NATURAL-LANGUAGE PROCESSING
PART ONE: THE FIELD IN PERSPECTIVE

Technical Note 237

July 13, 1981

By: Gary G. Hendrix, Director
Natural Language Program
Artificial Intelligence Center

and

Earl D. Sacerdoti, Director
Research and Development
Machine Intelligence Corporation

To be published in BYTE Magazine, 1981.
Preparation of this paper was supported
by internal funding from SRI International
and Machine Intelligence Corporation.

ABSTRACT

This article deals with the problems of enabling computers to communicate with humans in natural languages, such as English and French, as distinguished from formal languages, such as BASIC and PASCAL. Major issues in natural-language processing are discussed by examining several experimental computer systems developed over the last decade. The intent of the authors is to demonstrate that natural-language processing techniques are useful now, to reveal the richness of the computations performed by human natural-language communicators, and to explain why the fluent use of natural language by machines remains an elusive aspiration.

CONTENTS

ILLUSTRATIONS

I INTRODUCTION

Through a process spanning thousands of years, natural languages (such as English and French) have evolved to meet the manifold needs of human beings to communicate and record a diversity of kinds of information in a wide variety of circumstances. Natural language is the linguistic medium of the butcher, the baker, and the candlestick maker; the poet and the lover; the politician and the preacher; the parent and the child. Even for the scientist and computer programmer, it is the mother tongue--the language one resorts to when formal expressions and intuitions fail. Natural languages stand in marked contrast to formal languages, such as BASIC and PASCAL, which were specifically designed to be easily understood by computers and are intended for the specialized task of expressing algorithms and data structures. The fluent use of natural language is an information-processing activity of great complexity. Endowing computers with this ability has for over twenty years been a major goal of research in artificial (or machine) intelligence, a branch of experimental computer science that studies the nature of knowledge and its manipulation.

Understanding the computational mechanisms that underlie the use of natural language is the central objective of _computational linguistics_*, a science at the juncture of artificial intelligence, philosophy, linguistics, and psychology. The two primary goals of this field are

* To understand how humans communicate.
* To create machines with human-like communication skills.

The first of these, a scientific goal, is pursued to help us understand

---

* The Association for Computational Linguistics is a professional society for people interested in this area; it publishes the _American Journal of Computational Linguistics_. For information, contact Donald Walker, SRI International AIC, Menlo Park, Ca. 94025. Readers are also referred to the American Association for Artificial Intelligence (contact Bruce Buchanan, Computer Science Department, Stanford University, Stanford, Ca. 94305), and the Cognitive Science Society, which publishes the journal _Cognitive Science_ (contact Donald Norman, Center for Human Information Processing, C-009, University of California at San Diego, La Jolla, Ca. 92093).

ourselves. In particular, although all of us are implicitly expert in the _use_ of natural language, we have only vague notions of the actual mental processes involved. A clearer insight into their essential nature and functioning might enable us to be better communicators, to train our children better in language skills, and even to design more efficient intercomputer communications.

The second goal, an engineering one, is pursued for a very practical purpose--to create machines that can communicate with people in languages, such as English, that they already know. At present, only a small segment of the population, those trained in the art of computer programming, can communicate with the computers in our midst. The advent of machines that understand natural languages should make it possible for virtually anyone to make direct, personal use of powerful computational systems.

Progress in computational linguistics is facilitated by pursuing both goals simultaneously. Creation of mechanical schemes for dealing with some aspect of natural-language processing sheds light on how it might actually be performed by the human brain. Similarly, evidence derived from the observing of how humans actually use their language suggests prospective computational mechanisms or, more often, provides valuable insights into the reasons particular mechanical processes fail.

To create computer systems that deal with certain significant subsets of natural-language phenomena, it will probably not be necessary to perform the task in a way closely simulating computational processes in the human brain. This should be no more surprising than the fact that mechanical dishwashers use a very nonhuman technique to produce a result equivalent to that of a human dishwasher. For interactions about very limited subject areas, we can hope to employ thoroughly nonhuman techniques in dealing with natural language. Nevertheless, machines concerned with any but the most mundane aspects of human language will almost certainly be forced to deal with human psychology. After all, natural language has evolved as an efficient tool for conveying information between human minds. One of the participants in a man-machine dialogue operates with all the constraints and richnesses of the human psyche; the other will inevitably have to take these into account.

The ultimate goal of creating machines that can interact in a facile manner with humans remains far off, awaiting both improved information-processing algorithms and alternative computing architectures. However, progress in the last decade has demonstrated the feasibility of dealing with natural-language input in highly restricted contexts, employing today's computers. Furthermore, microcomputer implementation of these limited language-processing techniques is leading to progressively more practical, cost-effective systems.

In this article (including Part Two, Technical Note 237) we offer an overview of the potential applications, experimental systems, existing techniques, research problems, and future prospects in this rapidly evolving field. We shall address major issues in natural-language processing by focusing on several representative systems, necessarily leaving much other important work unmentioned. For example, we shall not discuss the complex issues involved in understanding spoken (as opposed to typed) language. Our intention is to demonstrate that natural-language processing techniques developed over the last few years are useful now, to reveal the richness of the compuations performed by human natural-language communicators, and to explain why the fluent use of natural language by machines remains an elusive aspiration.

Because many BYTE readers may have an interest in building small natural-language processing systems themselves, we shall minimize our emphasis on linguistic and psychological considerations in order to devote more space to describing practical computational considerations. Part Two of this article focuses upon techniques for actually building a microcomputer system that can deal with natural language in very limited contexts.

II APPLICATIONS OF NATURAL-LANGUAGE PROCESSING

To motivate our discussion as to how one might approach the technological goal of creating a machine with human-like communication skills, let us consider some potential areas for the application of natural-language processing (NLP).

Machine Translation--Perhaps the oldest dream of computational linguistics is that of a mechanical device that can read documents written in one (natural) language and produce corresponding documents written in other languages, but with equivalent meaning. In fact, the birth of computational linguistics may be said to have occurred in 1946, when Warren Weaver and A. Donald Booth first suggested the use of a digital computer to create such a device. Even the Association for Computational Linguistics, the professional organization in this discipline, was originally named the Association for Machine Translation and Computational Linguistics.

Document Understanding--Beyond "simply" translating a document from one language to another, a device might read and understand documents, fitting the information they contain into a larger framework of knowledge. A practical device of this sort would read and assimilate a document in much the same way a human would. The device might subsequently produce abstractions of the document, alert people who are likely to be interested in it, or answer specific questions based on the information it contains. If such a device has read many documents, it might be able to act as a librarian, directing users to especially pertinent references.

Document Generation--A task related to document understanding is document generation. One may envisage a device that translates information stored in a formal language in a computer's memory into ordinary language. For example, the designer of an automobile engine might describe repair procedures in a formal language. (After all, we expect that the designing of mechanical devices will someday be done principally by computer systems, which may prefer formal languages.) From this formal description, instruction manuals could be generated mechanically in all the languages likely to be spoken by car buyers.

A more sophisticated system could generate special manuals for particular groups or individuals. Taking into account the likelihood that a mechanic will know a great deal about auto repair and neither need nor want to be told how to use a screwdriver, a smart system would generate a different manual for mechanics than for automobile owners, but on the basis of the same underlying information. More information on how to do elementary mechanical tasks would be included in manuals for less knowledgeable individuals. An ultimate system would tailor a manual to the background of each individual.

It is worth pointing out that a repair manual need not be written out in linear sequence in typical book format. Using a computer, advice about how to proceed on any particular problem could be dynamically generated to apply specifically to the task at hand. We shall return to this topic in Section III-C-3.

As Part Of A System--Perhaps the most interesting use of NLP is as part of a larger computer-based system. For example, one can imagine devices that not only communicate with users in English, but also

* Provide answers to questions by accessing large databases.
* Control such complex equipment as industrial robots, power generators, or missile systems.
* Furnish expert advice about medical problems, mechanical repairs, how to buy stocks, or what to cook for supper.
* Teach courses in a broad range of subjects.

An extreme example of a computer-based system that would use NLP as an integral component is a robot that communicates with its user in English. Such a robot might be expected to perform at least as many tasks involving the use of natural language as might be done by a human assistant.

The importance of these potential applications and the basic science needed to make them possible has long been appreciated by various scientific funding agencies within the United States government. In particular, current progress in the field is due largely to continued support over the years from the Defense Advanced Research Projects Agency, the National Science Foundation, and the Office of Naval Research.

III WHAT EXISTING SYSTEMS CAN DO

A. Systems for Interfacing to Databases

1. The Database Access Problem

One of the most important (and most feasible) areas for the application of NLP is in accessing data in databases. Billions of dollars have been spent in collecting and encoding such data. Yet this information is generally not readily available to the decision-makers who need it most. The situation is illustrated by the cartoon of Figure 1.

An executive in the widget business wants to direct a simple question to the black box his computer experts have developed for him. In particular, he would like to know, "How many widgets did we sell in August?" He is aware that the information is somewhere in the black box, but he lacks the specialized expertise needed to make the box understand him.

As shown in Figure 2, he must find an interpreter (computer programmer) who can understand his English question and translate it into a formal query to give to the machine. The problem is that programmers are always out drinking coffee when you need them, or they are working on some other project that they believe is more important than your project--so they cannot help you this week. When a programmer _is_ available, he will often misunderstand some details of the task requirements when he is first told and, once he does understand, his initial attempts at creating code will very likely produce bugs. The result is that, by the time an answer is extracted from the computer, it may no longer be timely and may not even be relevant!

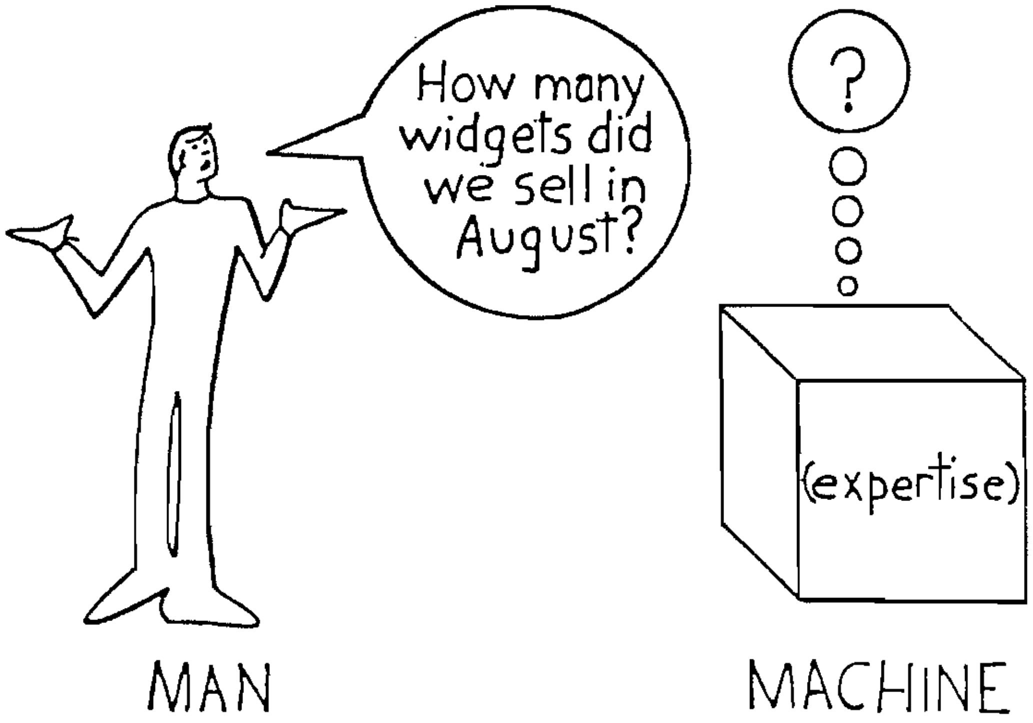
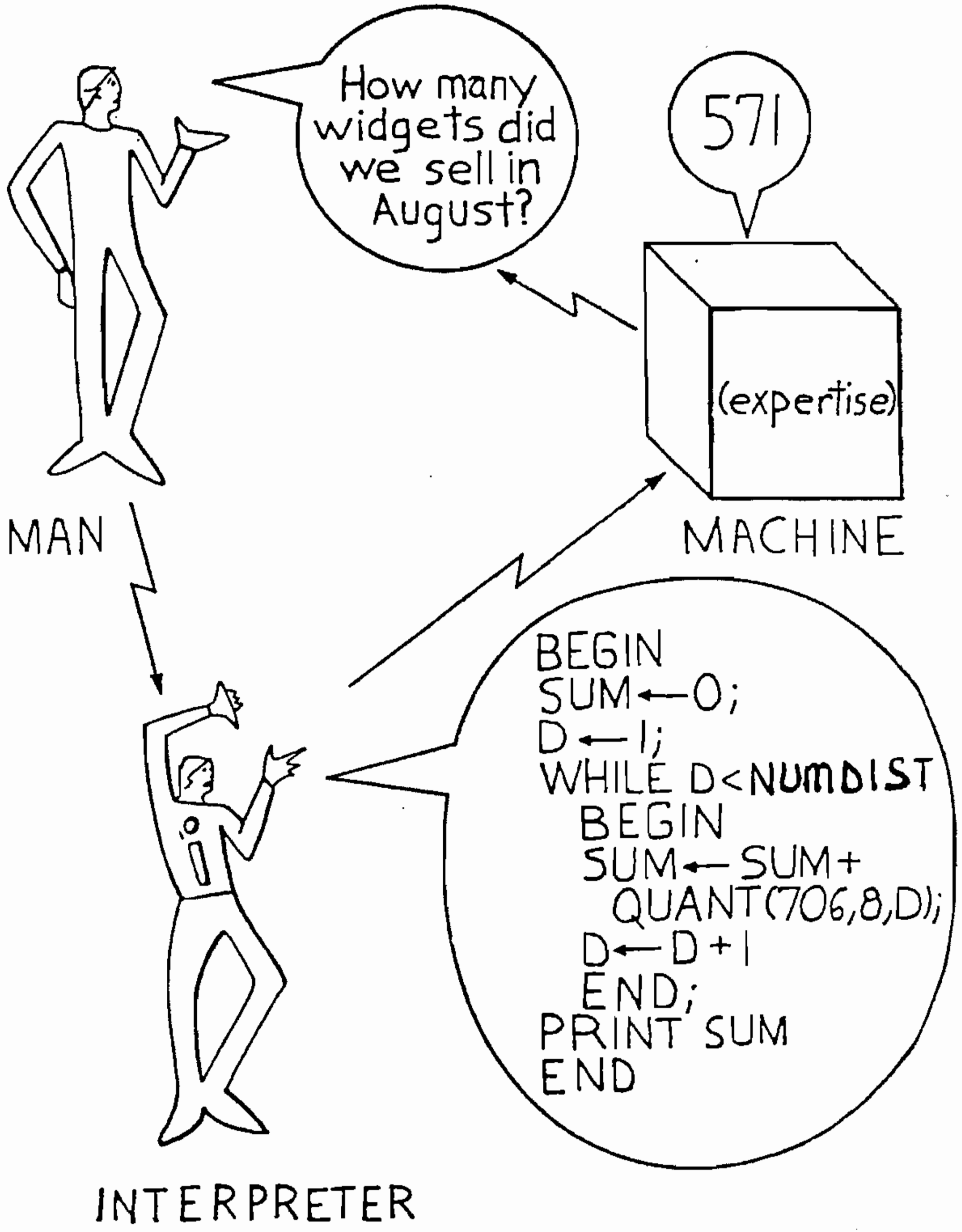


