UAH RESEARCH REPORT NO. 751

A NATURAL LANGUAGE INTERFACE TO DATABASES

FINAL REPORT

Prepared by:

D. R. Ford
Johnson Research Center
The University of Alabama in Huntsville
Huntsville, AL 35899

Prepared for:

John Wolfsberger
System Software Branch
Information and Electronic Systems Lab
George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

October 1988

ABSTRACT

This paper presents the development of a Natural Language Interface (NLI) which is semantic-based and uses Conceptual Dependency representation. The system was developed using Lisp and currently runs on a Symbolics Lisp machine. A key point is that the parser handles morphological analysis, which expands its capabilities of understanding more words.

1.0 Introduction:

Natural languages are the languages used by people in the course of their daily affairs, for example, English, French, Japanese, etc. Natural languages are used to express a broad range of ideas to others. Given enough attention, nearly any concept that comes to mind can be conveyed to another person through a common natural language. Some concepts are easy to express, such as, "I am hungry," whereas others may require lengthy explanations. The prime characteristic of natural languages is that they can be used to express nearly all the concepts that occur to the people who speak and understand them.

The word *natural* emphasizes a contrast with artificial languages. Artificial languages are those that have been designed to be highly expressive over a limited range of ideas. Musical notation is an artificial language. Another set of artificial languages is programming languages. These are interesting because, like natural languages, they can be used to express a broad range of concepts. LISP, for instance, is an extendable language, that is, if an idea is difficult to express in its current form, it can be improved at will. But programming languages have been designed with their application to computers in mind, and this has affected their form. Programming languages have been written so as to be analyzed easily by computers.

1.1 Motivation:

Research in natural language understanding is concerned with making computers capable of using natural languages. There are two reasons for this. First, computers that can use natural languages would undeniably be a useful tool. It would mean that a person in need of information retrieval or information processing on a computer could obtain it without having to learn a computer language or go through an intermediary. They would not have to worry about becoming fluent in a "foreign" language and maintaining that fluency just to accomplish their jobs. A computer that could use natural languages could read normal text, providing users with access to computer-generated summaries or reports synthesized from reading several text sources.

The second motivation for natural language research is that it will increase our understanding of how human languages and minds work. To develop the technology for a computer to use language, we must first be able to say specifically what language is. We must be able to say precisely how the concepts we wish to express can be represented in the computer. Building computer programs requires this precision and attention to detail. A programming implementation of a theory of language can be used to identify flaws, inconsistencies, and areas of incompleteness that may go unnoticed.

1.2 Research Strategies:

Two research strategies are in use for conducting natural language understanding research: (1) the isolated phenomena strategy and (2) the entire system strategy. The first strategy tests specialized theories of restricted language phenomena, for example, analysis of syntactic structure of sentences or identifying the referents for pronouns. Programs are written to show that the theory can make some contribution to language understanding in the proper context. One criticism of the isolated phenomena strategy is that in focusing on a subproblem of natural language understanding, other subproblems must be assumed solvable by some other methods. Sometimes the other problems are difficult to solve. The danger is that attention may unwittingly be focused on the easy or less crucial problems, leaving the difficult ones unattended.

Another criticism of the isolated phenomena strategy is that success is difficult to measure. Evaluation mus rely primarily on plausibility arguments. Without demonstrative evaluation, deficiencies and inconsistencies in a large piece of work may easily go undetected.

The second research strategy designs and builds entire natural language understanding programs. This strategy is closer to applied research than to pure research. For a given allotment of research resources, less can be applied to particular subproblems than with the isolated phenomena strategy. It may also become mired in implementing many mundane programs, as well as, numerous interesting ones. However, the entire system strategy is protected

against assuming away the major problems or concentrating on nonessentials. Because the entire system strategy requires some attention to the complete range of language analysis, major problems receive attention first. Another advantage of this strategy is that it lends itself to impartial evaluation. Because an entire system must be built to "do something" -- whether answering questions, generating summaries, or conducting fact searches -- some way must be found to test whether it can do it and to measure how well it is done.

1.3 Methodology:

The methodology of natural language understanding research is straight forward: develop a theory and program it, then study the behavior of the resulting program for flaws in the theory. The cycle starts again with a revised or altogether new theory and a corresponding program.

