtain the data requested in a query. (As discussed previously, the user's query is translated to IDA query format by the front-end grammar.) At each iteration of the algorithm, the system possesses a list of relations already chosen (empty in the first iteration) and a list of fields in the query which have not yet been covered by chosen relations, ie that cannot be found in the attribute lists of the relations already selected. The next uncovered field in the query is selected, and the algorithm attempts to find a relation that

(1) covers the selected field,

(2) has a direct link (via a common attribute) to a relation already chosen (if one exists), and

(3) covers as many uncovered fields as possible.

IDA processes the query from left to right, attempting to minimize the number of relations which will need to be accessed. For example, given the relations (and associated attributes)

SHIP: (NAME CLASS TYPE PTP TPD CONVOY)
SHIPCLASS: (CLASS TYPE LGH DRAFT)

IDA accesses only the SHIPCLASS relation to determine

(? LGH) (CLASS EQ 'BELKNAP')

(ie WHAT IS THE LENGTH OF BELKNAP CLASS SHIPS?), since both LGH and CLASS are found in that file. For the IDA query

(? LGH) (NAME EQ 'FOX')

(ie WHAT IS THE LENGTH OF THE FOX?), IDA generates two retrieval requests to the data base, for simplicity also shown in IDA query language format, of the form

IN SHIP RELATION: (NAME EQ 'FOX') (? CLASS)
IN SHIPCLASS RELATION: (? LGH) (CLASS EQ 'BELKNAP'),

where (CLASS EQ 'BELKNAP') was the response to the initial query. In this example, IDA builds a reference to both the SHIP and SHIPCLASS relations, essentially generating the join of these two relations over the common attribute CLASS. This file linkage information is also contained in the structural schema known to IDA. In actual system testing, however, it was found that IDA does not always consider files having direct linkage to ones already chosen, but simply selects any file connected to the chosen file and containing the requested field. The current IDA does not determine the shortest path to use for data retrieval, but instead selects whatever path it finds first (whether it is a direct path or an indirect one through several intermediate nodes). Although an optimization procedure in such a case may be time-consuming, processing time spent for that purpose may be considerably less than the data retrieval time that results from a long access path in which the values of intermediate links must also be found in the data.

The source of this problem relates to problems discussed in the previous section. When the front-end grammar is given the request "WHAT IS THE SPEED OF XYZ?" the attribute "SPEED" is translated to a specific field name used in the data base. For example,

12. NOSC Code 8321 ltr rpt, Expanded Relational Models for the Fleet Command Center and At-Sea Commander's Databases, March 1977.

if XYZ is a ship, the current speed of XYZ is found in attribute PTS of the TRACKHIST relation. (The examples to follow will refer to a modified Bluefile data base structure created at NOSC/ACCAT.) Without a context-dependent translation, the grammar makes the same translation for SPEED (ie to attribute PTS) if XYZ is an aircraft type or a flight number. Context-specific patterns, on the other hand, would ensure that aircraft attribute names would be associated with aircraft, flight attributes with flights, ship attributes with ships, and so on. The trade-off is in the simplicity of a generalized grammar, as compared to problems of development and maintenance of a context-specific grammar. One attempt at a compromise is exhibited by reconsidering the above example. If XYZ has been interpreted as an aircraft type, IDA is given the query

(? PTS) (AIRNAME EQ 'XYZ').

In the IDA schema, PTS is defined as an attribute belonging to relations TRACKHIST and AIRCRAFT, but having a synonym of, say, MAXSPEED in the AIRCRAFT relation. This allows IDA to make the necessary translation to the context-specific attribute. Operationally, however, IDA has been found to be in error in similar situations, incorrectly answering the query

(? PTS) (NAM EQ 'XYZ')

(when XYZ is a ship) by accessing the AIRCRAFT file for PTS. Since the attribute PTS is not found in the SHIP relation, the first relation found to cover the field is selected, rather than the first relation directly linked to the one containing the specified field. If the order of the query is changed to

(NAM EQ 'XYZ') (? PTS)

it will be processed correctly, suggesting a modification to the algorithm whereby relations covering qualified fields are selected first, relations linked to those selected by a common attribute are next checked for uncovered fields, then relations indirectly linked through one or more intermediate relations are checked for remaining uncovered attributes. By appropriate search procedures, it may be possible to actually determine the shortest access paths to the required data in terms of the number of relations accessed, including intermediate relations.