Figure 1 The Black Box



```
BEGIN
SUM←0;
D←1;
WHILE D<NUMDIST
  BEGIN
  SUM←SUM+
    QUANT(706,8,D);
  D←D+1
  END;
PRINT SUM
END
```

Figure 2 The Interpreter

2. The LADDER System

To produce timely answers to questions and quickly clear up problems as to how a decision-maker's question is to be interpreted, the turn-around time must be cut from hours or days to seconds.

Several research groups around the world are attempting to do this by automating the programmer in Figure 2. For example, the LADDER system developed at SRI International [12] [11] is capable of translating a question such as

"TO WHAT COUNTRY DOES THE FASTEST SUB BELONG?"

into either the code of Figure 3 or that of Figure 4, depending on which database management system (DBMS) has the relevant data. An explanation of these segments of code is unnecessary here. The point is that systems exist that are capable of accepting simple English queries specifying what information a user wants, then generating fairly complex programs specifying how the computer is to retrieve the information.

Note that two problems are being confronted together:

(1) The system must translate from English into a formal language.

(2) The system must convert a statement of what is wanted into a statement of how to get it.

Problem 2 is concerned with automatic programming, an artificial intelligence problem currently receiving a great deal of attention in its own right [2]. (Fortunately, most work on NLP only needs to consider a highly restricted subset of this general problem.) What it is like to use a system such as LADDER is suggested by the transcript, shown in Figure 5, of an actual interaction. The system prompts users with a transaction number followed by a hyphen. The user can then type in a question or command. Query 1, "Give me the length of the Kenedy," contains a misspelled word. LADDER corrects this misspelling automatically, typing a note to the user directly below the input line. Once LADDER has corrected the spelling error and completed an analysis of the input, it types the message "PARSED!," and then displays its

English query--

To what country does the fastest sub belong?

DATALANGUAGE Query--

```
BEGIN
DECLARE Y1 STRING (,100) ,D='}'
DECLARE Y2 STRING (,100) ,D='}' Y2 = '00.0'
DECLARE Y3 INTEGER Y3 = 0
DECLARE Y5 STRING (,100) ,D='}' Y5 = 0
DECLARE Y4 STRING (,100) ,D='}' Y4 = 0
FOR R1 IN SHIPCLASCHAR WITH (R1.TYPE2 EQ 'S') AND
                                            (R1.TYPE1 EQ 'S')
        FOR R2 IN SHIPCLASDIR WITH (R2.SHIPCLAS EQ R1.SHIPCLAS)
                FOR R3 IN SHIP WITH (R3.UICVCN EQ R2.UICVCN)
                        BEGIN
                        Y1 = R3.MCSF
                        IF Y1 LE '99.9' AND Y2 LT Y1 THEN
                                BEGIN
                                Y2 = Y1
                                Y5 = R3.NAT
                                Y4 = R3.NAM
                                Y3 = 1
                                END
                        END
IF Y3 EQ 1 THEN
        BEGIN
        NSTDPORT.STRING1 = Y4
        NSTDPORT.STRING2 = Y5
        END
END
```

Figure 3 A Query in Datalanguage

interpretation of the user's query. The system then calls a remote database to retrieve the answer, in this case that the Length of the Kennedy is 1072 (feet).

Query 2 is not a complete sentence and, in fact, makes no sense when considered in isolation. But, in the context of the preceding query, it is clear that the intended meaning is "Give me the width and draft of the Kennedy." Leaving out pieces of a sentence like this is called ellipsis. Processing such elliptical inputs is harder

English Query--

To what country does the fastest sub belong?

DBMS20 Query--

```
     COMPUTE XSTRX11 = '00.0' $
     COMPUTE XY10 = 0 $
     FIND FIRST SHIPCLASCHAR RECORD OF BLUEAREA AREA $
13   IF ERROR-STATUS = 307 GO TO 14 $
     COMPUTE XAND16 = 0 $
     IF SHIPCLASCHAR-TYPE2 NE 'S' GO TO 17 $
     COMPUTE XAND16 = 1 $
17   IF XAND16 = 0 GO TO 15 $
     COMPUTE XAND18 = 0 $
     IF SHIPCLASCHAR-TYPE1 NE 'S' GO TO 19 $
     COMPUTE XAND18 = 1 $
19   IF XAND18 = 0 GO TO 15 $
     COMPUTE XSTRZ12 = SHIPCLASCHAR-MCS $
     IF XSTRZ12 LT '00.0' OR XSTRX11 LE XSTRZ12 GO TO 15 $
     COMPUTE XSTRX11 = XSTRZ12 $
     COMPUTE XY10 = 1 $
     COMPUTE XSTR29 = SHIPCLASCHAR-MCS $
15   FIND NEXT SHIPCLASCHAR RECORD OF BLUEAREA AREA $
     GO TO 13 $
14   * $
     IF XY10 = 0 GO TO XT $
     FIND FIRST SHIPCLASCHAR RECORD OF BLUEAREA AREA $
20   IF ERROR-STATUS = 307 GO TO 21 $
     COMPUTE XAND23 = 0 $
     IF SHIPCLASCHAR-TYPE2 NE 'S' GO TO 24 $
     COMPUTE XAND23 = 1 $
24   IF XAND23 = 0 GO TO 22 $
     COMPUTE XAND25 = 0 $
     IF SHIPCLASCHAR-TYPE1 NE 'S' GO TO 26 $
     COMPUTE XAND25 = 1 $
26   IF XAND25 = 0 GO TO 22 $
     COMPUTE XAND27 = 0 $
     IF SHIPCLASCHAR-MCS NE XSTR29 GO TO 28 $
     COMPUTE XAND27 = 1 $
28   IF XAND27 = 0 GO TO 22 $
     FIND FIRST SHIPCLASDIR RECORD OF BLUEAREA AREA $
30   IF ERROR-STATUS = 307 GO TO 31 $
     COMPUTE XAND33 = 0 $
     IF SHIPCLASDIR-SHIPCLAS NE SHIPCLASCHAR-SHIPCLAS GO TO 34 $
     COMPUTE XAND33 = 1 $
34   IF XAND33 = 0 GO TO 32 $
     SET SHIP-UICVCN TO SHIPCLASDIR-UICVCN $
     FIND SHIP RECORD $
35   IF ERROR-STATUS = 326 GO TO 36 $
     PRINT SHIP-NAM SHIP-NAT $
37   FIND DUPLICATE SHIP RECORD $
     GO TO 35 $
36   * $
32   FIND NEXT SHIPCLASDIR RECORD OF BLUEAREA AREA $
     GO TO 30 $
31   * $
22   FIND NEXT SHIPCLASCHAR RECORD OF BLUEAREA AREA $
     GO TO 20 $
21   * $
     GO TO XT $

     END
```

Figure 4 A Query in DBMS20

than it may seem; the system has to avoid such interpretations as "Give me the length of the width and draft," or even "Give width and draft the length of the Kennedy."

Query 3 illustrates the use of a pronoun, another linguistic construct that cannot be interpreted in isolation.

Query 4 demonstrates a more sophisticated form of ellipsis, in which the analysis involves more than just grammatical substitution of the new input into the old one. (This is accomplished in LADDER by the simple trick of looking for "WHAT ABOUT" at the beginning of an input, and then applying the standard elliptical technique to what follows.)

Query 5 illustrates a major benefit of NLP: the user can interact with the system in the terms of the job he is doing, letting the system be concerned with what is actually required from the database. Though question 5 is one question from the user's perspective, it in fact requires two questions of the database. First, the location of the Kimlow must be determined, and then the appropriate ships within 500 miles of that location must be retrieved. The process of translating from the user's terms to those of the database is one that gets in the way of the user's decision-making job, and is therefore an appropriate one to automate.

Interaction 6 causes a large body of information to be retrieved from the database. Although omitted from the figure to save space, a table summarizing the answer is displayed to the user.

Interaction 7 shows an example of a cooperative response to the user's question. The user asks a question about the readiness of French ships. However, the database contains no information about French ships. Therefore, the literal answer to the user's question, NONE, is inappropriate by itself, since it leads the user to believe that all French ships are unready. To provide a more informative response, LADDER presents additional information showing the rationale for producing the NONE response.

```
 1- Give me the length of the Kenedy.
                    spelling-> KENNEDY
 PARSED!
 Give the length of the ship KENNEDY JF


LEN = 1072

2- width and draft
PARSED!
Trying Ellipsis: GIVE ME WIDTH AND DRAFT OF THE KENEDY
Give the beam and draft of the ship KENNEDY JF

(BEAM 130 DFT 36)

3- Who is her commander?
Give the rank and name of the commander of the ship KENNEDY JF

(RANK CAPT NAME MOFFETT P)

4- What about the Fox?
PARSED!
Trying Ellipsis on: THE FOX

  WHO IS THE FOX COMMANDER

Give the rank and name of the commander of the ship FOX

(RANK CAPT NAME EVERETT J)

5- What U.S. merchant ships carrying vanadium ore are within 500 miles
    of the Kimlow?
PARSED!

Give the position of and time at position for the ship KIMLOW T

Give the name of merchant American ship S
   where the type of cargo on S is VNAD
         the great circle distance from the position of S to
                 15-33N, 30-10W, is less than or equal to 500

SHIP = GREENVILLE VICTORY, CRAIN ME, TOTOR

6- How fast are the U.S. subs with lengths greater than 150 feet?
PARSED!
Give the maximum cruising speed for and name of American ship S
   where the first character of the type of S is S
         the second character of the type of S is S
         the length of S is greater than 150

[A long table relating the names and maximum cruising speeds
 of American submarines is printed.]

7- What French ships are at readiness status 1?
PARSED!
Give the name of French ship S
   where the state of readiness of S is 1

 There is no French ship S
 NONE
```

Figure 5 A Dialogue With LADDER

LADDER's methods for dealing with natural-language inputs are very similar to those used by compilers and interpreters for such languages as BASIC and PASCAL. For example, a BASIC interpreter might deal with assignment statements by looking for the pattern

```
LET <variable> = <expression>.
```

This pattern could match an instruction of the form

```
LET X = 5 + Y
```

with "X" filling the role of the <variable> and "5 + Y" filling the role of the <expression>. Associated with this pattern, the interpreter would have a function for storing the value of the expression into the memory location named by X.