It sounds much easier than it is. One is not entirely free in forming his or her theories, because the theories will have to be programmed. The human brain is much more complex than any computer in existence, so computer implementations must, at best, be partial models of how the human mind works. The programs written to implement theories of language and cognition tend to be as large as the computer technology will readily allow. As a result, the programming arc of the cycle may require several man-years of effort. Finally, it is not clear how we should study the behavior of

the resulting programs, to characterize their achievements and understand their flaws.

2.0 SYSTEM DEFINITION:

Most major computer users are currently interested in natural language modules which fit on top of other coputer software. Given the current interest in expert systems, a natural language "front end" seems a very promising application of natural language understanding. However, the computer software best suited to benefit from a natural language front end is the database. Typically databases hold huge quantities of data which have to be stored in complex ways so as to secure the fastest access for the maximum proportion of queries. Many formal query languages have been designed to simplify the problem of getting the correct data from the monoliths.

These formal languages can be divided into "one-dimensional" and "two-dimensional" langauages. One-dimensional languages are composed of letters, numbers, and mathematical signs, while two-dimensional languages are more mnemonic in design. While there are shortcomings to both types of formal languages, many of these can be overcome by extending them and giving more intelligence to the interpreter.

By doing less with syntax and more with word meaning, the language can reduce the necessity for the user to conform to an artificial syntax. If the formal words have a meaning close to their natural meaning, the burden on the user is further reduced,

especially if all alternative natural language words with the same meaning are allowable synonyms in the formal language. A final refinement would be to make the remaining syntax correspond to the syntax of natural language.

A second improvement would be to reduce the requirement for the user to know about the details of the database. Natural language synonyms for all database names are an obvious start, but also the facility for the interpreter to navigate around the database would be a great advantage. The ability to use the structure of the data as well as the structure of the query to aid the interpretation of the user's input is a necessary feature.

A third improvement that would be possible with an intelligent interpreter is to recognize the user's intended query on the basis of incomplete or slightly erroneous input. Thus simple queries could be recognized in a very abbreviated form. However, more complex queries would need more complete expression.

Another improvement would be to enable the user to phrase his queries incrementally. Thus his original query might be ambiguous and the system could prompt him to select which interpretation was intended. Or else the could phrase a simple query and follow up with supplemental queries.

Finally, an intelligent interpreter could perhaps recognize logical inconsistencies in a query and warn the user, or answer a broader but consistent query. Also, curtness and breadth of use are two final desiralbe features.

Clearly all of these are capabilities which natural language front ends are aiming to provide. In particular, natural language is

powerful and precise where required. However, it is curt and it does provide a conversational interface which enables the user to ask incremental queries. The term "natural language" may be a drawback because it suggests capabilities beyond any current or foreseeable implementation. It tempts the user to ask common sense questions outside the area of the database's body of information, or evaluative questions within its area of information.

The term natural language front end is used for a good reason, however. It implies that the user should not consciously have to translate his queries into terms appropriate for the front end. Language users do unconsciously adapt their way of speaking to their audience, however, and the natural language subset accepted by the front end should be dense enough for the user to adapt without conscious effort.

As long as the user does not have a vastly over-optimistic expectation of what a natural language front end can do for him, it can enable him to get the right answers with less frustration, and little requirement for technical support.

3.0 SYSTEM DESIGN:

The above system definition was used for the system requirements. The basic design of the intellegent interpreter was taken from the tasks specified in the system definition. The front end should consist of: (1) the parser, (2) the formal query generator, and (3) the database access routines. Figure 1 contains a diagram of the front end.

This research effort focused on developing the parser for the front end. In order to accomplish the system requirements, it was decided to use a semantic-based parser along the lines of those developed by the Natural Language Group at Yale Univeristy. A particularly robust parser was developed by Michael Dyer while at Yale and this was chosen as the basic model for this research effort.

The parser consists of a dictionary of words, expressions, and the expectations associated with both; a program that performs the conversion from words and/or expressions to CD representation; and a monitoring program that keeps track of which expectations have been generated and performs the actions associated with an expectation when the conditions of the expectation are fulfilled. These components combined are often referred to as a parser.

The parser is intelligent about a limited subset of English and can extract the intended meaning from sentences. A dictionary is used by the parser as a data file and provides the CD representation for the words and expressions.