If the context is recognized by the grammar (eg a specified aircraft name is associated to "AIRNAME" in the IDA query), the simplest solution may still be to perform the necessary translation of the attribute category (eg SPEED) to specific attribute labels associated with the identified context. The translation could occur before or after IDA is called to process the IDA query language translation of the user's input. Nonetheless, the above discussion covers a more general situation: no matter how well the query context is specified, multiple file accesses through one or more intermediate relations will always be a possibility. Efficient determination of the shortest access paths would enhance system operation in all cases.

Grammatical adaptation to user context, as proposed in the previous section, leads to consideration of the possibility of dynamic data base conformity to user requirements. At least, the query system's "view" of the data base structure could adapt to context by concentrating strictly on those access paths recognized as satisfying the user's information needs. By restricting the number of relations used, shortest path determinations become considerably more efficient, particularly when the total number of relations in the data base is large. Furthermore, the data base remains unchanged; only the internal model of the data base structure changes. In the IDA context, the system would construct a data submodel on the basis of user queries, gradually shifting from the total structural model to the submodel as

the latter becomes more capable of covering all fields of the user's requests. Using a modified covering algorithm, as discussed above, undefined links or attributes encountered during processing would signal an incomplete representation of the structure in the submodel, whereby the total model would be used to determine the response. Links, relations, and attributes in the total model used to generate the query response but not yet incorporated into the submodel would then be added to the submodel, providing a broader coverage for subsequent queries. The resultant structure would support the specific information retrieval task defined by the user. As with contextual grammars, context-specific submodels of the data base structure provide fundamental building blocks of the mission profile concept.

An even more ambitious goal is to create a context-specific partition of the data base itself — either a physically separate substructure or a collection of pointers that allow rapid and direct access to portions of the data base of interest to the user. The first alternative requires creation of substantial updating features to ensure the integrity of data in the separate partition. This is similar to the update problem for a network of distributed data bases, many of which have certain data elements in common. Maintaining data integrity and currency in a distributed data base is a nontrivial problem. The Query 3 system, which is a structured English query system,* creates a "snapshot" of a portion of the data base by copying entire tuples of a relation as specified by the user. No attempt is made to keep the values of those elements current with the main data base, however. The second alternative demands a great deal of knowledge on the part of the query system, specifically with regard to the physical location of data in the machine, and requires a detailed interface with the data base management system. The advantage, of course, is that although data are not changed or copied, explicit access paths to the data of interest are created, greatly reducing the long search and retrieval times which can occur for large files. Implementation details for such a scheme are beyond the scope of this paper, particularly with regard to specification of necessary interactions between — in this case — LADDER and Datacomputer. The concept does appear worthy of investigation prior to defining firm query system requirements for Navy command and control use.

INTELLIGENT SUPPORT OF USER INTERACTIONS

The previous two sections have discussed three methods by which a query system can assist a user semantically: by correctly interpreting the meaning of his inputs, by facilitating the retrieval of desired data, and by creating an environment to fulfill specific mission or task-related information requirements. The discussions dealt with how the system is to process queries, thereby focusing on the internal functions of the query system. In this section, the emphasis shifts to the system "front end," the user-system interaction itself, although the presentation does not address the physical man-machine interface (ie how the user interacts with the terminal). How the system interacts with a user is of primary concern since it determines how "comfortable" the user will be with respect to his emotional and mental state of mind. Many features can be incorporated to create a user-friendly environment, such as prompting, tutorial assistance, user-specified formats for output responses, meaningful and precise error diagnostics, protection against system failure, auxiliary terminal activity during long query processing and retrieval

*A version of Query 3 is resident on the University of Southern California Information Sciences Institute System E DEC KL 20/40 (ARPA node 116); on-line documentation is available on that system.

delays, and so on. Since combinations of such features can be found in nearly any user-interactive system, the implication exists that making the environment more benign enhances system utility. This contention is not argued here; it is accepted as intuitively obvious.