Similarly, LADDER uses patterns such as

```
WHAT <BE> THE <SHIP-ATTRIBUTES> OF <SHIP-DESCRIPTION>
```

which can match sentences such as

```
WHAT ARE THE LENGTHS AND DRAFTS OF US CARRIERS?
```

Just like an interpreter or compiler, LADDER associates a function with each pattern. The function associated with the example just cited would produce calls to the DBMS to retrieve attributes of ships, and would take as parameters the names of the attributes and a description of the ships of interest. Most of LADDER's knowledge about language and the world is implicitly encoded in its grammar and associated functions. The grammar contains a great deal of information on the subject of the particular database being queried and is by no means a standard grammar of English. A grammar of this type is called a <u>pragmatic</u> or <u>semantic grammar</u> [4], [11].

3. <u>Summary of LADDER-like Capabilities</u>

In Section VI we shall have more to say about what systems like LADDER can do. For now, it is sufficient to point out the following:

* The computer capability shown in the transcript of Figure 5 is of considerable practical utility.

* LADDER deals with a relatively large and complex database, which includes over 100 fields in fourteen files, and has records for 40,000 ships.
* LADDER has been performing at this level of capability since 1976. (Except the cooperative responses such as those in Interaction 7 are relatively new. See [14].)
* There are several systems in laboratories around the world that are capable of essentially the same level of performance shown in Figure 5. These include [31], [24], [26], [25], [14], and [10].

4. Limitations

For restricted classes of applications, systems such as LADDER can provide language-processing capabilities that users find highly useful. Nevertheless, LADDER falls far short of being an ideal system, both conceptually and linguistically.

LADDER's concept of the world is based on the underlying conventional database management system to which it provides access. DBMSs can effectively store large numbers of individual, concrete facts such as

THE KENNEDY IS OWNED BY THE U.S.

but they are incapable of dealing in a general way with more logically complex notions, such as disjunction, quantification, implication, causality, and possibility. DBMSs act as if they were dealing with information about a world containing a fixed number of objects and relationships among them, and as if the objects and relationships were immutable.

Perhaps LADDER's most important linguistic deficiency is its limited notion of linguistic context. With minor (though useful) exceptions, LADDER treats each input as if it were given in isolation. To perceive the problem, let us consider the question

WHO ARE THE CAPTAINS OF THE US TANKERS?

Isolated from all contexts, this question should be interpreted as a request for the names of the commanding officers of all the US tankers in the database. But, if a user has just asked the question "What is the status of convoy C86?" and has been given information on a number of ships in the convoy, including two U.S. tankers, the sample question should then elicit the captains' names for only the two tankers in the convoy. LADDER ignores the context, however, answering the question as if it had been asked in isolation.

The ability to follow a changing context and make accurate references to objects salient therein is a fundamental characteristic of human communication. In fact, about half the words we use in ordinary speech are found in _definitely determined noun phrases_ (DEF NPs), the linguistic constructions most often used to refer to objects in context. Let us note, for example, all the DEF NPs underlined in the dialogue shown in Figure 6.

The need to understand context throws considerable doubt on the very idea of building natural-language interfaces to systems with knowledge bases that are independent of the language-processing system itself. This is because the information in the knowledge base may be needed simply for comprehension of a question. For example, to understand the phrase "the filter" in

> IF I CHANGE THE OIL IN MY CAR, WHERE SHOULD I LOOK
> FOR THE FILTER?

it is necessary to know that automobiles use oil cleaned by a filter. Such knowledge makes possible the assumption that such a filter, namely the one on the user's car, is the referent of "the filter." The point is that we cannot translate the question into a formal query to an auto maintenance system unless the translation system also has at least some information about the nature of auto maintenance.

B. _Systems for Dealing with Dynamic Microworlds_

1. _The SHRDLU System_

SHRDLU, a system developed by Terry Winograd at MIT around 1970 [29], was one of the first systems to deal with some of the complexities of context and to address a domain of greater logical complexity than can be handled by a conventional DBMS. An example of

A: HOW DO I REMOVE THE FLYWHEEL?

E: FIRST LOOSEN THE TWO SMALL ALLEN HEAD SETSCREWS HOLDING IT TO THE SHAFT THEN PULL IT OFF.

A: THE TWO SETSCREWS ARE LOOSE BUT I'M HAVING TROUBLE GETTING THE WHEEL OFF.

E: USE THE WHEEL PULLER. DO YOU KNOW HOW?

A: NO

E: LOOSEN THE SCREW IN THE CENTER AND PLACE THE JAWS AROUND THE HUB OF THE WHEEL; THEN TIGHTEN THE SCREW.

A: THE LITTLE METAL SEMICIRCLE FELL OFF WHEN I TOOK THE WHEEL OFF.

Figure 6 Sample From Human-Human Dialogues

the types of interactions SHRDLU could support is shown in Figure 7. At the beginning of the dialogue, both user and system are talking about a small microworld of blocks configured as shown in Figure 8.

In Interaction 1, SHRDLU is asked to pick up a big red block. There is only one such block in the microworld, so SHRDLU knows what to pick up. Complicating the situation is the fact that a green block is resting on the big red block, so SHRDLU is forced to devise a plan that involves moving the simulated robot arm to the green block, lifting the green block, moving the block to a clear place on the table, placing the green block on the table, releasing the green block, returning to the red block, grasping the red block and, finally, lifting it up. Once all this is done (after considerable computation time), the microworld configuration is as shown in Figure 9 and SHRDLU responds with a simple "OK."

In Interaction 2, the user asks SHRDLU to grasp "the pyramid." But there are two pyramids in the microworld context, and the linguistic

1 - PICK UP A BIG RED BLOCK.

OK.

2 - GRASP THE PYRAMID.

I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.

3 - FIND A BLOCK WHICH IS TALLER THAN THE ONE YOU ARE HOLDING AND PUT IT IN THE BOX.

BY "IT" I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.

OK

4 - WHAT DOES THE BOX CONTAIN?

THE BLUE PYRAMID AND THE BLUE BLOCK.

5 - WHAT IS THE PYRAMID SUPPORTED BY?

THE BOX.

6 - HOW MANY BLOCKS ARE NOT IN THE BOX?

FOUR OF THEM.

7 - WHY DID YOU CLEAR OFF THE RED BLOCK?

TO PICK IT UP.

8 - WHY DID YOU DO THAT?

BECAUSE YOU ASKED ME TO.

Figure 7 A Dialogue With Winograd's SHRDLU

context contains no clues to choose between them. SHRDLU realizes that it cannot identify the referent of "the pyramid" and tells the user so.

Interaction 3 reveals SHRDLU's ability to deal with a fairly complex noun phrase containing a relative clause and a comparative construction, and to handle problematic words such as "one" and "it." The result of this exchange is the microworld configuration shown in Figure 10. Note that the system has found it necessary to put down the big red block it had been holding.