Many modifications and extensions have been made to the McDYPAR-based parser used in this research. There is an expressions feature which allows the definition of expressions in the lexicon. An expression is defined as a phrase of two or more words that has a different meaning when parsed together than if the words are parsed separately. Morphological analysis, which allows suffixes and/or prefixes to be removed recursively from the word in an attempt to identify the root of the word, has also been added. This allows only the root forms of regular verbs, nouns, etc. to be defined in the dictionary with any associated meaning-changing

suffixes and prefixes. Morphological analysis is performed on both words and expressions.

Another extension involves identifying pronouns and their referents. This is accomplished in the usual manner, saving the most recent instance of many types of nouns and replacing the pronoun with the referent's definition. This has proven to be a fairly reliable method, no doubt partly due to the limited, description-oriented simulation domain.

Other modifications to the parser include handling numbers, both floating point and integer, as units of meaning and recognizing sentence voice and other grammatical structures as concepts within a sentence. In this study, minor modifications have been made to the CD primitive set in order to customize the parsing environment to the particular domain. The modifications include adding new key-word identifiers to definitions, and creating new items that could be used to fill slots in the conceptual dependency representation. Also, the code for some expectations was modified to allow physical objects to perform certain actions.

3.1 CONCEPTUAL DEPENDENCY:

For a machine to be able to understand a user's natural language, a theory must exist to allow the transfer of the process to machines. Such a theory does exist and was developed by Dr. Roger Schank at Yale University. His theory is called Conceptual Dependency (CD). The essence of CD is that underlying all natural languages is a representational scheme for concepts, and that the

purpose of natural language is to communicate concepts. Thus, in order to understand natural language, the symbols used to convey concepts must first be converted into an underlying representational scheme and structures. This scheme is called CD representation.

CD theory was always intended to be a theory of how humans process natural language that was explicit enough to allow for programming it on a computer. An underlying principle of CD is that meaning representation must consist of concepts and relations between concepts. Also, there should be restrictions as to what qualifies as either. The syntax of the conceptual level consists of all the rules for constructing relationships between concepts on this level.

The basic components of CD are : (1) conceptual analysis, (2) expectations, (3) primitive acts, and (4) conceptualization. Conceptual analysis is the process of converting the natural language input into CD representation. The analysis of a sentence consists of two activities: (1) adding expectations to a list and (2) checking the expectation list to determine if any have been satisfied. Whenever a word or expression is analyzed, it is checked for any associated expectations. If any are attached to the word, they are added to the expectation list and the list is checked to see if any expectations have been satisfied. This process is repeated for each word and/or expression in a sentence.

The analysis of a sentence is driven by the execution of these expectations. Also, the expectations utilize the conceptual rules of CD. This is the knowledge provided by the syntax of the conceptual level which describes how conceptual categories can be combined.

Expectations are the basic mechanisms of a comprehension. An expectation is a description of a situation that is recognized as possibly becoming true in the future. Associated with any expectation is a list of actions to be performed when the expected situation comes into existence. Thus, an expectation consists of conditions that must be satisfied and the actions to be performed when the conditions are satisfied. This is a general kind of world knowledge that allows appropriate response to a situation as soon as it is recognized. The importance of expectations is that they prepare sets of actions for use, if needed, and they narrow the perception of future situations. Thus, expectations are used to predict what concepts and words may occur and eliminate most of the ambiguity involved in language processing.

Primitive acts are the basic elements into which all verbs are divided. Also, they are the basic structure used to construct a conceptualization. Schank has defined eleven primitive acts. These involve actions that can be applied to objects by actors. An act refers only to what an actor has actually done and is treated separately from the possible consequences of the action.

A conceptualization is the basic unit of the conceptual level out of which thoughts are constructed. It is the resulting CD representation of a sentence plus any inferences. A conceptualization can be composed of an actor, act, object, recipient, direction, and state.

CD representation allows the computer to process a natural-language description in sentence form in order to extract the meaning expressed by the user. This meaning is represented as

concepts. From the concepts, translation to other languages, both human and computer, is possible. Thus, the computer is capable of performing the first task performed by the analyst (i.e., understanding the natural-language description of the model) identified earlier.

