JETS: ACHIEVING COMPLETENESS THROUGH COVERAGE AND CLOSURE

Tim Finin, Bradley Goodman and Harry Tennant

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Work in progress on JETS, the successor to PLANES, is described. JETS is a natural language question answering system that is intended to interface users to a large relational data base. The architecture is designed to extend the conceptual coverage of JETS to better meet the conversational and data base usage requirements of users. The implementation of JETS is designed to gain a high degree of closure over concept manipulation, contributing to a solution to the problems of perspicuity and scale. Specific examples are given of concept manipulation through the implied relationships of modification and of an approach to problem-solving through the use of frames.

JETS is a natural language question answering system that is currently under development. Its goals and design are an outgrowth of the experience gained in the development and user environment testing of PLANES [13, 14]. This paper will describe JETS in terms of motivation, performance goals, strategies for achieving the goals, and techniques for measuring performance.

The PLANES system was developed to study the problems of natural language access to a very large data base. The data base which was used contained naval aircraft maintenance and flight records, and is the same data base that JETS will be applied to. The thrust of PLANES was toward the investigation of natural language data base interfaces that could be engineered to comply with response time and machine size constraints while enabling the user to express questions in the language to which he is accustomed. PLANES accepts user utterances that may or may not comply with standard grammar, that may include pronominal reference, and that may include some forms of ellipsis. In an effort to make an objective assessment of the achievements of PLANES, testing was conducted using the techniques described in [12]. The testing revealed that in addition to problems of linguistic completeness, there are interesting questions about the conceptual completeness of question answering systems. JETS is the response to those questions.

1. Coverage and Completeness

The performance of question answering systems can be thought of as having two nearly independent components: the elements of conceptual coverage and the elements of linguistic coverage. The conceptual coverage of a system refers to the set of concepts that it can deal with. The linguistic coverage of a system refers to the set of linguistic features that have been included in it to allow for variations in the way the concepts are referenced. Syntactic structure, anaphoric reference and ellipsis are all examples of elements of linguistic coverage. Conceptual coverage and linguistic coverage are attributes of the program. The adequacy of the conceptual and linguistic coverage may be measured against the demands that a set of users place on the system. The measures are called conceptual and linguistic completeness, respectively. The attributes, conceptual and linguistic coverage, and the measurements, conceptual and linguistic completeness, are described in more detail in [12].

The major work on PLANES, as on most question answering systems, has been to provide the user with a habitable [15] system through high linguistic completeness. Our testing shows that linguistic completeness is not the only limiting factor in building habitable natural language systems. The users wanted to be able to refer to certain concepts, but were unable to do so. For example, users often asked questions about the structure of the data base or asked for definitions. They made statements to set the context for later questions, or attempted to summarize a portion of the dialogue expecting agreement or disagreement with their summaries. They made presuppositions about the data that were not Justified. In short, there were a significant number of concepts that users attempted to reference or capabilities that they attempted to use that were not included in PLANES.

In addition to the voids in conceptual coverage that were discovered while testing PLANfiS, it was observed that the users readily adapted to the limitations of PLANES[1] linguistic coverage. The user's "intellectual overhead" or "distraction from the problem" was not

This work is supported by the Office of Naval Research under Contract N0O014-75-C-O612.

measurable, but they did appear to be able to express themselves. We feel that improving the linguistic completeness of question answerers is indeed an area where attention should continue to be focused. However, linguistic completeness is only part of the problem of building habitable natural language systems. The work on JETS is centered on improving the conceptual completeness of natural language question answering systems.

With this motivation, our work on JETS has four goals. First, we intend to enlarge the domain of discourse. Instead of simply enabling a question answerer to accept questions about a data base, it must also be able to handle references to objects defined in the discourse. This includes characterizing the responses to queries that it generates to resolve references to its own utterances by the users. It should be able to handle references like "the last plane I mentioned[1]* and "the last question**. It should be able to clarify for the user how it interprets vague, ambiguous, or technical phrases.