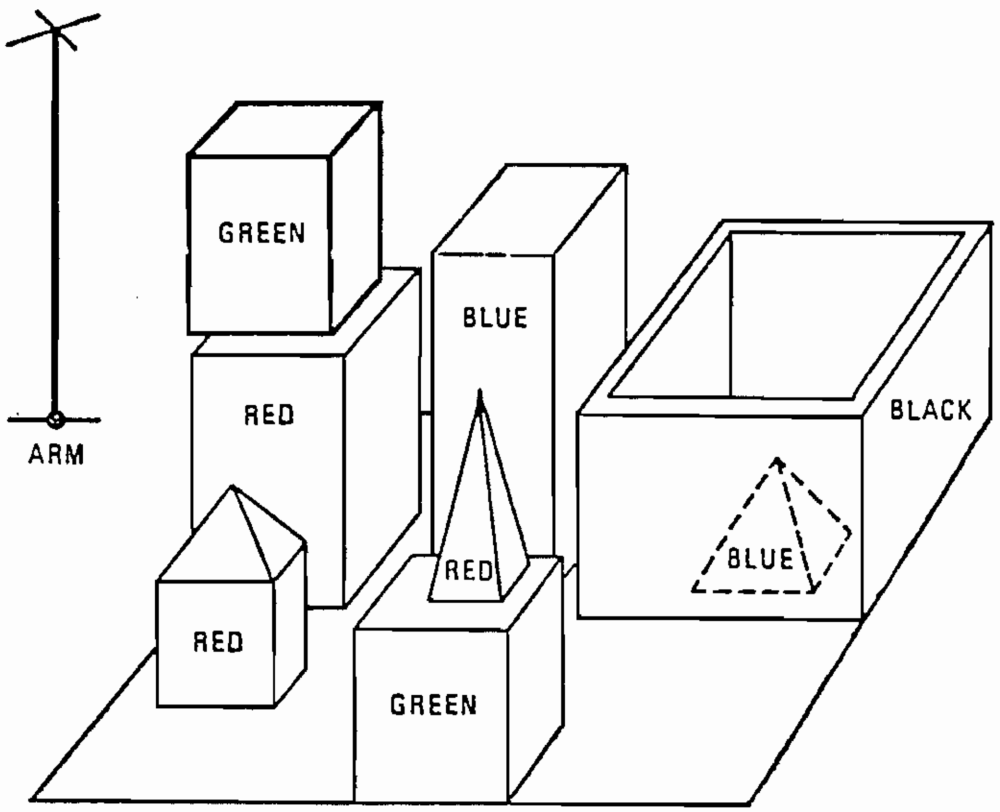


Figure 8 Initial State of Microworld

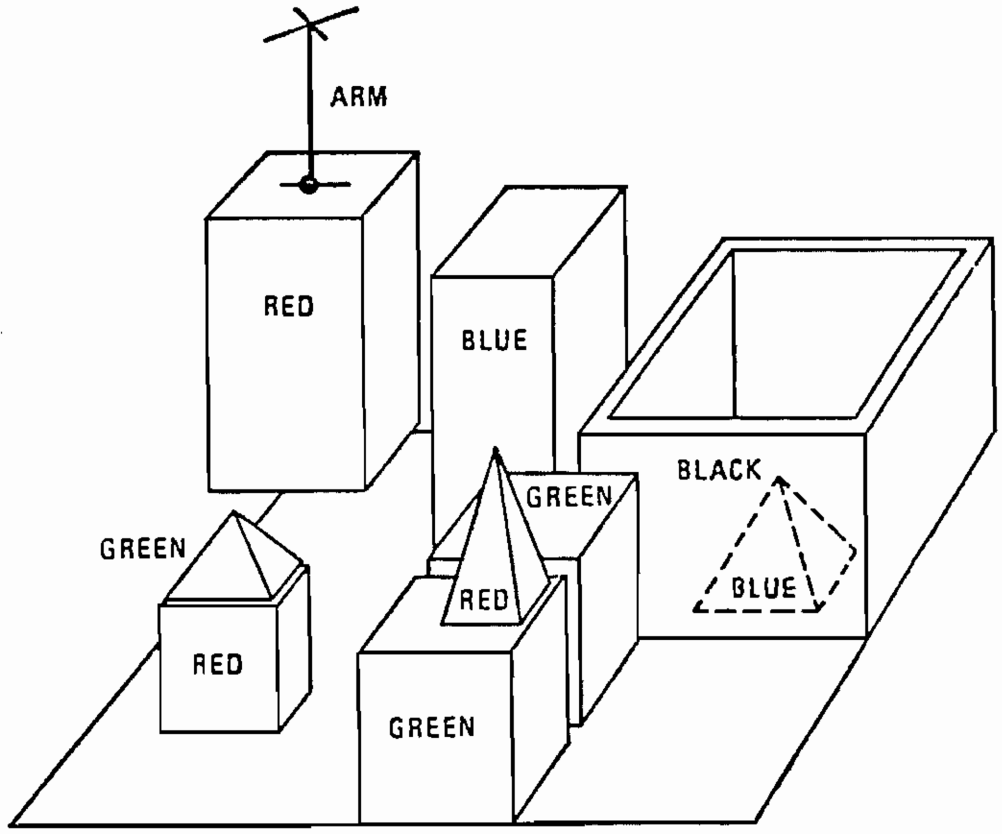


Figure 9 Intermediate State of Microworld

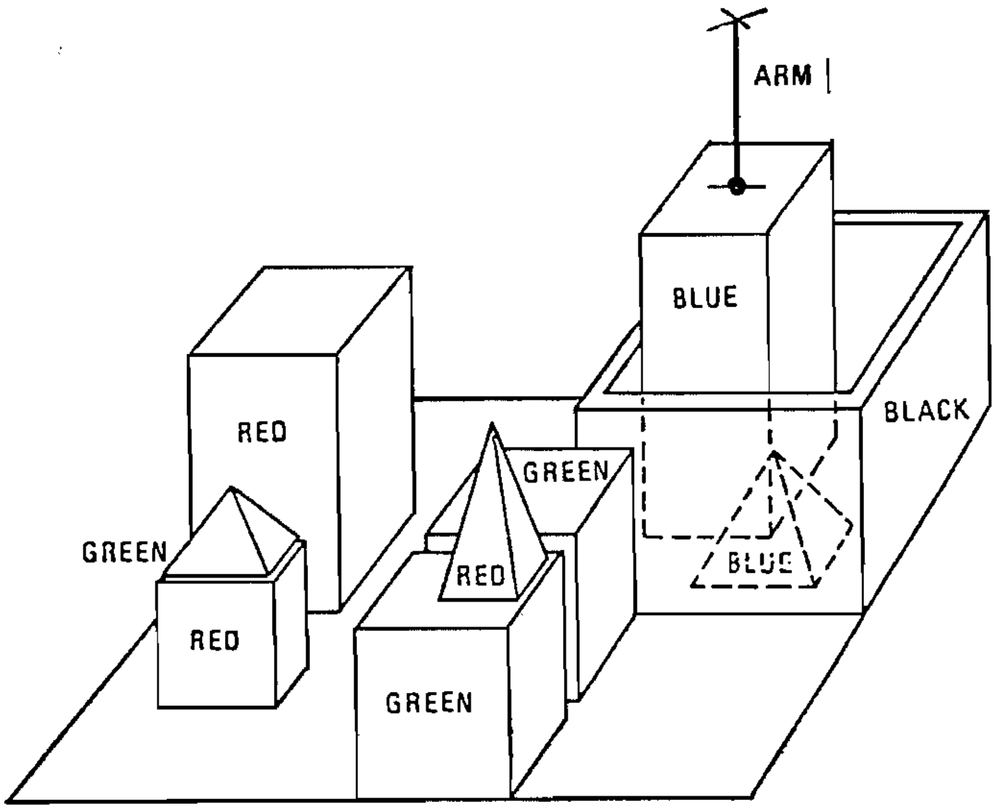


Figure 10 Final State of Microworld

Interaction 4 illustrates SHRDLU's question-answering ability. In the process of answering this question, the blue pyramid and blue block become the most salient objects in the linguistic context.

In Interaction 5, the user again (as in 2) makes use of the phrase "the pyramid." But this time SHRDLU is able to assign a specific referent to the noun phrase. In particular, SHRDLU picks out the pyramid that was most recently mentioned in the conversation.

In Interaction 6, SHRDLU displays its ability to count and to handle negative constructions. Its response suggests a key limitation of SHRDLU--it assumes it knows everything there is to know about the microworld. In particular, it assumes it knows about all the blocks. To see how this simplifies things, ask yourself how many people (or blocks) are not in the room you are in.

Interactions 7 and 8 show SHRDLU's rudimentary ability to deal with references to actions, rather than just to simple objects, and to recall the history of previous actions and their causal linkage--essential skills for entering into conversations about dynamic worlds.

2. Limitations

The sample SHRDLU dialogue clearly illustrates the system's capacity to cope with dynamics in both the physical and linguistic environments.

As observed by Wilks [28], SHRDLU's apparent power derives in large measure from dealing with a logical, small, simple, closed microworld. The microworld is logical in that all the facts about it can be stated in terms of first-order logic. This is fortunate because powerful problem-solving methods exist for dealing with bodies of facts in this form. But these powerful methods are computationally expensive and become increasingly inefficient as objects and facts proliferate and become more complex.

Fortunately for SHLRDU, its microworld contains fewer than a dozen objects, even counting the robot arm, and their interrelationships and the actions that can change those interrelationships are relatively simple. For example, there are no messy objects like pieces of rope or bodies of water that can assume an infinite variety of shapes or that can split or combine to form new, complex objects. Furthermore, there are no cerebral creatures in the microworld to complicate the picture by having wants, goals, beliefs, and the like.

Finally, the microworld is closed in that it is assumed to be completely knowable. If the box is empty and SHRDLU plans to put the blue block in it, there is no possibility that some gremlin will fill up the box with junk, or turn it upside down, or break it, or nail a lid on it, or perform any of infinitely many potential gremlinesque activities. If SHRDLU wants to know whether the box is empty, it simply tries to prove that it contains something, and watches to see if the proof fails. Anything in the microworld that is knowable is provable, and therefore anything that cannot be proved must be false.

The world that people deal with in their everyday conversations is of course extensive, complex, and largely unknown and unknowable. To use natural language to converse about the real world, more sophisticated methods are needed.

C. Systems with Knowledge of Ordinary Situations

1. The SAM System

One of the more interesting attempts to deal with ordinary human situations, in contrast to interfacing with a database or a model of a microworld, was made by Roger Schank and Robert Abelson, aided by their students at Yale University. Their system, SAM (for Script Applier Mechanism) [22], was built to cope with certain kinds of everyday problems. For example, the system is told the following story:

John went to a restaurant.
He ordered the lamb.
He paid the cashier and
left the restaurant.

Then the system is asked:

What did John eat?

Most of us would think it trivial for a system to answer that John ate the lamb--but nowhere in the story above is this explicitly stated! Nor is it directly deducible from what was said. To understand the story, the system must have both knowledge of what usually happens in restaurants and an ability to apply that knowledge to particular situations.

Schank and Abelson encoded SAM's knowledge about everyday situations in formal constructs called scripts. The information contained in a script about restaurants is shown in Figure 11. It includes a list of players who participate in the normal routine of a restaurant, a list of props supporting the action, and a sequence of generic actions that characterize what usually happens when a customer visits a restaurant.

The information in this script can be used to support a variety of commonsense-reasoning tasks, including a reply to the question "What did John eat?," which was posed above. The processing is as follows. The system identifies "John went to a restaurant" with Action 1 from the script. In doing this, John is assigned the role of the customer. The system identifies "He ordered the lamb" with Action 4, assigning the role of the food to "the lamb." (Notice that with "he" referring to John, the customer is the same in both 1 and 4.) "He paid the cashier" is identified with Action 9 and "[he] left the restaurant" with Action 10.

Although not all the actions in the script were explicitly mentioned, it is reasonable (but not strictly necessary) to assume that they happened nonetheless. In particular, Action 6, the customer eating the food, probably did happen. Moreover, because the entities playing

Players: customer, server, cashier

Props: restaurant, table, menu, food, check, payment, tip

Actions:

1. Customer goes to restaurant
2. Customer goes to table
3. Server brings menu
4. Customer orders food
5. Server brings food
6. Customer eats food
7. Server brings check
8. Customer leaves tip for server
9. Customer gives payment to cashier
10. Customer leaves restaurant.

Figure 11 The Restaurant Script

the various parts in the script remain constant throughout its enactment, the system assumes that, for this particular visit to the restaurant, the customer in each action is John and the food is "the lamb." Therefore, Action 6 particularizes to "John ate the lamb," providing the answer to the original question.

Information in scripts can be used for more than just answering questions; it can be used to produce "paraphrases" of a story. For example, SAM can convert the original story

> John went to a restaurant. He sat down. He got mad.
> He left.

into the "paraphrase"

> John was hungry. He decided to go to a restaurant. He went to one. He sat down in a chair. A waiter did not go to the table. John became upset. He decided he was going to leave the restaurant. He left it.

The restatement of the story is clearly not a true paraphrase, in that it adds many details based on speculation about what happened. But such an ability to speculate on the basis of knowledge about how our everyday world is structured is the very feature that makes SAM interesting.

Much of SAM's knowledge is not about natural language at all, but rather about our everyday world. SAM clearly demonstrates that understanding natural-language stories about mundane actions requires more than a knowledge of language--it requires a knowledge of the world itself. The more one studies language, the more one realizes that fluent communication in natural language is a process of the total intellect. Language, thought, and knowledge are inextricably intertwined.

2. Limitations of SAM

SAM's scripts provided one of the first mechanisms in a language processor for dealing with the structured sequences of actions that make up much of ordinary life, but they suffer from a number of limitations:

* Only a single object can serve the role of player or prop. This makes it impossible to handle stories about restaurants with many tables, customers, or servers. The problem of figuring out what the phrase "the customer" refers to becomes trivial, because there can be only one customer.
* The actions in a script follow a strictly linear sequence, making it impossible to deal with alternative possibilities, simultaneous or overlapping actions, or a repetition of actions.
* It is difficult to determine which particular script or scripts are appropriate for understanding a given story.

3. The TDUS System

The SHLRDU example discussed earlier suggested the potential richness of interactive dialogue in context. The SAM example showed how inference, i.e., filling in the blanks regarding what was implied as well as what was explicitly stated, is essential in understanding natural language. To determine how knowledge-based inference and dialogue management interact, as well as to work toward solving a problem of great practical value, a group of researchers at SRI International investigated cooperative, task-oriented man-machine dialogue [19]. They developed a system called TDUS (for Task-Oriented Dialogue Understanding System), whose goal was to communicate with a human apprentice about various repair operations on electromechanical equipment. The key research problems considered were those concerned with how to encode knowledge about the repair operations and how to follow the context of a dialogue as the apprentice moved from task to task in the course of performing a repair operation.

In TDUS, information about how various tasks can be performed is recorded in data structures called procedural networks [21], which can be viewed as generalizations of scripts. Simplified procedural nets are shown in Figures 12, 13 and 14. For example, the net of Figure 12 indicates how the task of installing a pump for an aircompressor can be decomposed into a number of subtasks. In particular, the first subtask is to attach the pump to the platform. Once this is done, either the aftercooler elbow or the brace may be installed. Once the aftercooler elbow is installed, the aftercooler may be installed. Once the brace is installed, the pulley can be installed. When both the aftercooler and the pulley have been installed, regardless of the order in which these jobs were done, the task of installing the pump is complete.

It will be observed that, much like a script, the procedural net associates an action with a number of subactions. But, as opposed to the strict sequence of actions in a script, the procedural net imposes only a partial ordering on subactions. Moreover, subactions are usually associated with procedural nets of their own, which specify in yet greater detail how tasks are decomposed. For example, the "attach pump" action referred to in Figure 12 is described further in Figure 13, while the "secure with bolts" action referred to in Figure 13 is described further in Figure 14. The net of Figure 14 contains a loop specifying the repeated procedure of using a wrench W to tighten each bolt.