Conceivably, the user-system interaction could become even more friendly if it could characterize the capabilities of each user in some way and, on the basis of this characterization, could provide a level of assistance and guidance commensurate with the preceived abilities and limitations of the user. The characterization, or "profile" (hence, "user profile"), would be generated from previous interactions with the user. Evaluation criteria could be defined to assess the user's capabilities with respect to various aspects of system operation and his knowledge of system features. Using this model of the user's capabilities, the system would control certain portions of the user-system interaction, such as prompting, tutoriai assistance, and other areas for which the system knows that the user requires aid. The profile, therefore, would not only reflect specific features desired by and for the user in the traditional sense, but would also contain the system's understanding of the user's current level of proficiency, frequency of use (to determine whether a refresher course is warranted after a long period of inactivity) and other characteristics.

The user profile is clearly distinct from the concept of mission profile in that the former characterizes the user and his preferences and abilities, whereas the latter characterizes the user's areas of concern, data access paths, and query grammer appropriate to specific missions and retrieval tasks. The user profile concept may be expanded even further: it may be possible for the system to generate a model of the conceptual approach to a problem as exhibited by a user, particularly with regard to specific types of missions or tasks performed frequently by the user. Repetition of tasks provides a statistical base from which to generate the profile. By observing the user's problem-solving methodology and by creating a model of these processes, the system would become an intelligent assistant in subsequent tasks, prompting for information or automatically presenting data in the logical sequence appropriate to that user's conceptual outlook. Together with mission profiles, then, the system would contain personalized problem-solving approaches for the accomplishment of specific tasks, adapting to changes in the user's domain of interest as well as to modifications of the user's analytical approach.

The achievement of mission and user profiles, as described herein, certainly faces more rigid constraints than the reaches of one's imagination. The nature of those constraints, however, has not yet been determined. At present, implementation of mission and user profile concepts, at least on a preliminary level, is considered to be within current computing capabilities. Within an experimental environment, such as that provided by the ACCAT facility, the utility of such applications of artificial intelligence to query processing in a Navy command and control environment can be measured and evaluated, thereby supporting the generation of functional requirements for future systems.

CONCLUSIONS

LADDER has been found to be an extremely useful tool both in determining the utility of natural language query systems for information retrieval in a command and control environment and for studying user-supportive features which may enhance the utility of any query system. These features induce the concept of an intelligent assistant: a query system which actively aids the information retrieval process by (1) delineating the scope of the problem, (2) performing the routine, preliminary footwork that establishes the context for the user, and (3) assisting the user's interactions and query processing through intelligent

guidance and organization of information. LADDER provides an effective framework from which the feasibility of such concepts can be proven, both from a theoretical standpoint and with respect to actual user interactions in command and control situations.

RECOMMENDATIONS

1. Modify LADDER to incorporate user profile and mission profile concepts described herein.

2. Conduct experiments with operational users to examine the effects of these concepts on performance of the query system.

REFERENCES

1. NOSC Code 8321 ltr rpt, Performance of a Natural Language Query System in a Simulated Command Control Environment, by HG Miller, RL Hershman, and RT Kelly, May 1978.

2. REQUEST: A Natural Language Question-Answering System, by WJ Plath; IBM Journal of Research and Development, vol 20, July 1976, p 326–335.

3. An English Language Question Answering System for a Large Relational Database, by DL Waltz; Communications of the ACM, vol 21 no 7, July 1978, p 526–539.

4. BBN Report 2378, The Lunar Sciences Natural Language Information System: Final Report, by WA Woods, RA Kaplan, and B Nash-Webber; Bolt, Beranek and Newman, Inc, Cambridge MA, 1972.

5. Final Technical Report, Project 4763, Mechanical Intelligence: Research and Applications, by ED Sacerdoti (ed); SRI International, Menlo Park CA, December 1977.

