Shifting to a Higher Gear

in a

Natural Language System

by

Dr. Bozena Henisz Thompson
Dr. Frederick B. Thompson

Technical Report #4128

submitted to

1981 National Computer Conference
Chicago, Illinois
May 1981

Computer Science Department
California Institute of Technology
Pasadena, California 91125

Shifting to a higher gear in a natural language system

by BOZENA HENISZ THOMPSON

and

FREDERICK B. THOMPSON
California Institute of Technology
Pasadena, California

ABSTRACT

We have completed the development of the REL System, a system for communicating with the computer in natural language concerning a relational database. We have been using that system in a series of experiments on how people actually do communicate in solving an intellectual task. These experiments, together with our general experience with REL, and related work elsewhere, have led us to the specification and development of a new system, the POL (Problem Oriented Language) System. POL is an evolutionary extension of REL, preserving what has worked, and extending and adding new capabilities to meet observed needs. These improvements include more responsive diagnostics, handling of sentence fragments, inter knowledge base communications, and new facilities for building and extending the knowledge bases of users. This paper introduces POL.

INTRODUCTION

We have completed the development of the REL System, a system for communicating with the computer in natural language concerning a relational data base. We have been using that system in a series of experiments on how people actually do communicate in solving an intellectual task. These experiments, together with our general experience with REL, and related work elsewhere, have led us to the specification and development of a new system, the POL (Problem Oriented Language) System. POL is an evolutionary extension of REL, preserving what has worked, and extending and adding new capabilities to meet observed needs. This paper introduces POL.

KNOWLEDGE BASED SYSTEMS IN THE RAPIDLY CHANGING ENVIRONMENT

The continuing rapid increase in both the capability and availability of computers has raised the expectations of what they can do for us. In the near future they should be able to respond intelligently to directions we give to them in our own natural language. Intelligent response to natural language implies more than the understanding of the structure of language. It also implies knowledge of the meanings of the technical terms we use in dealing with the complex problems of our work domains, and the ability to use that knowledge in formulating responses. In communicating about technical matters, we make use of many facts and relationships that are tacitly understood. Thus the computer must not only have available to it a body of facts, a database, but knowledge of the relationships that tie that data together. We will refer to this wider body of knowledge as a "knowledge base."

A knowledge base concerning a given subject area contains the relevant data concerning that area; the notion of knowledge base incorporates and extends the notion of database. When one queries a database, one expects to get back only the raw data one asks for. Most database systems go somewhat beyond simple recitation of data that has been put into them, providing, for example, statistical reduction of that data. A knowledge base goes well beyond this, incorporating knowledge of the domain that it can use in digesting the query in conjunction with its data.

A simple example may be useful. Consider a system concerned with the loading of cargo ships. One can instruct it to put various items into the ship's cargo spaces. If the system is knowledgeable, it will be able to answer a query concerning the remaining area available in a given cargo space, for it will know that the remaining area is the total area less that occupied by shipments and that it can compute the area occupied by a shipment from the dimensions of each particular type of shipment, dimensions contained in its data.

Knowledge is closely associated with language. Certainly it is knowledge of the language that gives a system the capability to respond to natural language queries. In the above example it was knowledge of the term "remaining area" that allowed the system to give a useful reply concerning the remaining area of a cargo space after a variety of shipments had been placed there. Thus a knowledge base is a language-database package where the language component includes the semantics of the domain of application.

Knowledge base systems cover a wide range. We first characterize this range and then identify that area within this range where our interests lie. At the low end of this spectrum are the programming languages, such as Fortran, Pascal, and Conniver; they know nothing about the domain of the user. The high end of the spectrum is open ended, as there is no "most knowledgeable" system.

The most intelligent systems now are typified by the medical diagnostic systems such as MYCIN. In these systems, the physician who is not a specialist in a particular field of medicine can call up the computer, hold a highly technical dialogue concerning the history and symptoms of his patient, and expect to get diagnostic information reflecting the knowledge of the best specialists in the field. We point out several properties of such systems. The task of putting into a system the best knowledge available in the field is expensive in both time and resources. Further, the source of this knowledge is not the user, but experts removed in time and place. To make such systems economically viable, the domain must be stable, the technical terms of the domain widely known and unambiguous, the application must be widespread and important, and the expectation must be that the knowledge base will change only slowly with time.

In contrast to this type of knowledge domain, there is the knowledge base of the typical research team, management staff, or administrative office. A research team may be designing, constructing a prototype and testing a new device; the inventory control staff of a firm may be keeping track of and reordering a large variety of parts; a government agency may be administering contracts and evaluating proposals. In each of these cases the user is intimately involved in the maintenance of his or her knowledge base. It may appear that the structures of these respective knowledge bases are rather static; however, this is not the case. Indeed it is the constant shifting in structure of the knowledge base and its associated vocabulary that mark these organizations that must operate in a constantly changing environment. Not the least of these changes is in the personnel themselves, each with their own ways of doing things, ways which must be reflected in the base. These knowledge base systems are the properties of their users, and the principal tasks of their development and maintenance lies with their users.

REL and POL are designed as knowledge base systems for these rapidly changing environments.

In these systems there are two levels of change that are important. First, there is change by the users themselves. They must be able not only to modify the various data items in the knowledge base, but to extend and modify the structure of the base itself. They must be able to add definitions and other abbreviated means for extracting and manipulating the data, and by these means add their own tacit knowledge and expectations so that the computer will respond meaningfully and succinctly to their further queries.