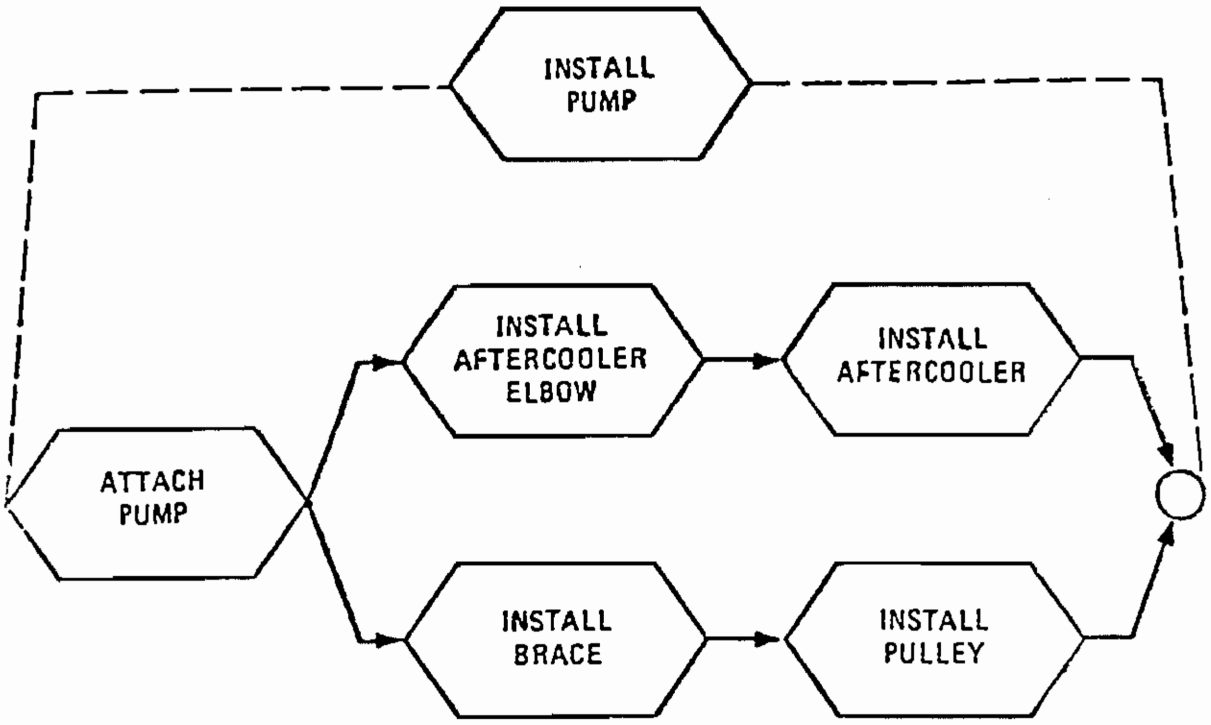


Figure 12 Procedural Network For Installing Pump

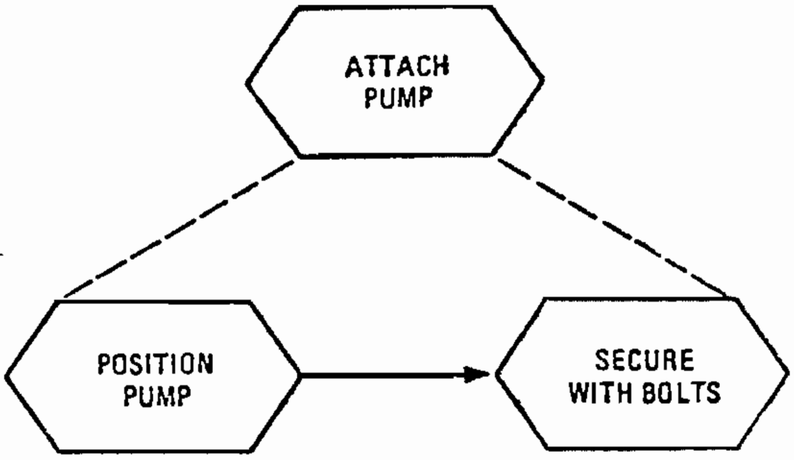


Figure 13 Procedural Network For Attaching Pump

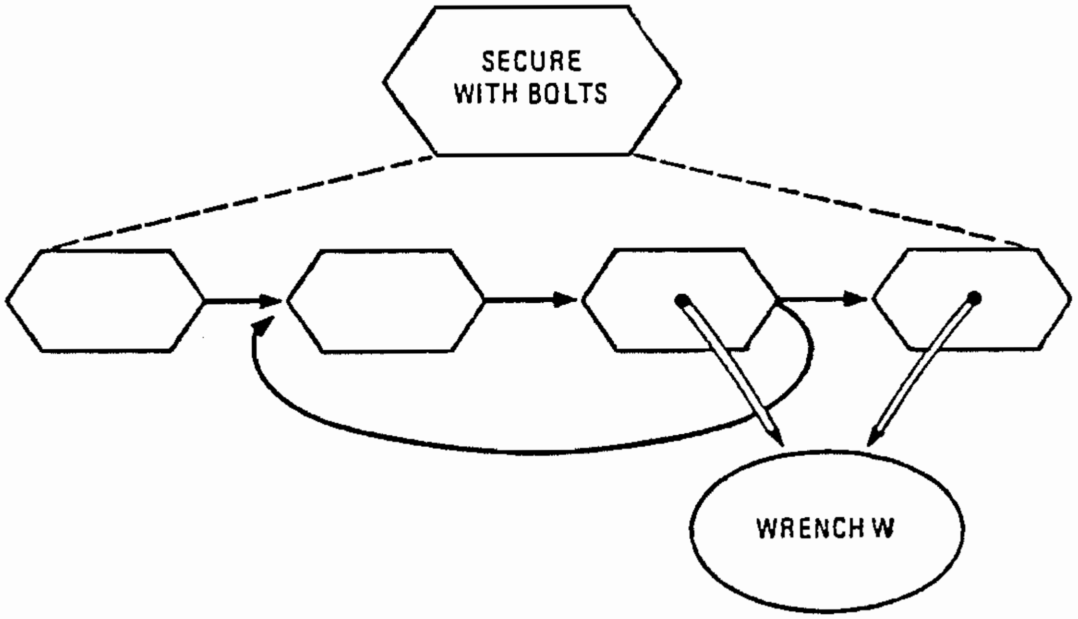


Figure 14 Procedural Network For Securing With Bolts

As mentioned above, a major problem for NLP systems is that of following the dialogue context and being able to ascertain the referents of noun phrases by taking the context into account. In preparing to build the TDUS system, Barbara Grosz [8] collected a number of dialogues between human experts and apprentices performing repair tasks. After constructing procedural nets for the tasks, it was discovered that, as a general rule, the structure of task-oriented dialogues closely follows the structure of the nets representing the decomposition of the task itself. As shown in Figure 15, if a task decomposes into Subtasks A and B, the dialogue will tend to start with general information about the overall task, then enter a subdialogue about Subtask A followed by a subdialogue about Subtask B.

Of greater interest is the fact that referential expressions tend to refer to objects salient in the current subtask or higher in the task hierarchy, but do not generally refer to objects in sibling subtasks. For example, if in the dialogue of Figure 15 a wrench W_1 is

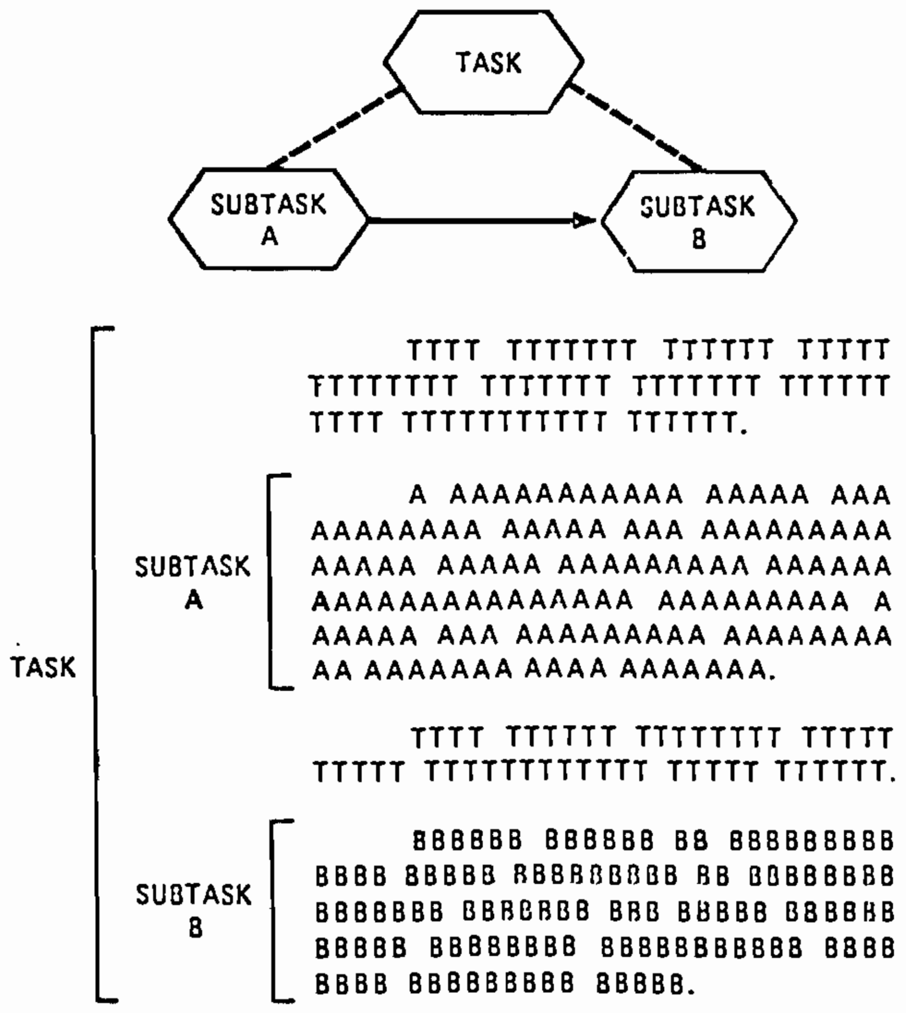


Figure 15 The Parallel Structure Of Tasks And Dialogues

mentioned in one of the initial utterances before entering subdialogue A, and if a second wrench W_2 is mentioned within Subdialogue A, then the phrase "the wrench" uttered in Subdialogue B is more likely to refer to W_1 than to W_2--even though W_2 was mentioned more recently. (This is clearly in violation of the rule used by SHRDLU in Interaction 5 of Figure 7.) In this regard, referential expressions in natural languages tend to follow much the same conventions as do variable references in block-structured programming languages, such as ALGOL and PASCAL. But the block structure of natural language is not indicated explicitly.

TDUS's ability to follow real-world tasks and to deal with shifts in both the linguistic and physical context is illustrated in the transcript of Figure 16, which relates to information contained in the nets of Figures 12-14. In Interaction 1 the apprentice sets the stage by specifying where he is in the overall task. This allows TDUS to pinpoint exactly where in the INSTALL-PUMP net the apprentice believes himself to be. In Interaction 2, the user refers to a bolting action and to a wrench used in that action. Neither of these is in the top-level representation of the ATTACH-PUMP net, but they both occur in the subordinate net of Figure 14. TDUS finds this subnet and determines that no other relevant subnet refers to a different bolting action. Therefore, the system can respond by specifying the wrench mentioned in the net shown in Figure 14.

In Interaction 3 the apprentice asks for the location of the bolts. TDUS does not operate under the closed-world assumption and realizes that there are some things it does not know. Luckily, the user finds the bolts by himself.

In Interaction 5 the apprentice makes reference to "the wrench." TDUS determines that the reference is to the wrench mentioned in Interaction 2, and tells the user its location. In Interaction 6, TDUS resolves "it" as pertaining to the same object.

In Interaction 7 the apprentice tells TDUS that the pump-attaching operation has been completed. From this the system infers that all the constituent subtasks have also been done. To make sure, it

```
1- I AM ATTACHING THE PUMP.

   OK.

2- WHICH WRENCH SHOULD I USE TO BOLT IT?

   USE THE BOX-END WRENCH.

3- WHERE ARE THE BOLTS?

   I DON'T KNOW.

4- I FOUND THEM.

   OK.

5- WHERE IS THE WRENCH?

   THE BOX-END WRENCH IS ON THE TABLE.

6- I FOUND IT.

   OK.

7- I ATTACHED THE PUMP.

   OK.

THE FOLLOWING HAS BEEN ASSUMED:

   YOU PUT THE PUMP AT THE PLATFORM.  YOU BOLTED THE PUMP TO
   THE PLATFORM WITH THE BOLTS.

8- SHOULD I INSTALL THE PULLEY NOW?

   NO.

THE NEXT STEP IS:

   INSTALL THE AFTERCOOLER ELBOW ON THE PUMP

      OR

   INSTALL THE BRACE ON THE PUMP.

9- I'M DOING THE BRACE NOW.

   OK.

"DO" IS ASSUMED TO MEAN:

   YOU ARE INSTALLING THE BRACE.

10- I'VE DONE IT.

   OK.

"DO" IS ASSUMED TO MEAN:

   YOU INSTALLED THE BRACE.

THE FOLLOWING HAS BEEN ASSUMED:

   YOU PUT THE BRACE AT THE PUMP.

   YOU SECURED THE BRACE.

11- I'VE INSTALLED THE PULLEY.

   OK.
```

Figure 16 A Dialogue With TDUS

reviews these assumptions for the apprentice. This filling in of intermediate steps is much like what we saw earlier in the SAM examples.

In Interaction 8 the user poses a logically complex question involving the modal concept "should." TDUS's rudimentary notion of "should" is to interpret the apprentice's question as asking whether the action mentioned is compatible with one of the possible next steps in the procedural-net model of the task at hand. Since it is not, TDUS answers in the negative, but then goes on to outline the range of acceptable alternatives at this point in the process.

Interaction 9 is of considerable interest in that it contains the pro-verb "do." This is a reference to some action involving the apprentice and the brace. But just what that action is--is not spelled out. By examining the procedural net, TDUS determines that the likely thing to do with a brace in this context is to install it--so TDUS makes that assumption and reports it to the apprentice.

Interaction 10 demonstrates TDUS's ability to comprehend an extreme case of contextually dependent use of language. Neither "done" nor "it" conveys much information. Note in particular that "it" here refers not to some object in the world, but rather to an action that was previously alluded to by the phrase "doing the brace."

4. Limitations of TDUS

TDUS exhibits a reasonable understanding of the interplay among various types of possible real-world actions, and can follow the evolution of particular instantiations of those actions. However, it has very little understanding of the goals and motivations of the apprentice with whom it holds conversations.