6. NOSC Code 8321 ltr rpt, Advanced Command and Control Architectural Testbed Concept of Operations Plan, Volume 1, July 1977.

7. Datacomputer Version 5 User Manual; Computer Corporation of America, Cambridge MA, July 1978.

8. The Datacomputer – A Network Data Utility, by T Marill and D Stern; AFIPS Conference Proceedings, vol 44, May 1975, p 389–395.

9. INTERLISP Reference Manual, by W Teitelman; Xerox Palo Alto Research Center, Palo Alto CA, December 1975.

10. NAVCOSSACT Document 07A001A UM-04, Navy WWMCCS Software Standardization: Book 3 Query Module User Manual; Naval Command Systems Support Activity, May 1976.

11. NOSC Code 8321 ltr rpt, The Relational Model for the Bluefile Data Base (Revised), November 1976.

12. NOSC Code 8321 ltr rpt, Expanded Relational Models for the Fleet Command Center and At-Sea Commander's Databases, March 1977.

NATURAL LANGUAGE BIBLIOGRAPHY

This bibliography lists documentation not included under REFERENCES that concerns studies over the past 14 years on the use of natural language for querying data bases.

Conrad FP, BROWSER: A User-Oriented Information Retrieval System; MS Thesis, Dept of Computer Science, University of Illinois, October 1976.

Gabriel RP, and Waltz DL, Natural Language Based Information Retrieval; Proceedings of the 12th Allerton Conference on Circuit and Systems Theory, University of Illinois, October 1974, p 875–884.

Guiliano VE, In Defense of Natural Language; Proceedings of the ACM Annual Conference, New York, 1972, p 1074.

Harris LR, SIGART Newsletter 61, ROBOT: A High Performance Natural Language Processor for Data Base Query; ACM, New York, February 1977, p 39–40.

Harris, LR, User Oriented Data Base Query with the ROBOT Natural Language Query System; Proceedings of the 3rd International Conference on Very Large Data Bases, Tokyo, Japan, October 1977.

Hendrix GG, Human Engineering for Applied Natural Language Processing; Proceedings of the 5th International Joint Conference on Artificial Intelligence, Cambridge MA, August 1977, p 183–191.

Hendrix GG, Technical Note 138, *The LIFER Manual: A Guide for Building* Practical Natural Language Interfaces; SRI Artificial Intelligence Center, Menlo Park CA, February 1977.

Malhotra A, Technical Report TR-146, Knowledge-Based English Language Systems for Management Support: An Experimental Analysis; Project MAC, MIT, Cambridge MA, February 1975.

Montgomery CA, Is Natural Language an Unnatural Query Language?; Proceedings of the ACM Annual Conference, New York, 1972, p 1075.

Mylopoulos J, Borgida A, Cohen P, Roussopoulos N, Tsotsos J, and Wong H, TORUS – A Natural Language Understanding System for Data Management; Advanced Papers of the 4th International Joint Conference on Artificial Intelligence, Tbilisi USSR, September 1975, p 414–421. (Available through MIT Artificial Intelligence Lab, Cambridge MA)

Petrick SR, On Natural Language Based Computer Systems; IBM Journal of Research and Development, vol 20, July 1976, p 314–325.

Petrick SR, Research Report 4457, Semantic Interpretation in the REQUEST System; IBM Thomas J Watson Research Center, Yorktown Heights NY, 1973.

Plath WJ, Research Report 4947, String Transformations in the REQUEST System; IBM Thomas J Watson Research Center, Yorktown Heights NY, 1974.

Schank RC, Identification of Conceptualizations Underlying Natural Language; Computer Models of Thought and Language, RC Schank and KM Colby, ed; Wott, Freeman, San Francisco CA, 1973, p 187–247.

Simmons RF, Answering English Questions by Computer: A Survey; Communications of the ACM, vol 8 no 53, 1965.

Simmons RF, Natural Lanaguage Question-Answer Systems: 1969; Communications of the ACM, vol 13 no 15, 1970.