3.2 AN EXAMPLE:

In order to test the functionality of the parser, a test domain of simulation was chosen. This was based on our familiarity with the simulation world and the inability to gain access to NASA personnel with database expertise. No blame is being placed. Efforts were made, but due to time constraints and availability we were unable to make the connections. Thus the following example will be used to illustrate the capability of the parser.

The parser takes as input a written description of the process that the user desires to simulate. This description is in sentence form in the user's own words. A sample input is found in Table 3. This description is a simplified version of a printed circuit board moving through one machine in a manufacturing facility. The acronyms, DIP and VCD, signify different types of machines in the manufacturing process and what they stand for is not relevant to the problem discussion at this point. The description in Table 3 is typed directly into the computer. If any mistakes are made in typing, the user can backup and correct them. However, once the inputting process is completed, no corrections can be made by the user. To let the parser know that the input is complete and ready to be

processed, the user types "Done." on a separate line. Once the parser recognizes this word, it begins separating the sentences and the words to determine the meaning of each.

The words and expressions contained in Table 4 are representative of those contained in the PARSER. For every word and expression there are two elements: (1) the definition and (2) the expectations. The definition of a word or expression is found after the key-word "def", and is a CD structure or part of a structure. For

TABLE 3

AN EXAMPLE INPUT TO THE PARSER

> The printed circuit boards arrive at the DIP machine according to a uniform distribution with a minimum of 8 seconds and a maximum of 12 seconds. The printed circuit boards are processed at the DIP machine for 3 seconds. The printed circuit boards then proceed to the VCD machine.
> Done.

example, the word ARRIVE has a CD structure for a definition and the expression PRINTED CIRCUIT BOARD has a part of a CD structure for its definition.

Values of the components of the CD structure that are used for definitions can be a specific item or computer code to determine what the value should be. Look at the definition for ARRIVE in Table 4 . The value for the ACTOR component is NIL or unknown. This is a specific value. However, the OBJECT component contains computer code to locate the value that should be associated with this

component. This code is given what to search for and where in the sentence to search.

The other element, expectations, of a word or expression is located after the key-word "demons". These expectations are the name or names of various computer code needed to complete the definition and/or CD representation. For instance, the expression PRINTED CIRCUIT BOARD contains two expectations: (1) save-object and (2) how-many. The first expectation needs no additional information to perform its task, but the second one needs the word "quantity", and a variable called "suffix20" and the normal ending to form the plural of "printed circuit board".

The parser takes the first sentence from the input and looks for any expressions that may be contained therein. An example of an expression in this case is PRINTED CIRCUIT BOARD. The conceptual dependency representation is removed from the expression and stored in the working area of the parser. Also, any expectations are placed on a list. Symbols and pointers are used to keep track of which expression and/or word goes with which sentence.

After the expressions have been identified, then the remaining elements of the sentence must be words. The parser starts at the beginning of the sentence and looks up each word and locates the associated conceptual dependency representation. An example of this can be seen in Table 4. The conceptual dependency representation for the word ARRIVE is found after the key-word "def", and includes everything after the opening parenthesis and before the second closing parenthesis following " 'after."

As the word is placed into the working area, any associated expectations are placed on the expectation list. An example is in Table 4. The expectations for the word ARRIVE are located after the word "demons" and also inside the conceptual dependency representation itself. After the words "object" and "to", an arrow (<==) appears. This indicates that these values are to be determined by the expectations that follow. This process is the same for words as it is for expressions.

After each word or expression is placed into the working area, all expectations on the expectation list are tested. If any expectation passes the test, then it is allowed to complete its action. Completing an action entails connecting various conceptual dependency representations for words and expressions to form a representation for a sentence or filling in values that are missing. For instance, the expectations attached to the "object" and "to" slots of the word ARRIVE will locate values for these slots and insert them. This process repeats itself until all the words and/or expressions in all of the sentences of the input are checked. After this, the conceptual dependency representations for each sentence are placed on a list that would be sent to the formal query generator. An example of this representation is in Table 5.