Second, the natural language component should take on greater responsibility for pragmatic knowledge. It should have two distinct components. One component contains knowledge about the data in the data base, specifically including relationships between data types and between individual data elements. The other component contains knowledge about the activities that are reported on in the data base, rather than just the data in the data base. We have seen users refer to concepts that are related to the activities, but that are not in the data base. A natural language system should be able to understand these references if only to be more helpful in explaining why the user*s question cannot be answered. The following responses could be generated by JETS only if references to concepts outside the range of the data base were understood:

1 There is no data on individual pilots in the data base.

2 Aircraft are not grouped by squadron in the data base, but they can be grouped by permanent unit assignments.

3 The data base does not describe all combat aircraft, only fighter and attack aircraft.

4 The data base does not specify the part that failed, only the system or subsystem that contained the failure.

The third goal for JETS is to break away from the familiar model for question answerers of interpreting each user utterance as a data base query. We have observed that users do not intend eaoh utterance as a query. Some utterances set the oontext for later discourse or give general Instructions such as data formatting. Some users take several sentences to specify one query, while others couch several queries in one sentence. JETS builds an internal description of the user*s utterance, and takes action on it depending on the user demands that are implied in the utterance.

Oar fourth goal is to evaluate the performance of JETS with respect to its conceptual completeness. Designing and testing explicitly for linguistic completeness Is a goal for future research. Our conceptual completeness tests will consist of giving short descriptions of the domain of discourse to subjects, each acting in two roles. First, acting as composers, they will generate data base problems. Then, acting as users, they will try to solve problems generated by the other subjects. By having subjects who are only slightly familiar with the data base compose problems, the problems will not be biased toward those that can actually be solved using the data base. This is intended to simulate the data base problems that would occur to a casual user, one who is not very familiar with the exact contents of the data base.

The users will interact with JETS through a human interpreter, whose task will be to maintain the conceptual content of the user utterances while making them comprehendable to JETS. In other words, the interpreter compensates for potential deficiencies in linguistic coverage. The conceptual completeness of the system can be measured by computing the fraction of user utterances that were properly handled by JETS.

2. Closure

The architecture of JETS centers on establishing adequate conceptual coverage of the data base and its contents, the activities that the data base describes, and the dialogue between the user and system. The implementation of JETS, on the other hand, centers on capturing a degree of closure in the manipulation of concepts. Closure, as defined by Woods [16], implies handling user utterances which are consistent with the domain, but which may not have been specifically foreseen. Closure can be approached by basing interpretation on rules that are as general as possible.

The knowledge of JETS is organized into a network of frames. There are many kinds of links associating frames with one another, but two of the most useful are the generality/specificity links. With these links, the conceptual system can be thought of as a directed tree of frames rooted at the most general concept. The more specific frames, those closer to the leaves of the tree, inherit the information stored with their ancestor frames and add more specific information to that. The fundamental advantage of the abstraction hierarchy is that the classification of specific concepts into more general ones promotes the identification of regularities among the concepts. The hierarchy encourages the use of interpretation rules that are written to apply to the most general frames possible, and then automatically apply to their more specific descendents. The conceptual component of JETS will be implemented in FRL [4], a language for building frame based systems.

Use of the frames In JETS will be made primarily by a set of interpretation rules. These rules are represented as frames and currently have three major slots: a pattern, an action and a condition.

Implementing the interpretation component as a set of rules facilitates the insertion and deletion of rules. This means that incremental changes to the rule set may be made without necessitating the understanding of elaborate interpretation procedures.

The rules are organized hierarchically by generality, the more general rules being closer to the root. The hierarchical organization of rules is useful in resolving problems of contention. When more that one rule qualifies to be applied in a given situation, and one rule is a descendent of another, the more specific rule is chosen. In addition, the hierarchical organization of rules helps identify regularities among the rules. Regularities among the interpretation rules promote the writing of more general rules, thus contributing to gaining closure. Thus, we see several points which should prepare JETS to handle the problem of scale, one of the most important problems of natural laaguage systems: the ready ability to make incremental changes to interpretation rules, the concept clustering advantages of frames, and the identification of regularities among both the concept frames and the interpretation rules.