An exchange well beyond the capability of TDUS is shown in the following actual dialogue between a novice and an expert mechanic:

1- <u>WHAT DO I DO NEXT?</u>

REMOVE THE BOLT.

2- <u>HOW DO I GET IT OFF?</u>

USE THE RATCHET WRENCH.

3- <u>WHAT'S A RATCHET WRENCH?</u>

IT'S ON THE TABLE.

The key point to note here is that, in Interaction 3, the response is not a direct answer to the question. If TDUS could answer this question at all, it would very likely respond with a dictionary definition such as "a tool for grasping and turning the head of a bolt, consisting of fixed or adjustable jaws mounted on a pawl that is engaged by the toothed end of a gripping bar." The human being that produced the answer to 3 understood the motivation behind the question. In particular, he understood that the question was asked

so the inquirer would know the description of a ratchet wrench,

so he could find it,

so he could grasp it,

so he could have use of it,

so he could remove the bolts.

Understanding the inquirer's hierarchy of goals, the respondent addresses one of the goals closer to the end of the chain (finding the wrench). In this fashion, the cooperative respondent saves the inquirer the trouble of having to take the step of locating the wrench from its description.

D. Beyond Current Systems

Researchers in computational linguistics have only recently begun to appreciate the impact on natural-language communication of what the participants in a conversation know about each other's knowledge, beliefs, plans, and goals. To appreciate the importance of such knowledge, let us consider the situation illustrated in Figure 17.

A young mother is giving a birthday party for Junior, and Grandma has come to help. Grandma's task is to light the candles on the cake, so she asks, "When shall I light the candles?" The mother replies, "We'll have cake as soon as the children wash their hands," which informs Grandma that it will be about five minutes. The mother has observed that her big-eared kids are listening, so she phrases her response to serve multiple purposes for multiple audiences. With her one statement she tells Grandma when the candles need to be lit and, at the same time--in a very nice, indirect way--tells the children to get their dirty hands washed. The mother knows that her response to Grandma will serve this purpose, because she knows that

* The children want the cake.
* Her response to Grandma will convey to them the information that all that stands in the way of their getting it is to wash their hands.
* If they know that all that stands in the way of their getting cake is to wash their hands, they will perform the ritual forthwith.
* It takes them about five minutes to wash their hands.

Similarly, Grandma now knows that she should light the candles in five minutes, because she knows that the mother knows all the circumstances just outlined, and that the mother knows Grandma knows that the mother knows it. Thus, Grandma can infer that the mother expects her to understand that the children are being told to wash their hands, that they are motivated to do it right away, and the result five minutes later will presumably be washed hands reaching avidly for cake.

No system is currently capable of handling language with this level of sophistication, but a number of researchers are actively engaged in

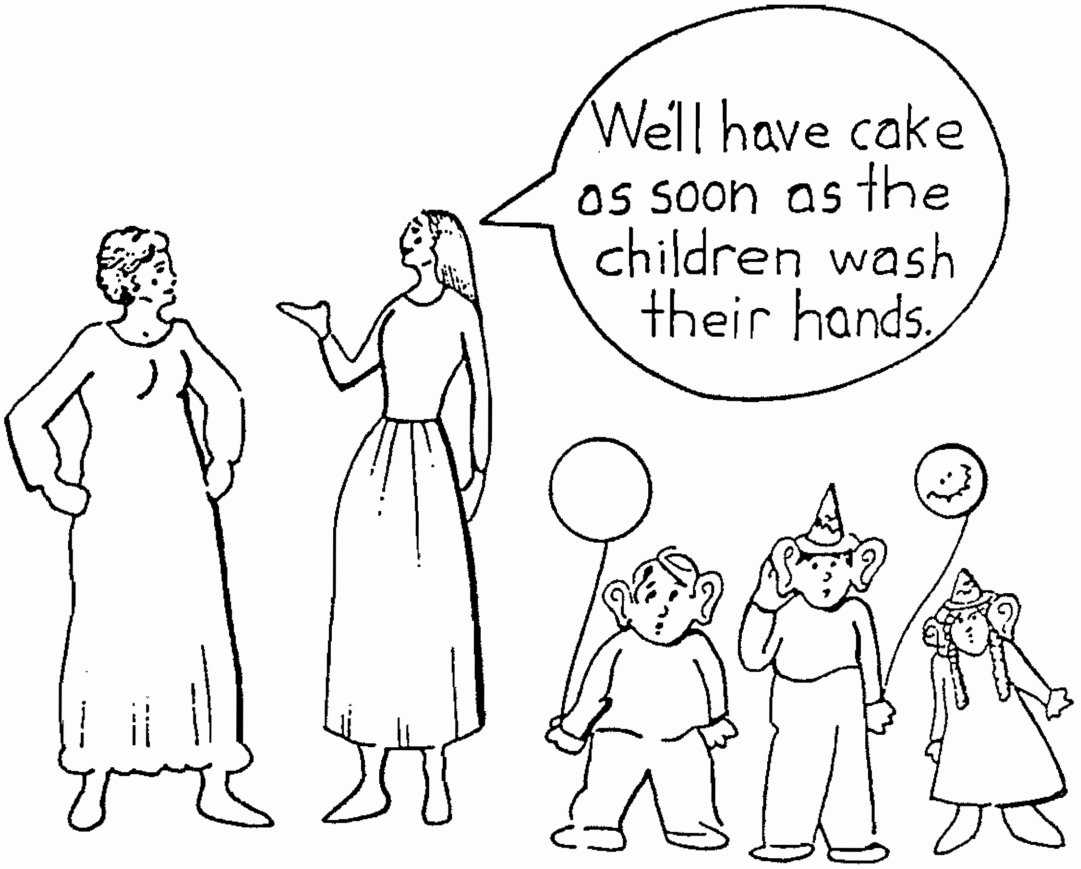


Figure 17 A Response Serving Multiple Purposes

studying the various problems involved [3], [5], [7], [13], [14], [16], [18], [6], [23], [27]. Much of this work is concerned with the difficulties of interpreting not just what is literally said, but also of establishing the underlying intention. Should this work succeed, eventually systems may be capable of the kind of reasoning indicated in Figure 18.

USER: THE TOOLBOX IS LOCKED.

SYSTEM: (WHY IS HE TELLING ME THIS? I ALREADY KNOW THE BOX IS LOCKED.

(I KNOW THE USER NEEDS TO GET IN. PERHAPS HE IS TELLING ME THE BOX IS LOCKED BECAUSE HE BELIEVES I CAN SOMEHOW HELP.

(TO GET IN TAKES A KEY. THE USER KNOWS THIS AND KNOWS I KNOW IT. THE KEY IS IN THE DRAWER. IF THE USER KNEW THIS HE WOULD JUST UNLOCK THE BOX. THEREFORE, HE MUST NOT KNOW IT.

(I CAN MAKE HIM COME TO KNOW IT BY SAYING "THE KEY IS IN THE DRAWER." I AM SUPPOSED TO HELP. I WILL SAY IT.)

THE KEY IS IN THE DRAWER.

Figure 18 Beyond Literal Interpretation

The examples of the locked toolbox and the birthday party support a central point: communicating in natural language is an activity of the total intellect. Seen in broad perspective, the use of natural language can be placed in a general framework that seeks to account for all human activity. Within this framework, humans are seen as intelligent beings motivated by complex sets of goals they seek to fulfill by planning, executing, and monitoring sequences of actions--some of which are physical, some linguistic. That is, uttering a sentence is just as much an action as taking a step or taking a bath. Whereas the usual purpose of a physical action is to alter the physical world, the usual purpose of a linguistic action is to alter the mental states of the hearers. In the latter case, the desired modification may be to add knowledge, change a mood, or establish a new goal for the hearers.

A speaker may plan and execute linguistic actions to change some aspect of a hearer's mental state, not as an end in itself, but as part of an overall plan to achieve some more ambitious end. Just as a child might push over the first domino of a long row to make them all tumble in sequence, a lifeguard at the beach may yell "Shark!" at swimmers to set off a chain of reasoning in their minds that will result in a mad dash for the shore, which is the lifeguard's intended mechanism for accomplishing his primary goal of preserving life.

Given this view of how language works, it becomes less important to ask what a given utterance means (what does "Shark!" mean?), and more important to ask about the effect it produces. People in advertising have an explicit understanding of this concept, but all of us use it implicitly when we understand the agony conveyed by the string of curses uttered by the handyman who smashes his finger, and when we realize that our friend's question, "Do you know the time?" deserves more than a "yes" or "no" answer.

The understanding of poetry can even be cast in this mold. The poet deliberately triggers certain chains of inference in his listeners. Indeed, an important element in the appreciation of poetry is the listener's awareness of the interplay among the inference chains he is led to follow, the chains he follows partway that turn out to be not quite appropriate, and the surface meanings of the sentences comprising the poem itself. To experience this, just consider the title of T. S. Eliot's poem, "The Love Song of J. Alfred Prufrock."

IV THE NATURE OF NATURAL-LANGUAGE RESEARCH

The previous section has discussed the capabilities and limitations of specific NLP systems, but it must be recognized that these systems are merely spinoffs of the underlying science. In essence, most NLP researchers do not think of themselves as engineers seeking to evolve ever better NLP systems, but rather as scientists concerned with the following related problems:

* Identification of sources of knowledge that are necessary for understanding or generating natural language.
* Discovery or devising of mechanisms for encoding and applying such knowledge in a mechanical device.
* Creation of integration frameworks to control and coordinate the application of a variety of knowledge sources.

Once sources of knowledge have been identified, whole subdisciplines come into being to study the associated bodies of knowledge, their structure, and methods for their computerization. Some of the major knowledge sources are discussed below.

<u>Lexical knowledge</u> concerns individual words, the parts of speech they belong to, and their meanings.

<u>Syntactic knowledge</u> has to do with the grouping of words into meaningful phrases. For example, it is syntactic knowledge that distinguishes between the following two sentences:

NAME THE PARTS OF THE PUMP THAT <u>WAS</u> FIXED BY JOE.

NAME THE PARTS OF THE PUMP THAT <u>WERE</u> FIXED BY JOE.

In particular, it is the syntactic number distinction between WAS and WERE that indicates whether the pump or the parts were fixed.

Syntactic ambiguity is a common source of trouble in NLP systems. For example, decisions about where to associate the prepositional phrase "on the table" in

PUT THE HAMMER IN THE TOOLBOX ON THE TABLE.

can lead to any one of the interpretations

PUT THE HAMMER THAT IS IN THE TOOLBOX ONTO THE TABLE.

PUT THE HAMMER INTO THE TOOLBOX THAT IS ON THE TABLE.

WHILE YOU STAND ON THE TABLE, PUT THE HAMMER INTO THE TOOLBOX.

Compositional Semantics is the knowledge of how to compose the (literal) meaning of large syntactic units from the semantics of their subparts. Its utility is illustrated by the pair of sentences

THE MAN HELD THE NUT (with a wrench).
THE WRENCH HELD THE NUT.

These two sentences are syntactically identical, but the subject of the first sentence is the agent of the action "hold," whereas the subject of the second is the instrument used by the agent. The lexical entry for the verb HOLD indicates that it is used to refer to actions in which an agent (usually a person) using an instrument exerts a force on some physical object. The syntactic subject of the verb might refer either to the agent or the instrument. But the semantics of HOLD is that these roles must be filled by objects of mutually disjoint classes of objects. Utilization of this knowledge allows a system to assign the role of agent to THE MAN, but assign the role of instrument to THE WRENCH.

Discourse Knowledge concerns the way clues from the current context are used to help interpret a sentence. For example, if we have just been talking about this month's issue of BYTE, the noun phrase "the magazine" in

I'VE ALREADY READ MY COPY OF THE MAGAZINE.

is easily understood in this context as referring to this month's issue of BYTE. Yet each of us has personal knowledge of hundreds of issues of various magazines. The ability to pick out the one of current interest is based on specific knowledge of the current situation.

World Knowledge is concerned with information about how the world is currently configured and about physical constraints upon possible configurations. For example, we understand

PRESIDENT REAGAN FLEW TO CALIFORNIA.

as meaning that he was flown to California as a passenger in an airplane. Had the sentence been about a bird, we might have taken the sentence to mean that the bird did the flying.

As an example of how knowledge about the current physical situation can be of aid in understanding sentences, let us again consider the sentence

PUT THE HAMMER IN THE TOOLBOX ON THE TABLE.

which was discussed in the foregoing section on syntax. If we know that the hammer is currently in a toolbox on the floor, then the only interpretation of the sentence is to lift the hammer out of the toolbox and place it onto the table. The other interpretations are ruled out, because they are impossible in the current state of the world.

Knowledge of Mental States relates to comprehending the knowledge and goals of other participants in a dialogue. The use of such knowledge is shown in the locked-toolbox example in Figure 18.