The CD representation in Table 5 is the result of the parser processing the English input in Table 3. The key word SENT-NUM is the connecting link between the input and the output of the parser. There are three sentences in the input and three CD representations generated. The first sentence in Table 3 is represented by the

TABLE 4

EXAMPLES OF WORDS AND EXPRESSIONS IN THE DICTIONARY

WORDS:

```
(ENTER    def  (ptrans actor nil
                       object * <== (exp-wrt-voice
                                      '(process-object pronoun)
                                       'before)
                       to * <== (exp-wrt-voice
                                  '(complex process-actor pronoun)
                                    'after))
          demons   ((get-sentence-number)  (determine-voice)))


(ARRIVE   def  (ptrans actor nil
                       object * <== (exp-wrt-voice  '(process-
                                      object pronoun) 'before)
                       to * <== (prep '(at) '(complex   process-actor
                                      pronoun)  'after))
          demons   ((get-sentence-number)  (determine-voice)))
```

EXPRESSIONS:

```
( (PRINTED-CIRCUIT-BOARD)
        def  (process-object name (printed-circuit-board))
        demons ((save-object) (how-many 'quantity 'suffix20 's)))


( (NORMAL DISTRIBUTION)
       def  (dist-type name (normal)
                    mean * <== (exp-statistic '(mean)  'after)
                    sd * <== (exp-statistic '(standard-deviation)
                                'after))
       demons  (ins-bef '(ptrans do) 'dist))
```

PTRANS statement in Table 5. There are six major parts to this statement: (1) Actor, (2) Object, (3) To, (4) Voice, (5) Sent-Num, and

(6) Dist. The basic structure of the PTRANS is attached to the verb ARRIVE. The Actor approximates the subject of the sentence which is unspecified in the first sentence, i.e., how the printed circuit boards arrive at the DIP machine is not given. Thus, NIL is inserted after Actor.

The Object of the PTRANS is the printed circuit board and its destination is the DIP machine. This latter information is contained in the TO slot. The distribution information is contained in the DIST slot. The distribution is uniform, thus, it must contain a minimum and a maximum value. The time units are valuable in simulation, therefore, a record of the units is maintained. The VOICE and SENT-NUM slots are self-explanatory, and are used for internal bookkeeping in the parser.

Looking at the second sentence in the English input from Table 3, the corresponding CD representation in Table 5 is the DO statement. This implies that some action is being performed that causes a physical change in the composition of an object. The actor is the DIP machine, the object is the printed circuit board, and the action is "processed." Another important attribute is the duration of the action. Again the time units are significant, as well as, the number of units.

The third sentence in Table 3 is represent by the second PTRANS statement in Table 5. Here the ACTOR is unknown and the value of the TO slot has changed to the VCD machine. The OBJECT is still the printed circuit board.

TABLE 5

AN EXAMPLE OF THE CONCEPTUAL DEPENDENCY INPUT TO THE FORMULATOR

```
(PTRANS  ACTOR  NIL
         OBJECT (PROCESS-OBJECT NAME (PRINTED-CIRCUIT-
                                      BOARD) QUANTITY (1))
         TO (PROCESS-ACTOR CLASS (STATION)
                           NAME (DIP-MACHINE) QUANTITY (1)
                           PREPOBJ (PREP IS (AT)))
          VOICE (ACTIVE)
          SENT-NUM (1)
          DIST (DIST-TYPE NAME (UNIFORM)
                          MIN (STATISTIC NAME (MIN)
                          MEASURE (TIME NAME (SECOND)
                                         BASE-UNITS (1)
                                         QUANTITY (8)))
                          MAX (STATISTIC NAME (MAX)
                          MEASURE (TIME NAME (SECOND)
                                         BASE-UNITS (1)
                                         QUANTITY (12)))))

(DO ACTOR (PROCESS-ACTOR CLASS (STATION)
                         NAME (DIP-MACHINE) QUANTITY (1)
                         PREPOBJ (PREP IS (AT)))
    OBJECT (PROCESS-OBJECT NAME (PRINTED-CIRCUIT-BOARD)
                           QUANTITY (1))
     VOICE (PASSIVE)
     SENT-NUM (2)
     DURATION (TIME NAME (SECOND)
                    BASE-UNITS (1) QUANTITY (3)))

(PTRANS ACTOR NIL
         OBJECT (PROCESS-OBJECT NAME
                                (PRINTED-CIRCUIT-BOARD)
                                QUANTITY (1))
         TO (PROCESS-ACTOR CLASS (STATION)
                           NAME (VCD-MACHINE)
                           QUANTITY (1)
                           PREPOBJ (PREP IS (TO)))
          VOICE (ACTIVE)
          SENT-NUM (3))
```

The formal query generator and the database access routines were not covered in this research effort but will be addressed in the future phases of this project.