3. Achieving Semantic Closure

A part of our work is centered on achieving closure in the area of semantic interpretation. Our focus is an attempt to adequately cover basic noun modification, in particular, the interpretation of noun-noun modification [3J.

The problem of interpreting strings of nouns related through modification is a complex one. For our purposes, we divide the problem into three subprobleras: lexical interpretation, modifier parsing and concept modification.

By lexical Interpretation we mean the process of mapping the lexical items (in this case the nouns in the string) into appropriate concepts. The principal difficulty here is handling words with multiple senses.

Modifier parsing is the process of discovering the internal structure associated with the string of nouns or the concepts which result after lexical interpretation. For example, a string of three nouns, N1 N2 N3, might have the structure ((N1 N2) N3) or the structure (N1 (N2 N3)). The first structure would be chosen for the string "engine damage reports" and the second for the string "replacement oil pump".

The term conceptual modification refers to the problem of assigning an interpretation to an instance of one concept modifying another. For example, when the ENGINE concept modifies the DAMAGE concept in the phrase "engine damage" we want to fill the damaged objects role in the DAMAGE concept with the ENGINE concept. A more complex example is the interpretation of the phrase "engine housing acid damage". Here, the desired result is something like:

> A RESULT of a DAMAGE event in which the damaged object is a HOUSING part of an ENGINE and the causes is a CORROSION event in which the corrosive agent is an ACID and the corroded object is the HOUSING part of an ENGINE.

These three subproblems are, of course, interrelated and cannot be completely decoupled. In our initial research we are concentrating on the problem of Conceptual modification. Our goal is that, given any two concepts that are correctly interpreted by the system, their combination (i.e. through modification) will be correctly interpreted by the system. Note that the correct interpretation of a concept does not imply that the system should be able to "handle" the concept in the sense of answering questions about the concept or relating it to the data base.

The problem of interpreting noun-noun modification brings the issue of closure into focus. The essential feature of noun-noun modification is that the semantic relationship which exists between the two nouns is not explicit in the utterance. Moreover, a large number of relationships may, in principle, be possible between the two concepts represented by the nouns. It is the responsibility of the system to attempt to infer or discover an appropriate relationship, given its understanding of the two concepts involved, general pragmatic knowledge, and the current discourse context.

3.1 An example: Time

As an example, consider the use of a time phrase used to modify a noun, as in the phrases "January Skyhawks repairs" and "1976 flights". If the system can interpret phrases referring to time (as almost any system must) then it should attempt to interpret the modification of any other concept which could conceivably have a time phrase attached to it.

In our semantics, a time phrase can only be used to modify a concept which is, or can be viewed as, a kind of an EVENT. A minimal amount of closure is achieved when any event or event related concept can be successfully modified by a TIME concept. What if the modified concept is not an EVENT but something else, say an OBJECT? If a time phrase is hypothesized to modify something which is a kind of OBJECT, our system will attempt to derive an underlying event associated with that object to attach the time phrase to. For example, in the standard PARTS-SUPPLIERS-PROJECTS domain, the phrase "January parts" might suggest the interpretations: "parts which were snipped in January", "parts which were received in January", or "parts which were ordered in January". In such an impoverished domain this is almost trivial, as one can precompute the set of events in which a concept can partake.

In a semantically rich domain, such as our 3-M data base [7], the problem is much more difficult. One can not (or perhaps should not) always enumerate the potential relationships which might exist between even two simple concepts. The ability to handle references to entities and relations mentioned earlier in the discourse makes the problem even more complex. This allows for more potential relationships between any two concepts. For example, the phrase "the January planes" could be used to

refer to a set of planes introduced previously in the discourse. The successful interpretation of this phrase would require a search through the recent discourse to discover a set of planes which was involved in an event which occurred in January.