V COST AS A FUNCTION OF CAPABILITY

The preceding sections sketched a spectrum of NLP capabilities--ranging from isolated questions about the data in conventional databases, through the literal interpretation of utterances in dynamic contexts, to an understanding of the underlying goals and mental states of participants in a dialogue. As would be expected, progression through this spectrum entails rapidly escalating costs in two areas: the research and programming effort required to reach a particular level of capability, and the computing resources (measured in the number of machine instructions that must be executed and the memory requirements) needed to function at a given level.

This situation is illustrated graphically in Figure 19, which plots cost as a function of system capability. Although the diagram shows a sharp rise in cost with increased capability, the situation is probably even more dramatic than indicated, and the cost scale might best be interpreted as being logarithmic.

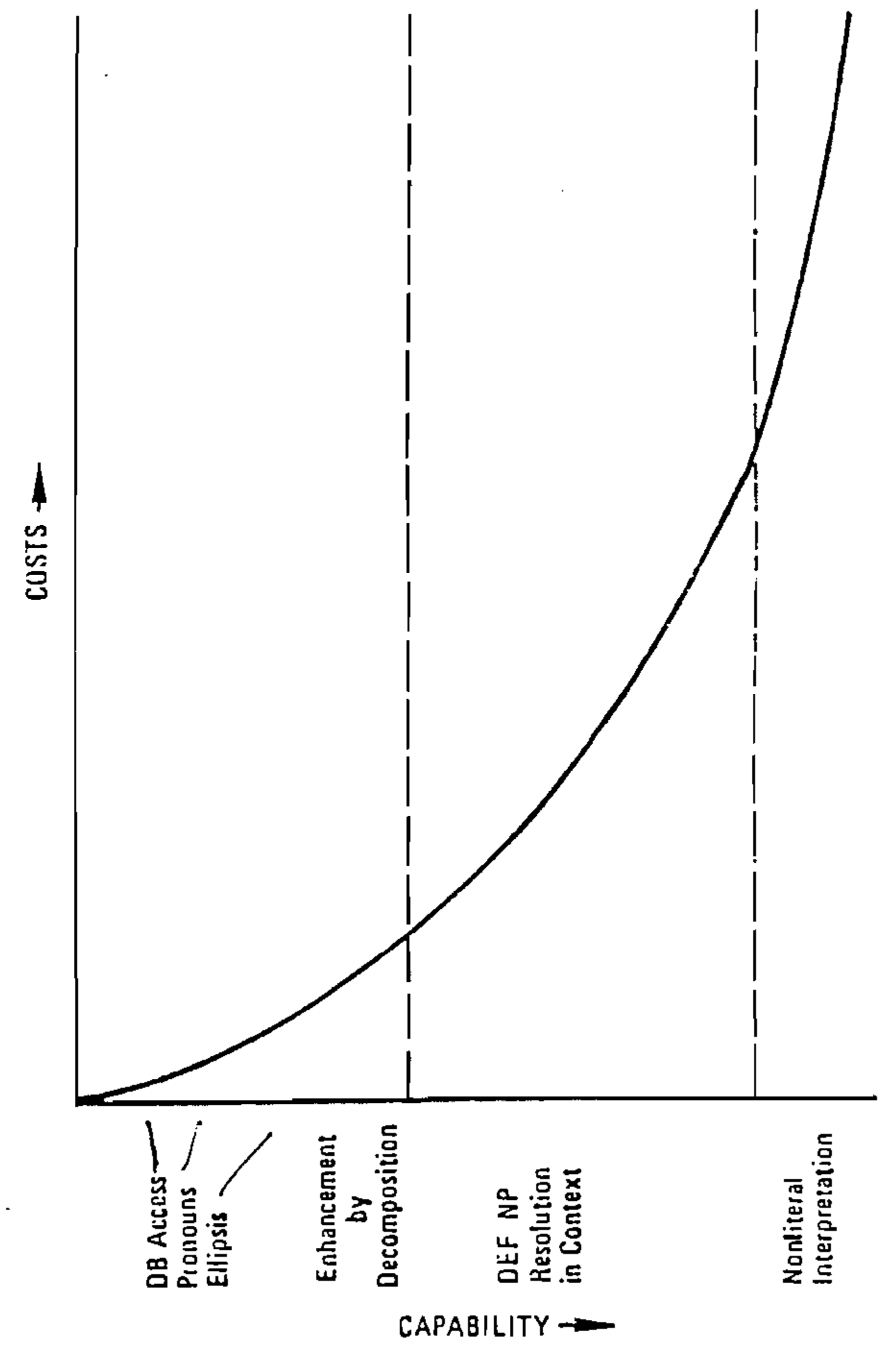


Figure 19 Cost As A Function Of Capability

The capability dimension has been separated into three major regions by two dividing lines. It appears that capabilities beyond (to the right of) the leftmost line require systems with explicit models of concrete objects in the world, the relationships among them, and the types of processes that can alter those relationships. The more advanced capabilities beyond the rightmost line require further enhancements for modeling such things as the mental states of dialogue participants.

Figure 20 reproduces the curve of Figure 19, but also shows curves for developing three kinds of systems. Systems on Curve A are built without the use of explicit models. They cover most of the principal linguistic phenomena needed for accessing conventional databases and, up to the point at which Curve A intersects Curve B, can be constructed and operated more economically than other types of systems. As the need for world models increases, attempts to circumvent the problem by using various ad hoc methods become prohibitively expensive or downright impossible.

Systems on Curve B are built with explicit world models that lack the ability to deal with intensional concepts*, such as the goals and beliefs of others. Although these systems have greater capability potential than those on Curve A, they entail considerable initial costs in the construction of machinery to support and exploit the models.

----

* Intensional concepts are those that take into account the meaning, rather than just the truth values, of logical sentences. In standard logic, the truth value of a complex formula depends <u>only</u> on the truth of its subexpressions; e.g., the truth of (P OR Q) depends only on the truth of P and the truth of Q; no other properties of P and Q matter. The operator BELIEVE, however, is intensional because the truth of "A believes that P" depends on the meaning of P, not just its truth value. The problem here is that many of the rules of standard logic, such as substitution of equals for equals, do not apply when intensional operators are involved. To use a classic example, since "the morning star" and "the evening star" refer to the same object, it must be the case that "the morning star is Venus" is true if and only if "the evening star is Venus" is true. However, it might be that "John believes the morning star is Venus" is true, but that "John believes the evening star is Venus" is false because, although the two embedded sentences possess the same truth value, they differ in meaning.

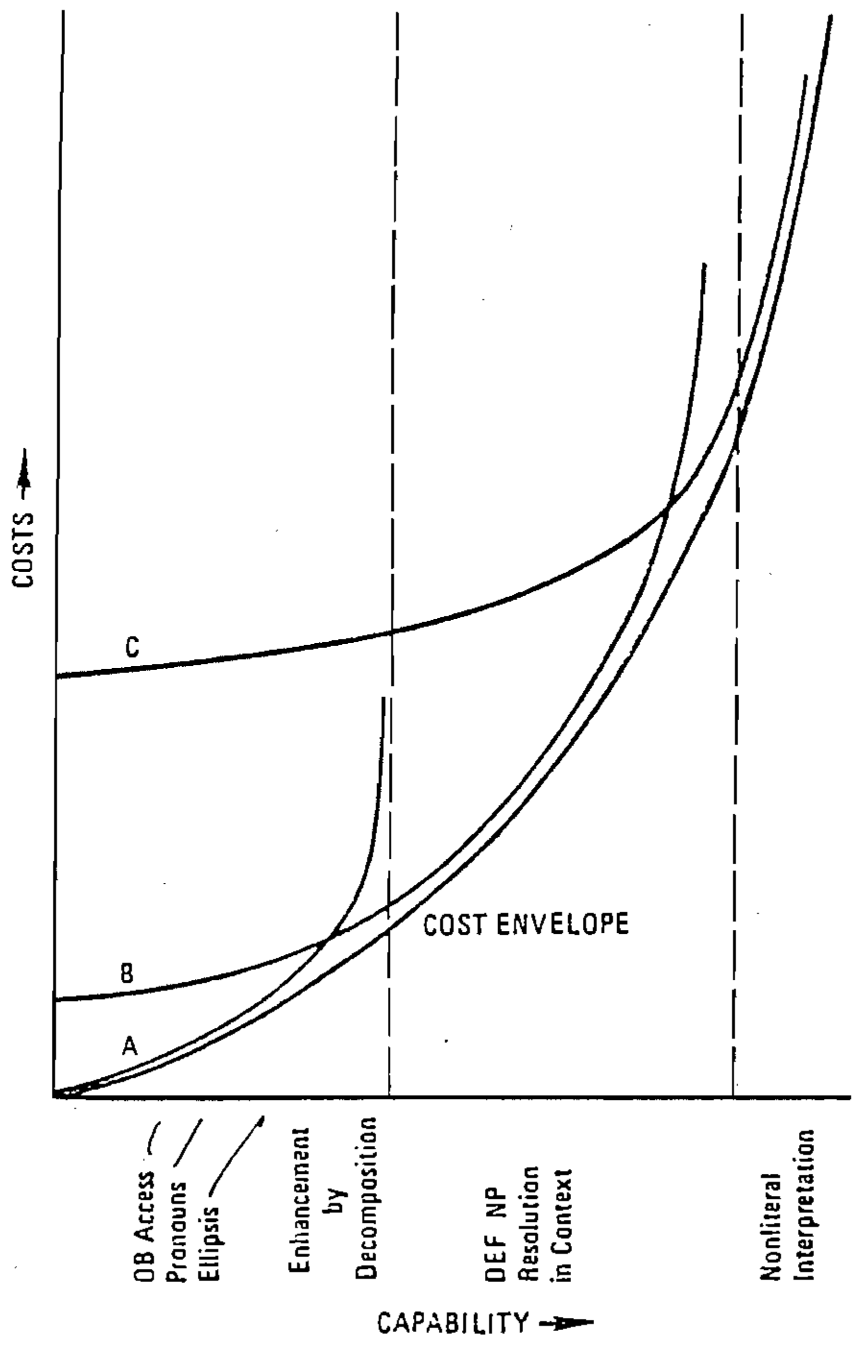


Figure 20 Costs For Three Types Of Systems

Systems on Curve C, if any existed, would be built upon a knowledge base supporting many intensional concepts. The initial costs of creating computational machinery to support this level of sophistication appear to be quite high.

In today's computing environment, and in light of the current state of the art in NLP, the only systems that perform robustly and efficiently are systems of Type A. As mentioned in Section III-A-3, there are a number of systems of this type, including one (INTELLECT [10]) that is available now as a commercial product.

A number of experimental systems of Type B have been built, including SHRDLU, SAM and TDUS, discussed above. But these systems are currently of little practical value because they are relatively slow and use models that, while consuming considerable memory, cover pieces of the world too small to be of much more than academic interest.

Computational linguists and workers in related fields are devoting considerable attention to the problems of Type-C systems; but currently only bits and pieces of components for such systems are being constructed. For example, Moore [17] and Appelt [1] have devised formal methods for reasoning about the knowledge of others, and Perrault, Allen, and Cohen [6] [18] have devised systems that actually plan speech acts. But there is a huge gulf between the first experiments that demonstrate the feasibility of a principle and the creation of useful systems based on that principle. Our current state of understanding in NLP is perhaps similar to that of the ancient makers of Chinese fireworks as compared with modern space research. In terms of achieving fluency in NLP, our current experiments are merely fireworks poking a few holes in the darkness.

Nevertheless, even though the fluent use of natural language by machines may still be years away, it is important to realize that many practical applications of NLP can be supported by systems of Types A or B, both of which can be built <u>now</u>. Although the computational costs for NLP will always be relatively high when compared with machine languages, the introduction of VLSI technology promises to ease the attendant dollar cost. In particular, processes that were once performed only in the laboratory, on research computers costing over $1 million, are becoming practicable on personal machines. As a general trend, the expense of programming continues to rise while computer hardware continues to drop in price. For some applications, we may have already reached the point at which it is cheaper to create systems that use subsets of English than it is to hire and train people to use formal languages.

VI MORE ON THE CURRENT CAPABILITIES OF TYPE-A SYSTEMS

To reach a better understanding of the types of natural-language processing that are possible on a cost-effective basis using technology available today, consider the dialogue of Figure 21, which shows some of the more advanced types of processing available in the LADDER system.

Interaction 1 shows how LADDER deals with sentences it cannot understand. After trying to interpret the input as first a complete and then an elliptic sentence, LADDER prints an error message indicating that the word NY is not known to it. The system also indicates that to complete the partial input "How far is the Kennedy from the port of," the system needs a construction matching its internal concept of a <PORT>. In Interaction 2, the user seeks information about what LADDER would accept as a valid expression for a <PORT>. This is a very interesting question because it is not about the data in the database, but is rather a question about the language-processing system itself. LADDER can process questions about a variety of knowledge sources, including special routines that inspect its own internal information about the linguistic constructions it can accept. Using one of these special routines, LADDER produces a list of possible expressions for <PORT>, including the expression NEW YORK, which the user identifies as being what he intended by NY.