System Development Corporation TM-WD-5711/006/00, Natural Language Interface: Final Technical Report for Phase I, 30 October 1976.

Thompson FB and Thompson BH, Practical Natural Language Processing: The REL System Prototype; Advances in Computers, M Rubinoff and MC Yovits, ed; Academic Press, New York, 1975, p 109-168.

Waltz DL, Natural Language Access to a Large Database: An Engineering Approach; Advanced Papers of the 4th International Joint Conference on Artificial Intelligence, Tbilisi USSR, September 1975, p 868-872. (Available through MIT Artificial Intelligence Lab, Cambridge MA)

Wilks Y, A Preferential Pattern-Seeking Semantics for Natural Language Inference; Artificial Intelligence, vol 6 no 1, spring 1975, p 53-74.

Winograd T, Understanding Natural Language; Academic Press, New York, 1972.

Woods WA, Transition Network Grammars for Natural Language Analysis; Communications of the ACM, vol 13 no 10, October 1970, p 591-606.

APPENDIX A: MODIFICATIONS FOR DEMONSTRATING FEASIBILITY OF ADAPTIVE GRAMMAR CONCEPTS

This appendix shows the modifications made to the LADDER grammar to demonstrate the feasibility of the adaptive grammar concepts. No attempt is made to describe precisely how the modifications accomplish the desired effects, since this would require substantial discussion of the organizational structure of the grammar as well as the processing logic performed by LADDER. For details of the mechanics of these operations, see LADDER documentation* or contact NOSC/ACCAT analysts familiar with these procedures.

---

*Hendrix GG, Technical Note 138, the LIFER Manual: A Guide for Building Practical Natural Language Interfaces; SRI Artificial Intelligence Center, Menlo Park CA, February 1977.

Final Technical Report, Project 4763, Mechanical Intelligence: Research and Applications, by ED Sacerdoti (ed); SRI International, Menlo Park CA, December 1977.