3.2 An examples: Sets

We have introduced our JETS system to the concept of a SET. In order to achieve a high degree of semantic closure, the system should be able to form the ooncept of a SET over a wide domain of objects. Our previous system, PLANES, handled sets in an unsatisfactory way. One could refer to sets of objects of certain types but not others. For example, PLANES could understand descriptions of and references to a set of aircraft or maintenance codes but it could not handle sets of parts or "how malfunctioned" codes. Such shortcomings are particularly bad in that they mislead users. If a user was successful in using a description of a set of objects which PLANES understood, he could quite reasonably infer that PLANES understood the general ooncept of a set and could form one of arbitrary objects.

Given that sets can be represented and formed in a uniform way over the widest possible domain, we must turn our attention to issues of interpreting the modification of the set concept. The ability to form sets of arbitrary elements will be of limited use if the semantic interpretation rules do not allow one to modify such sets in a general way. Thus, if the system knows what it means for concept X to modify concept Y, then it should know what it means for concept X to modify a concept Z where Z is a set whose members are concepts which are Y's.

Oar approach is to include a meta-rule for sets which uses rules applicable to the particular domain of a set. Our SET frame has a slot for a typical members
as well as one to receive the actual members, if there are any. This slot refers to a frame which describes the typical member of the set. Whenever we wish to modify a set by another concept, this meta-rule will search for primitive rules which Interpret modification of the set's typical ,members by that ooncept. The rules which are found to be applicable are then Invoked on the set's typical member and to the set's individual members, if any exist. This meta-rule for sets is shown in figure [1].

If a concept C modifies a set S then:

[1] Find a modification rule R which interprets the modification of S's typical member by C.

[2] Instantiate a new set S2.

[3] Fill S2's typical member with the result of applying rule R to S's typical member and the concept C.

[4] Fill S2's members with the result of applying R to each of S's members and C.

[5] Return the set S2.

An Interpretation Rule for Sets

figure [1]

Let's examine the use of this meta-rule for sets in the interpretation of the phrase "planes 3, 5, and 48". At the concept level, this phrase is represented as the general PLANE concept modifying a SET concept. This particular SET has the INTEGER concept for its typical member and, as individual members, the concepts for the integers 3, 5 and 48.

One of the rules applicable when a PLANE modifies an INTEGER, is a rule which interprets the integer as representing the plane's serial number. In our world, the only planes which have serial numbers are those in the data base. These planes are represented by the more specific concept 3M-PLANE. The action of this rule is to view (*) the plane concept as a 3M-PLANE and the integer concept as a SERIAL NUMBER. The 3M-PLANE's serial number slot is then filled with the SERIAL-NUMBER concept.

When this rule is found by the meta-rule for sets, it is applied to both the set's typical member filler (in this case the generic INTEGER concept) and to the set's members (which are the integers 3, 5 and 48). The result of all this is:

A SET with typical member = a 3M-PLANE
members =
(a 3M-PLANE with
serial number = an INTEGER with value = 3
a 3M-PLANE with serial number ...
a 3M-PLANE with serial number ...)

To handle the case where a SET is used to modify another concept, we include another meta-rule, shown in figure [2]. Consider the role of this rule in the interpretation of the phrase: "radar and navigation equipment failures".

If a set S modifies a concept C then:

[1] Find a modification rule R which interprets the modification of C by S's typical member.

[2] Instantiate a new set S2.

[3] Fill S2's typical member slot by invoking rule R on the typical member of S and the concept C.

[4] Fill S2's members by invoking R on each of S's members and C.

[5] Return the set S2.