At this point the user could rephrase his question by using NEW YORK in place of NY. But it is important for NLP systems to accommodate the linguistic constructions that their users are comfortable with--to accept inputs in the users' own terms rather than force the users to master the system's terms. In particular, the learning of synonyms is relatively inexpensive computationally and is quite helpful to users. In Interaction 3, the user types a statement in English to tell LADDER that NY is a synonym for NEW YORK. Then, in Interaction 4, the user asks that interaction 2 be tried again. This time, using the newly defined synonym, LADDER successfully interprets the question and produces the answer that the Great-Circle Distance from the Kennedy to New York is 1974 miles.

It is worth noting that LADDER must do more to answer Questions 2 and 4 than merely retrieve information from a database. Only the positions of ports and ships are stored in the database--not the distances between them. Thus, LADDER interfaces not only with the database, but also with programs that make calculations based on database information. Some of these are nontrivial. For example, to find how long it would take a ship to reach a given location, LADDER cannot simply divide the great-circle distance by the cruising speed of the ship, because the shortest path between two points on earth very often crosses land masses. So LADDER computes routes that avoid land masses, which requires a knowledge of world geography.

Interactions 4 through 8 illustrate LADDER's ability to learn new syntactic constructions as well as synonyms. Suppose a user has certain questions that he needs to ask repeatedly about different ships. Natural language is ideal for one-time questions, but for those used repeatedly, a shorthand version would be useful. If, as in Interaction 5, the user asks a shorthand question such as "Q1 Kennedy?" he will cause a syntax error. However, the user can easily tell LADDER, by giving an example in English, how a new shorthand is to be interpreted. This is done in Interaction 6. In response to this request, LADDER creates a new production rule that matches inputs that start with "Q1" and end with any expression designating a <SHIP>.

In Interaction 7, this newly defined construction is used to ask for information about the Kennedy. In Interaction 8, to obtain the same information for all the American subs in the North Atlantic, all the user has to type is "American subs in the North Atlantic." LADDER's elliptical-processing routine, operating on the newly defined construction just as on the standard ones, fills in the rest. Because there are multiple answers to the question, a table is produced to display the data retrieved.

In Interaction 9, the user tells LADDER to make a certain input sequence be equivalent to not just one, but a whole series of questions. It is as if the user were writing small programs in English, using English pronouns for formal parameters. In Interaction 10 the new construction is used, but with a different ship from the one used to define the construction.

As an extreme example of this ability to accommodate to user-defined constructions, in Interaction 11 the user tells LADDER a Japanese paraphrase of the English question, "What is the length of the Kennedy?" In Interaction 12, the user poses this question, but about the Fox rather than the Kennedy, entirely in Japanese.

The language-processing capabilities demonstrated in Figures 5 and 21, while far from those possessed by a fluent user of natural language, nevertheless seem quite adequate for a wide range of practical applications. This technology, using the rather exotic language INTERLISP, is available today in a large computing environment. It requires approximately 500 milliseconds to process a query--far less than the time needed to retrieve information from a large database. With proven algorithms and data structures in hand, it is now essentially an engineering task to instantiate this technology on smaller machines in widely available programming languages. Such engineering will require a considerable effort, but the path of development appears to be clear of major theoretical obstacles.

One of the practical problems currently limiting the use of NLP systems for accessing databases is the lack of trained people and good

```
1-How far is the kennedy from the port of NY?
Trying Ellipsis:  ELLIPSIS HAS FAILED
The parser does not expect the word "NY" to follow "HOW FAR IS
THE KENNEDY FROM THE PORT OF"
Options for next word of symbol are:  <PORT>

2- What is a <PORT?
PARSED!
"<PORT>" may be any sequence of words following on of the
patterns:
     <PORT>  => CAPE TOWN
             => NEW YORK
             => NEW ORLEANS
           [etc.]
3- Define NY like New York.
PARSED!
NY

4- redo 2
[i.e., How far is the Kennedy from the port of NY?]
PARSED!
(GCDIST 1974)

5- Q1 Kennedy
[System gives syntax error.]

6- Let "Q1 kennedy" be a paraphrase of "what is the
   position, fuel status and homeport of the Kennedy"
PARSED!
<S> => Q1 <SHIP>

7- q1 kennedy
PARSED!
(POSITION 4000N0600E  PCFUEL 90  PORT MAYPORT)


8- American subs in the North Atlantic
Trying Ellipsis: Q1 AMERICAN SUBS IN THE NORTH ATLANTIC

SHIP          POSITION        PCFUEL      PORT
STURGEON      3700N7600W      100         NORFOLK
WHALE         3750N7700W      100         NORFOLK
ASPRO         3000N3000W      100         NORFOLK
[etc.]

9- Let "show the forcestatus of the Kitty Hawk" be like
   "Display the employment and rediness condition of the
   Kitty Hawk.  Print her destination.  List ships in
   her organization."
PARSED!
[New production added to system.]

10- show the forcestatus of Kennedy
PARSED!
[questions defined in 9 for Kitty Hawk are answered
for the Kennedy.]

11- Define "Kennedy no nagasa wa ikura desuka" like
    "what is the length of the kennedy."
PARSED!
[Production added to system.]

12- Fox no nagasa wa ikura desuka?
PARSED!
LEN = 547
```

Figure 21 More Interactions With LADDER

support tools for creating the knowledge structures needed for each new database. In laboratory systems, researchers have manually compiled bodies of knowledge, such as information about the vocabulary employed in a particular application or about the logical structure of particular databases. Work has already begun on new methodologies to automate this task or even to make it entirely unnecessary [25], [15], [9].

VII CONCLUSION

Considerably more research in computational linguistics will be required before mechanical devices can be created that are fluent in the use of natural language. However, current research efforts are shedding new light on the types of knowledge required for communication in human languages, as well as on prospective mechanisms for encoding and applying that knowledge in computers. These efforts are showing that language use is not an isolated intellectual activity; rather, it also involves our basic facilities for commonsense reasoning and planning. A computer system that is fluent in a natural language will be a genuinely intelligent machine.

Although the fluent use of natural language by machines remains a long-term goal, a number of practical mechanisms have been developed to deal with significant fragments of language in specialized application areas. For many applications, an ability to communicate within such fragments is both sufficient for the task at hand and clearly preferable to forcing users to learn machine-oriented languages. Part Two of this article will focus on methods for providing this ability. In coming years we expect to see NLP employed in an increasing number of practical applications, enabling more and more people to interact directly and effectively with computer systems.

ACKNOWLEDGMENT

We wish to express our thanks to SRI International and Machine Intelligence Corporation for supporting the preparation of this article, and acknowledge the helpful critiques of early drafts that were provided by Barbara Grosz, Norman Haas, Robert Moore, Jane Robinson, and Donald Walker.

REFERENCES

1. D. E. Appelt, "A Planner for Reasoning about Knowledge and Action," _Proc. 1st Annual National Conference on Artificial Intelligence_, Stanford, California, pp. 235-239 (August 1980).

2. A. B. Biermann, "Approaches to Automatic Programming," _Advances in Computers_, Vol. 15, pp. 1-63 (1976).

3. B. Bruce, "Pragmatics in Speech Understanding," _Advance Papers of the Fourth International Joint Conference on Atificial Intelligence_, Tbilisi, Georgia, USSR pp. 461-467 (1975).

4. R. R. Burton, "Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems," BBN Report 3453, Bolt Beranek and Newman, Cambridge, Mass., (December 1976).

5. J. Carbonell, "Politics," in Schank and Riesbeck, pp. 259-317 (1981).

6. P. R. Cohen and C. R. Perrault, "Elements of a Plan Based Theory of Speech Acts," _Cognitive Science_, Vol. 3, No. 3, pp. 177-212 (1979).

7. B. J. Grosz "Utterance and Objective: Issues in Natural Language Communication," _Sixth International Joint Conference on Artificial Intelligence_. Stanford University, Stanford, California, pp. 1067-1076 (1979). (or _AI Magazine_ Vol. 1 No. 1, Pp. 11-20 (1980)).

8. B. J. Grosz, "Focusing and Description in Natural Language Dialogues," In _Elements of Discourse Understanding: Proc. of a Workshop on Computational Aspects of Linguistic Structure and Discourse Setting_, A.K. Joshi, I.A. Sag, and B.L. Webber, eds. Cambridge University Press, Cambridge England (1981).

9. N. Haas and G. Hendrix, "An Approach to Acquiring and Applying Knowledge," _Proc. 1st Annual National Conference on Artificial Intelligence_, Stanford, California, pp. 235-239 (August 1980).

10. L. R. Harris, "User Oriented Data Base Query with the ROBOT Natural Language Query System," _Proc. Third International Conference on Very Large Data Bases_, Tokyo (October 1977).

11. G. G. Hendrix, "The LIFER Manual: A Guide to Building Practical Natural Language Interfaces," SRI Artificial Intelligence Center Technical Note 138, Stanford Research Institute, Menlo Park, California (February 1977).

12. G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data," _ACM Transactions on Database Systems_, Vol. 3, No. 2 (June 1978).

13. J. R. Hobbs and D. A. Evans, "Conversation as Planned Behavior," Technical Note 203, Artificial Intelligence Center, SRI International, Menlo Park, California, (1979).

14. S. J. Kaplan, "Cooperative Responses from a Portable Natural Language Data Base Query System. Ph.D. Dissertation, University of Pennsylvania, Philadelphia (1979).

15. W. H. Lewis, "TED: A Transportable English Datamanager," _Proc. of the Principal Investigators' Meeting of the ACCAT Program_, Monterey, California (October 1979).

16. W. Mann, "Toward a Speech Act Theory for Natural Language Processing," ISI/RR-79-75, USC/Information Sciences Institute, Marina del Rey, California (1980).

17. R. C. Moore, "Reasoning About Knowledge and Action," AI Center Technical Note 191, SRI International, Menlo Park, California (October 1980).

18. C. R. Perrault and J. F. Allen, "A Plan-Based Analysis of Indirect Speech Acts," _American Journal of Computational Linguistics_ Vol. 6, No. 3-4, pp. 167-182 (July-December 1980).

19. A. E. Robinson et al., "Interpreting Natural-Language Utterances in Dialog About Tasks," SRI Technical Note 210, Artificial Intelligence Center, SRI International, Menlo Park, California (1980).

20. J. J. Robinson, "DIAGRAM: a Grammar for Dialogues," Technical Note 205, Artificial Intelligence Center, SRI International, Menlo Park, California (February 1980).

21. E. D. Sacerdoti, _A Structure for Plans and Behavior_. New York: Elsevier North-Holland (1977).

22. R. Schank and R. Abelson, _Scripts, Plans, Goals, and Understanding_. Hillsdale, N.J.: Lawrence Erlbaum Associates (1977).

23. R. Schank and C. Riesbeck, _Inside Computer Understanding_, Hillsdale, N.J., Lawrence Erlbaum (1981).

24. M. Templeton, "EUFID: A Friendly and Flexible Frontend for data Management Systems," _Proc. 1979 National Conference of the Association for Computational Linguistics_, San Diego, California (August 1979).

25. F. B. Thompson and B. H. Thompson, "Practical Natural Language Processing: The REL System as Prototype," pp. 109-168, M. Rubinoff and M. C. Yovits, eds., _Advances in Computers 13_ (Academic Press, New York, 1975).

26. D. Waltz, "Natural Language Access to a Large Data Base: An Engineering Approach," _Proc. 4th Internatioal Joint Conference on Artificial Intelligence_, Tbilisi, USSR, pp. 868-872 (September 1975).

27. R. Wilensky, "Meta-Planning," _Proceedings of the First Annual National Conference on Artificial Intelligence_, Stanford University, Stanford, California, pp. 334-336 (1980).

28. Y. Wilks, "Natural Language Understanding Systems within the AI Paradigm: A Survey and some Comparisons," in A. Zamplolli (Ed.) _Linguistic Structures Processing_. Amsterdam: North-Holland, pp. 341-398 (1977).

29. T. Winograd, _Understanding Natural Language_. New York, Academic Press (1972).

30. W. A. Woods, "Transition Network Grammars for Natural Language Analysis," _Communications of the ACM_, Vol. 13, No. 10, pp. 591-606 (October 1970).

31. W. A. Woods, R. M. Kaplan and B. Nash-Webber, "The Lunar Sciences Natural Language Information System," BBN Report 2378, Bolt Beranek and Newman, Cambridge, Mass., (1972).