PROCEDURES

```
(LISTQUOTE
  (LAMBDA (L)
    (ENQUOTE (SUBSTRING (MKSTRING L)
                        2 -2)))

(TAIL.END
  (LAMBDA (X)
    (SETQ X LIFER.REMAINING.INPUT)
    (SETQ LIFER.REMAINING.INPUT NIL)
    (SETQ LIFER.MULTI-WORD X)))

(CHECKWITHUSER
  (LAMBDA (X)
    (PROG (Y)
          (SETQ Y (AMB.RESOLUTION X))
          (RETURN (WANT.EG (CDR Y)
                           (CAR Y)))

(AMB.RESOLUTION
  (LAMBDA (X)
    (SELECTQ (ASKUSER 30 (QUOTE N)
                     (CONCAT (MKATOM (SUBSTRING X 2 -2))

" IS NOT RECOGNISED AS A VALID NAME.
     IS IT INTENDED TO BE:
          A.  SHIP
          B.  PORT
          C.  AIRFIELD
          D.  AIRCRAFT
          E.  AIR TRACK
          F.  SQUADRON
          G.  FLIGHT
          H.  SURFACE TRACK
          I.  SENSOR
          J.  COMMUNICATION DEVICE
          K.  WEAPON
          L.  WEATHER AREA
          N.  NONE OF THE ABOVE
(ENTER THE LETTER CORRESPONDING TO THE DESIRED INTERPRETATION OF "
                              (MKATOM (SUBSTRING X 2 -2))
                    "
  ENTER AN 'N' IF NONE OF THE ABOVE APPLY.)
      ")
```

```
                    (QUOTE ((A ".   SHIP")
                            (B ".   PORT")
                            (C ".   AIRFIELD")
                            (D ".   AIRCRAFT")
                            (E ".   AIR TRACK")
                            (F ".   SQUADRON")
                            (G ".   FLIGHT")
                            (H ".   SURFACE TRACK")
                            (I ".   SENSOR")
                            (J ".   COMMUNICATION DEVICE")
                            (K ".   WEAPON")
                            (L ".   WEATHER AREA")
                            (N "ONE OF THESE."]
               (A (SPCLSETUP X (QUOTE NAM)
                               (QUOTE <UNQ.NAME>)
                               (QUOTE <NAME>)))
               (B (SPCLSETUP X (QUOTE PORP)
                               (QUOTE <UNQ.PORT>)
                               (QUOTE <PORT1>)))
               (C (FIELDSETUP X (QUOTE FLDNAME)
                               (QUOTE <AFIELDS>)))
               (D (FIELDSETUP X (QUOTE AIRNAME)
                               (QUOTE <AIRCRAFT.NAMES>)))
               (E (FIELDSETUP X (QUOTE ATRACK)
                               (QUOTE <AIRTRACKS>)))
               (F (FIELDSETUP X (QUOTE AIRUIC)
                               (QUOTE <SQUADIDS>)))
               (G (FIELDSETUP X (QUOTE FLTNO)
                               (QUOTE <FLIGHTNO>)))
               (H (FIELDSETUP X (QUOTE TRACKNO)
                               (QUOTE <TRACKHIST>)))
               (I (FIELDSETUP X (QUOTE SENSNAME)
                               (QUOTE <SENSORNAMES>)))
               (J (FIELDSETUP X (QUOTE COMMNAME)
                               (QUOTE <COMMNAMES>)))
               (K (FIELDSETUP X (QUOTE WEPSNAME)
                               (QUOTE <WPNNAMES>)))
               (L (FIELDSETUP X (QUOTE WEXAREA)
                               (QUOTE <WAREAS>)))
               (PROG NIL
                     (PRINT "PLEASE TRY THE QUERY AGAIN..." NIL)
                     (RETFROM (QUOTE PARSE)
                              NIL T])

(FIELDSETUP
  [LAMBDA (NAME FLD SYMB)
    (PROG1 (CONS (LISTQUOTE NAME)
                 FLD)
           (COND
             ((EQP (COUNT NAME)
                   1)
              (MS SYMB NAME))
             (T (FP NAME (MKATOM (SUBSTRING NAME 2 -2))
                    SYMB])
```

```
(USERSPECIFIED
  [LAMBDA (SYMB X)
    (SELECTQ (ASKUSER 30 (QUOTE N)
                      (CONCAT "THE SPECIFIED NAME,  "
                              (SUBSTRING (MKSTRING X)
                                         2 -2)

                    "  , IS NOT CONTAINED IN THE SET OF NAMES
    "
                              [PROG NIL
                                    (COND
                                      ((NULL SYMB)
                                        (RETURN "FOR THIS OBJECT"))
                                      (T (RETURN (MKSTRING SYMB]


".
IF YOU WANT THE SPECIFIED NAME PUT INTO THIS SET FOR LATER USE, ENTER A Y
(FOR 'YES').  OTHERWISE, ENTER AN N (FOR 'NO').
")
                         NIL)
               (Y X)
               (PROG NIL
                     (PRINT "PLEASE TRY THE QUERY AGAIN..." NIL)
                     (RETFROM (QUOTE PARSE)
                              NIL T])



(SINGLE/MULTIPLE.WORD.NAME
  [LAMBDA (X SING MULT)
    (PROGN [COND
             ((EQP (COUNT X)
                   1)
               (USERSPECIFIED SING X)
               (MS SING X))
             (T (USERSPECIFIED MULT X)
                (COND
                  ((OR (EQUAL MULT (QUOTE <NAME>))
                       (EQUAL MULT (QUOTE <POPT1>)))
                    (FP X (LISTQUOTE X)
                        MULT))
                  (T (FP X (MKATOM (SUBSTRING X 2 -2))
                           MULT]
           (LISTQUOTE X])

(F0201
  [LAMBDA NIL
    (WANT.EG (QUOTE TRACKNO)
             (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <TRACKHIST>)
                                        (QUOTE <TRACKHIST>])

(F0202
  [LAMBDA NIL
    (WANT.EG (QUOTE ATRACK)
             (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <AIRTRACKS>)
                                        (QUOTE <AIRTRACKS>])
```