An Interpretation Rule for Sets

figure [2]

This phrase is interpreted as a SET whose typical member is a FUNCTIONAL-SUBSYSTEM and whose members are the concepts RADAR-SUBSYSTEM and NAVIGATION-SUBSYSTEM. In interpreting the

* Viewing a concept X as a concept Y is a process which maps the information in X into a newly instantiated Y concept. If X is a kind of Y, no mapping need be done, however.

modification of the FAILURE concept by this SET, the meta-rule for sets is invoked and attempts to find rules which guide the interpretation of a FUNCTIONAL-SUBSYSTEM modifying a FAILURE. The rule which is most applicable is one which Interprets the modifying concept (the subsystem) as filling the failure location role of the FAILURE concept. The final interpretation of this phrase results in a SET of FAILURES in which the typical member is a FAILURE in a FUNCTIONAL-SUBSYSTEM and which contains two members: a FAILURE in the RADAR-SUBSYSTEM and a FAILURE in the NAVIGATI0N-SUBSYSTEM.

The discussion so far has centered around the need for achieving closure and possible ways of accomplishing it in JETS. However, nothing has been done to help the system develop a plan to extract the required information from the data base in the form the user intended. This is where problem-solving frames are introduced. Problem-solving frames are to be used to describe problem domains and search strategies. They can range from very general sketches of a particular series of problem environments to specific suggestions on how to answer a certain request. The latter problem-solving frame might even be encoded in English—where a sequence of English instructions are given on how to put all of the information together at the end to answer the main request. Problem-solving frames will have to work Jointly with the frames used during the parsing and semantic analysis stages. Those frames describe the objects and events of the JETS' environment and how they are related to the data in the data base. Information gathered by those frames can be used to extract appropriate values to fill the empty slots in the problem-solving frames in order that the whole frame may become instantiated.

4.1 Well -Defined Problems

Before describing the details of problem-solving frames, a description of what a "problem" is should be discussed. A well-defined problem has been defined in the literature to consist of the following properties:

1 expressed in terms the solver can understand,

2 all information necessary for solving the problem should be in the problem statement,

3 the form of the solution is exactly specified, and

4 there must be a systematic (algorithmic) way(s) for testing a proposed solution [8].

In our natural language query system environment (in particular with our experience with PLANES) we find that many requests by users are not normally well-defined, i.e. they do not satisfy the criteria above. Before such requests can be handled, it is necessary to make the request a well-defined problem. Previous systems have brought in this additional knowledge primarily through two sources—using past context to fill in missing information and/or asking the user directly for it [2,14]. We want to introduce the use of world knowledge as a third way of obtaining such information. At the same time we want to use the world knowledge to detect requests not answerable with data available in the data base.

Collecting together such information still does little for bringing the system closer to developing a plan for answering the request. The problem-solving frames are to propose general techniques—such as problem reduction (reducing a problem to simpler subproblems)—for solving problems. To be able to develop a set of problem-solving frames that classify problems and techniques for solving them, we must categorize the "kinds" of problems encountered in our data base environment (based on the assumption that the data base world restricts us enough that the number of interesting classes of problems is manageable). Our studies have shown that the following kinds of requests occur most often: statistical analyses (correlation of data, etc.), categorization (classifying data into categories), association of data types to each other, ranking ("top five", etc.), plotting, causality (generalize concept), very general inferencing and operations on sets.

Another major contribution would be the ability to provide different "views" to a problem and to the values stored in the data base. In essence an "overlay" of the data base could be generated for a particular problem. An example would be data stored in a daily format in the data base viewed as if weekly data existed. This same mechanism could be applied to bring two seemingly disjoint concepts in a problem together.

4.3 A. Scenario

A sample scenario of the use of problem-solving frames can be seen in Figure [3]. They are activated by key words and phrases in the user's request and as side-effects to the instantiation of particular frames during the semantic analysis stage of JETS. The activated frames analyze the request to decide if they can provide any guidance in formulating a plan capable of rendering a solution to the problem. If no such assistance can be offered, the frame is deactivated; otherwise complete control is passed to that problem-solving frame. Figure [4] gives an example of a problem-solving frame for comparison.(*) Notice the use of production-like rules that associate specific conditions and actions. These actions include drawing in other problem-solving frames and invoking specific functions such as MAKE-UNITS-EQUAL which tries to convert the units of measurement (time, length, etc.) of the individual fields into equivalent forms.