4.0 CONCLUSIONS:

From the work that has been done with constructing a Natural Language Interface (NLI) both in the past and with this current effort, it is apparent that connecting an interface to databases is very feasible and desirable. However, because building a natural language understanding system is such a domain specific task, much work will be required to tailor the NLI to an appropriate domain of interest. Also, from the recent research we believe it is more advantageous to pursue a dual line of investigation of which type of interface will be better, either the semantic-based or the keyword. Another aspect that needs to be looked at closely is testing these interfaces with "real" humans, i.e., let the people who will use these systems test them and use other non-interested parties. This will let us see if the systems are robust enough to be of use to prospective users, and allow the system to be tailored to specific needs.

5.0 REFERENCES:

Bobrow, D. G. "Natural Language Input for a Computer Problem-Solving System." in *Semantic Information Processing,* ed. M. Minsky, Cambridge, MA: MIT Press, 1968.

Craig, J. A.; Brenner, S. C.; Carney, H. C.; and Longyear, C. R. "Deacon: Direct English Access and Control." in *Proceedings of the Fall Joint Computer Conference,* Montvale, NJ: AFIPS Press, 1966.

Green, B. F.; Wolf, A. K.; Chomsky, C.; and laughery, K. "Baseball: An Automatic Question Answerer." in *Computers and Thought,* ed. E. Feigenbaum and J. Feldman, New York: McGraw-Hill, 1963.

Harris, L. R. "User Oriented Data Base Query with the ROBOT Natural Language Query System." *International Journal of Man-Machine Studies* 9 (1977): 697-713.

Hendrix, G. G. "Human Engineering for Applied Natural Language Processing." in *Proceedings of the International Joint Conference on Artificial Intelligence,* Cambridge, MA: MIT, 1977.

Lindsay, R. K. "Inferential Memory as the Basis of Machines Which Understand Natural Language." in *Computers and Thought,* eds. E. Feigenbaum and J. Feldman, New York: McGraw-Hill, 1963.

Riesbeck, C. K. "Conceptual Analysis." in *Conceptual Information Processing,* ed. R. C. Schank, New York: American Elsevier, 1975.

Riesbeck, C. and Schank R. C. "Comprehension by Computer: Expectation-based Analysis of Sentences in Context." Research Report 78, Yale University, 1976.

Simmons, R. F. "Answering English Questions by Computer: A Survey." *CACM* 8 (January 1965): 53-70.

Simmons, R. F. "Storage and Retrieval of Aspects of Meaning in Directed Graph Structures." *CACM* 9 (March 1966): 211-215.

Simmons, R. F.; Klein, S.; and McConlogue, K. "Toward the Synthesis of Human Language Behavior." *Behavioral Science* 7 (July 1962): 402-407.

Waltz, D. L. "Natural Language Access to a Large Data Base." in *Advaance Papers of the Fourth International Joint Conference on Artificial Intelligence,* Cambridge, MA: MIT, 1977.

Waltz, D. L. and Goodman, B. A. "Writing a Natural Language Data Base System." in *Proceedings of the International Joint Conference on Artificial Intelligence,* Cambridge, MA: MIT, 1977.

Weizenbaum, J. *Computer Power and Human Reason.* San Francisco: W. H. Freeman, 1976.

Weizenbaum, J. "Eliza- A Computer Program for the Study of Natural Language Communication Between Man and Machines." *CACM* 9 (January 1966): 36-45.

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

Woods, W. A. "Procedural Semantics for Question-Answering Machine." in *Proceedings of the Fall Joint Computer Conference.* Montvale, NJ: AFIPS Press, 1973.

Woods, W. A. "Semantics for a Question Answering System." Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, 1967.

Woods, W. A.; Kaplan, R. M.; and Nash-Webber, B. *The Lunar Sciences Natural Language Information System: Final Report.* Report 2378, Cambridge, MA: Bolt, Beranek and Newman, 1972.