PRODUCTIONS

```
E<ACFT> (((<AIRCRAFT.NAMES>)
     (LISTQUOTE (LIST <AIRCRAFT.NAMES>)))
    ((<AIRCRAFT> <AIRCRAFT.NAMES>)
     (LISTQUOTE (LIST <AIRCRAFT.NAMES>)))
    ((<AIRCRAFT> <INDEFINITE>)
     (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE
                                              <AIRCRAFT.NAMES>)
                                (QUOTE <AIRCRAFT.NAMES>)]

E<AIRFIELD> (((<FIELDSPEC> <AFIELDS>)
     (LISTQUOTE (LIST <AFIELDS>)))
    ((<FIELDSPEC> <INDEFINITE>)
     (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <AFIELDS>)
                                (QUOTE <AFIELDS>)))
    ((<AFIELDS>)
     (LISTQUOTE (LIST <AFIELDS>)]

E<COMM.DEVICE> (((<COMMSPEC> <COMMNAMES>)
     (LISTQUOTE (LIST <COMMNAMES>)))
    ((<COMMSPEC> <INDEFINITE>)
     (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <COMMNAMES>)
                                (QUOTE COMMNAMES)))
    ((<COMMNAMES>)
     (LISTQUOTE (LIST <COMMNAMES>]

E<FLIGHT> (((<FLIGHT.NO> <FLIGHTNO>)
     (LISTQUOTE (LIST <FLIGHTNO>)))
    ((<FLIGHT.NO> <INDEFINITE>)
     (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <FLIGHTNO>)
                                (QUOTE <FLIGHTNO>)))
    ((<FLIGHTNO>)
     (LISTQUOTE (LIST <FLIGHTNO>]

(<PORTPNTR> (((<PORTSPEC> <PORT1>)
     <PORT1>)
    ((<PORTSPEC> <INDEFINITE>)
     (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <UNQ.PORT>)
                                (QUOTE <PORT1>)))
    ((<PORT1>)
     <PORT1>)))

 E<SENSOR> (((<SENSORSPEC> <SENSORNAMES>)
      (LISTQUOTE (LIST <SENSORNAMES>)))
     ((<SENSORSPEC> <INDEFINITE>)
      (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <SENSORNAMES>)
                                 (QUOTE <SENSORNAMES>)))
     ((<SENSORNAMES>)
      (LISTQUOTE (LIST <SENSORNAMES>]
```

```
(<SHIPPNTR> (((<SHIPSPEC> <NAME>)
    <NAME>)
   ((<SHIPSPEC> <INDEFINITE>)
    (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <UNQ.NAME>)
                               (QUOTE <NAME>)))
   ((<NAME>)
    <NAME>)))

[<SQUADRON> (((<SQUADIDS>)
    (LISTQUOTE (LIST <SQUADIDS>)))
   ((<SQDRN> <SQUADIDS>)
    (LISTQUOTE (LIST <SQUADIDS>)))
   ((<SQDRN> <INDEFINITE>)
    (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <SQUADIDS>)
                               (QUOTE <SQUADIDS>)]

[<TRACK.NUM> ([(<SURFTRACK> <TRACKHIST>)
    (WANT.EQ (QUOTE TRACKNO)
             (LISTQUOTE (LIST <TRACKHIST>]
   ((<SURFTRACK> <INDEFINITE>)
    (F0201))
   [(<AIRTRK> <AIRTRACKS>)
    (WANT.EQ (QUOTE ATRACK)
             (LISTQUOTE (LIST <AIRTRACKS>]
   ((<AIRTRK> <INDEFINITE>)
    (F0202))
   [(<TRACKHIST>)
    (WANT.EQ (QUOTE TRACKNO)
             (LISTQUOTE (LIST <TRACKHIST>]
   ((<AIRTRACKS>)
    (WANT.EQ (QUOTE ATRACK)
             (LISTQUOTE (LIST <AIRTRACKS>]

[<WEATHERAREA> (((<WEXSPEC> <WAREAS>)
    (LISTQUOTE (LIST <WAREAS>)))
   ((<WAREAS>)
    (LISTQUOTE (LIST <WAREAS>)))
   ((<WEXSPEC> <INDEFINITE>)
    (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <WAREAS>)
                               (QUOTE <WAREAS>]

[<WPN> (((<WPNSPEC> <WPNNAMES>)
    (LISTQUOTE (LIST <WPNNAMES>)))
   ((<WPNSPEC> <INDEFINITE>)
    (SINGLE/MULTIPLE.WORD.NAME <INDEFINITE> (QUOTE <WPNNAMES>)
                               (QUOTE <WPNNAMES>)))
   ((<WPNNAMES>)
    (LISTQUOTE (LIST <WPNNAMES>]
(<ZZ1> (((<WHAT> <BE> <ATTRIBUTE> <OF> <INDEF.NAME>)
    (NCONC (MAPIN ? <ATTRIBUTE>)
           <INDEF.NAME>))
   ((PRINTQUERY <ZZ1>)
    LIFER.TOP.GRAMMAR)))
```

```
[<INDEF.NAME> (((<DET> <INDEF.NAME>)
    <INDEF.NAME>)
   ((<ACFT>)
    (WANT.EQ (QUOTE AIRNAME)
             <ACFT>))
   ((<SQUADRON>)
    (WANT.EQ (QUOTE AIRUIC)
             <SQUADRON>))
   ((<FLIGHT>)
    (WANT.EQ (QUOTE FLTNO)
             <FLIGHT>))
   ((<SENSOR>)
    (WANT.EQ (QUOTE SENSNAME)
             <SENSOR>))
   ((<WPN>)
    (WANT.EQ (QUOTE WEPSNAME)
             <WPN>))
   ((<COMM.DEVICE>)
    (WANT.EQ (QUOTE COMMNAME)
             <COMM.DEVICE>))
   ((<SHIPPNTR>)
    (WANT.EQ (QUOTE NAM)
             <SHIPPNTR>))
   ((<PORTPNTR>)
    (WANT.EQ (QUOTE PDEP)
             <PORTPNTR>))
   ((<AIRFIELD>)
    (WANT.EQ (QUOTE FLDNAME)
             <AIRFIELD>))
   ((<WEATHERAREA>)
    (WANT.EQ (QUOTE WEXAREA)
             <WEATHERAREA>))
   ((<TRACK.NUM>)
    <TRACK.NUM>)
   ((<INDEFINITE>)
    (CHECKWITHUSER <INDEFINITE>]
```