In summary, the value of this type of frame is that it provides a general forum for taking all the information provided by the frames instantiated

* The frames in our system are actually being written in FRL but the examples in this paper were written in a more simplified form for readability.

```
U: Compare NORS and NORMUS percentages by squadron
   and by wing for May 73.

J: Invoking COMPARE frame.  Invoking PERCENT frame.

   NOR (Not Operationally Ready) not in the data
   base.  It can be obtained by ADDING NORS (NOR due
   to Scheduled  Maintenance) and NORMUS (NOR due to
   Unscheduled Maintenance).

   SQUADRON  partitions the existing planes up.  No
   such data is stored in the data base.  Such
   partitioning can be done by:
      (1) plane serial number
      (2) plane type
      (3) none of the above
   Please select one of the above:

U: 2

J: WING linked to SQUADRON. PLANE TYPE absorbing
   WING.

   I have interpreted your request as follows:

     1 Retrieve NORS, NORMUS by plane types for time
       period of May 1973.

     2 Form NOR=NORS+NORMUS.

     3 Generate NORS%=NORS/NOR and
       NORMUS%=NORMUS/NOR.

     4 Construct table of PLANE TYPE, NORS
       percentage, NORMUS percentage.

     5 List by PLANE TYPE the maximum of NORS
       percentage or NORMUS percentage.

   Does this satisfy your request?

U: yes

J: Executing...

      PLANE TYPE   NORS%   NORMUS%
          A7         71%     29%
          F4         55%     45%
      SKYHAWKS       27%     73%

          A7: more NORS
          F4: more NOR
          SKYHAWKS: more NORMUS

                    A Scenario

                    figure [3]
```

```
Invoking Concept:
  COMPARE, MORE, LESS, CORRELATE, ANALYZE, ...

Context(s):
  Requires instantiation of two or more entities
  to compare:
    (1) one FIELD over two or more ranges, or
    (2) two or more different FIELDs:  FIELD1,
                  FIELD2,...

Actions:
 Context=(1):
  if FIELD.TYPE=num & CONCEPT=more
     then invoke greater;
  if FIELD.TYPE=num & CONCEPT=less
     then invoke less;
                .
                .
                .
 Context=(2):
  if FIELD1.TYPE=num & FIELD2.TYPE=num
                     & CONCEPT=more
     then invoke greater;
  if FIELD1.TYPE=num & FIELD2.TYPE=num
                     & CONCEPT=less
     then invoke less;
  if FIELD1.TYPE=num & FIELD2.TYPE=set
                     & CONCEPT=more
     then invoke cardinality-of-set
          invoke more;
  if FIELD1.TYPE=num & FIELD2.TYPE=num
                     & CONCEPT=compare
     then invoke find-relationship;
  if CONCEPT=associate
     then invoke check-correlation
          invoke find-relationship;
                .
                .
                .
Global Actions:  make-units-equal

                    COMPARE Frame

                    Figure [4]
```

command is either the sort of question which can be understood directly, or another command which can ultimately be broken up into a sequence of questions that can be answered. They will also be used to generate dialogue with the user when vague terms must be further explained before a formal query could be generated (e.g. defining "worst" for the particular user and question).

during parsing and semantic analysis and analyzing it so that a program can be generated to answer the question. In other words, it is not responsible for writing a program at the formal query level but only to decide what question to answer, to break the question up into a sequence of simpler requests if the question is too complex, to provide higher level mathematical/statistical analysis of the data returned, and to call on a formal query generator to generate the actual formal query.

4.4 Handling vague. and Complex Requests

The other type of problem-solving frame not yet described will be used in analyzing vague and complex questions. Each frame will consist, in essence, of a sequence of simpler commands, where each simpler

In JETS, this kind of problem-solving frame is useful for report generation. The following describes what a problem-solving frame for requests like "Does trend analysis of failure and maintenance rates differ significantly from the corresponding rates of new aircraft?" would have to do.(*)

1 define "differ significantly" (user-defined or system default),

2 retrieve for each maintenance action X whether it was scheduled or unscheduled and time since last maintenance on same system,

• Taken from list of questions most asked by 3-M Naval personnel[7].

3 form maintenance rate for each maintenance action,

4 project trend of maintenance rates by linear regression,

5 calculate average mean time between failure,

6 apply (1) through (5) to new aircraft,

7 generate table of trend of failure rate and maintenance rate for all data vs. trend of failure rate and maintenance rate for new aircraft, and

3 compute standard deviations for the trend of failure rates and maintenance rates for all aircraft and for Just new aircraft.

The problem-solving frame writes all of the program required except the generation of the basic formal queries (the ones that retrieve the actual fields used to calculate the failure and maintenance rates) which are generated by the formal query generator [13].

5. Conclusion

This paper has discussed the goals and design of the JETS natural language question answering system. Our work on JETS is strongly motivated by our experience with the earlier PLANES system. The evaluation of PLANES highlighted its problems and shortcomings. A major goal of the work on JETS is to increase the completeness of the conceptual coverage of the system. The implementation is expected to begin during the summer of 1979. JETS will then be tested to measure linguistic and conceptual completeness. We feel that including evaluation as an integral part of the project is Important for three reasons. First, it will help us to keep issues of closure in focus. Second, it will create a detailed record of performance, making JETS' capabilities and limitations explicit. Third, the record of evaluation will provide a basis of comparison for assessing the performance of otner natural language systems.

REFERENCES

[1] Bobrow, R. J. and J. S. Brown, "SYSTEMATIC UNDERSTANDING: Synthesis, Analysis, and Contingent Knowledge in Specialized Understanding Systems", in D. G. Bobrow and A. Collins, Representation and Understanding, Academic Press, New York, 1975, pp. 103-129.

[2] Codd, E.F., R.S. Arnold, J-M Cadiou, C.L. Cnang and N. Roussopoulos, RENDEZVOUS Version 1: an Experimental English Language Query Formulation System for Casual Users of Relational Data Bases, IBM Report RJ2144, San Jose, 1978.

[3] Finin, T., "The Semantic Interpretation of Noun-Noun Modification", Working Paper 21, Advanced Automation Group, Coordinated Science Lab, University of Illinois, 1979.

[4] Goldstein, I. P. and R. B. Roberts, "The FRL Manual", MIT AI Memo 409, 1977.

[5] Hemphill, Linda and James Rnyne, "Models for Knowledge Bases in Natural Language Query Systems", (submitted for publication).

[6] McDermott, D., "Planning and Acting", Cognitive Science, vol. 2, no. 2, pp. 71-109(1978).

[7] NALDA (Naval Air Logistics Data Analysis) System Data Requirements Determination Report, Naval Aviation Integrated Support Center, Patuxent River, Maryland, 1975.

[8] Newell, A. and H. Simon, Human Problem Solving, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, pp. 72-75.

[9] Raphael, Bertram, THE THINKING COMPUTER Mind Inside Matter, W.H. Freeman & Co., San Francisco, 1976, pp. 146-150.

[10] Schank, R. C. and R. P. Abelson, Goals, Plans, Scripts and Understanding: an Enquiry into Human Knowledge Structures, Erlbaum Press, N.J., 1977.

[11] Tennant, H., "The PLANES Database", Working Paper 14, Advanced Automation Group, Coordinated Science Lab, University of Illinois, 1978.

[12] Tennant, H., "Experience with the Evaluation of Natural Language Question Answerers", Proc. IJCAI-79, 1979.

[13] Waltz, D. L. and B. A. Goodman, "Writing a Natural Language Data Base System", Proc. IJCAI-77, 1977, pp. 144-150.

[14] Waltz, D. L., "An English Language Question Answering System for a Large Relational Data Base", CACM, vol. 21, July, 1978, pp. 526-539.

[15] Watt, W. C., "Habitability", American Documentation, July, 1968, pp. 338-351.

[16] Woods, W. A. "A Personal View of Natural Language Understanding", SIGART Newsletter, February, 1977, pp. 17-20.