FIXED PHRASES

```
(<AIRCRAFT> ((AIR CRAFT))
  ((AIR PLANE)))

(<AIRTRK> ((AIR TRACK)))

(<COMMSPEC> ((COMM DEVICE))
  ((COMMUNICATIONS DEVICE)))

(<FIELDSPEC> ((AIR FIELD))
  ((AIR BASE)))

(<PORTSPEC> ((PORT OF CALL))
  ((PORT OF))
  ((HARBOR OF)))

(<SQDRN> ((SQUADRON NUMBER))
  ((SQUADRON #)))

(<SURFTRACK> ((SURFACE TRACK)))

(<WEXSPEC> ((WEATHER AREA)))

(<WPNSPEC> ((WEAPON TYPE)))
```

SETS

```
(<AIRCRAFT> (AIRCRAFT AIRPLANE))
(<COMMSPEC> (COMMUNICATIONS RADIO RECEIVER TRANSMITTER DEVICE))
(<FIELDSPEC> (AIRBASE AIRFIELD BASE FIELD))
(<FLIGHT.NO> (FLIGHT))
(<PORTSPEC> (HARBOR PORT PORT-OF-CALL))
(<SENSORSPEC> (SENSOR))
(<SHIPSPEC> (PLATFORM SHIP VESSEL))
(<SQDRN> (SQUADRON))
(<SURFTRACK> (TRACK))
(<WEXSPEC> (AREA WEATHER))
(<WPNSPEC> (BOMB MISSILE ORDNANCE ROCKET WEAPON))
```

PREDICATES

```
(<INDEFINITE>  TAIL.END)
```