The second level of change is at the application programmers' level. Preparing the knowledge base for a particular using community constitutes a major task. For a system like MYCIN, the clerical aspects of this task are dwarfed by the time resources of true experts. The mundane aspects of actually putting their knowledge into the computer can be relegated to far less costly resources. However, in the case of systems on which we are focusing, the "experts" are more local to the using group and are called upon far more frequently to prepare knowledgeable working environments to fit the newly arising tasks or circumstances confronting their user clients. The systems on which we are working must support both levels of change.

THE REL/POL KNOWLEDGE BASE SYSTEMS

In designing the REL and POL Systems, we sought to build into the system all aspects that would be common to all or most rapidly changing knowledge base environments. These include managing the input and output, data storage and retrieval, and processing of the language—parsing and semantic interpretation. They also provide facilities for creating and for extending and modifying a knowledge base, for adding and changing data and vocabulary, and for handling definitions. These capabilities are essentially identical in REL and POL. Since they are adequately covered in the REL documentation,[11,13] we will cover them only briefly here. Here are some of the main features of POL.

The core of a knowledge base system is its language processor. The POL language processor consists of four parts:

- the Preparser, which takes care of such tasks as line continuation, multiple blanks, recognizing whole numbers, and looking up all words in the lexicon;
- the Parser, which develops the parsing tree for the input by using a pattern matching algorithm in conjunction with the particular grammar table for the specific language involved;
- the Semantic Processor, which uses the parsing tree to compose the interpretive routines that are associated with the grammar rules;
- the Output Processor, which does a variety of final editing on the results of semantic processing to prepare the lines to be output.

Following is some information about the various techniques we use in each of these four steps of message processing.

The most interesting aspect of the Preparser is the lexicon processing method. We use a triple hashing technique. Given an identifier, or word, to look up in the lexicon, we hash it one character at a time into the first hash bit table of 2**14 bits. If the corresponding bit in this table is 0, then the segment so far hashed can not be the initial segment of a word in the lexicon, and we can stop. Otherwise we refer to the second hash bit table, again of 2**14 bits. If this corresponding bit is 0, then the segment hashed so far is not itself in the lexicon. Only on success here do we finally hash down to 64 hash buckets and search the appropriate bucket for identical key. This technique is in general very fast, since the hash bit tables can be kept in highspeed memory. Moreover, it results in a particularly fast, lexicon-driven spelling corrector.

There are today two parsing algorithms being used for grammar-driven natural language processing, the top down ATN parser, developed by Woods,[14] and the bottom up Powerful Parser of Kay.[6] Both are chart parsers in the sense of R. Kaplan,[4] and it is now known that their basic algorithms are

essentially identical (as shown by Papachristidis[7]). REL and POL use the Kay parser. We feel that the bottom up approach provides the basis for more useful diagnostics and for more useful information in handling sentence fragments such as ellipses, false starts and added information.

We do not order our grammar rules or multiple parsings, thus we get all possible parses of the input message. A distinctive feature is that ambiguities are handled internal to the semantic processor routine, and thus interpretive routines, associated with the grammar rules, do not have to be aware of the possibility of ambiguity. The semantic processor also expands definitions, including instantiation of variables. This again leaves the interpretive routines to deal only with local problems of the immediate phrase and its immediate constituents.

An interesting aspect of the Output Processor is that it distinguishes between diagnostic messages and substantive answers. Any interpretive routine, looking at its local context, can output a message and may mark it as a diagnostic. At the end of output processing, the Output Processor considers all messages, diagnostic and substantive alike. If there is at least one substantive response, all diagnostics are repressed. Remaining ambiguous messages are edited for redundancy. We give two illustrations of the use of this mechanism.

First, suppose the input message was "What is the rank of the radiation officer of the Alamo?" In the processing of this message, an interpretive routine will be called to determine the radiation officer of the Alamo; suppose it finds by reference to the data base that the Alamo does not have a radiation officer. This interpretive routine then issues the diagnostic message: "There is no radiation officer of the Alamo." and otherwise signals that it was unsuccessful, aborting the remainder of semantic processing. This meassage would then be output.

As a second example, suppose that the term "gross sales" was multiply defined, having two entries in the lexicon; one that resulted in summing the sales for a given product, the other summing the sales made by a given sales person. In answering the question "What is the gross sales of diodes?" the interpretive routine interpreting the construction "gross sales of diodes" would be called twice by the semantic processor, once for each definition of "gross sales." In the first case, it would issue a substantive response; in the second case it would issue a diagnostic: "Diodes is not a sales person." The Output Processor, seeing that a substantive response is forthcoming, would repress the diagnostic, and the user would get only the response he or she desired.

The data base organization underlying both REL and POL is the relational one. It is complete, in the technical sense of that term as used in relational data base theory. We have devoted a great deal of attention to optimization of data base algorithms, particularly in regard to access to peripheral storage. Indeed, efficiency of processing was a paramount concern throughout the development of REL. A principal objective of REL was to establish that fully operational, natural language, relational data base systems could be realized in the near term. A primary requirement therefore was good response time. Total throughput message processing time, on an IBM370/3032 computer and using a data base developed by others for testing just such systems (the Navy "blue" file), is averaging about four seconds. The total response time from input to output of the answer for the query

> What is the destination and cargo type of each ship whose port of departure was some Soviet port?

is less than 10 seconds.

THE REL EXPERIMENTS

We have learned a great deal from REL and, in particular, from the series of experiments on human-to-human and human-to-computer communication. The majority of these experiments involved a real life task of loading Navy cargo ships. The total time spent by subjects was over 50 hours, which yielded for final comparisons 20 face-to-face protocols, 11 terminal-to-terminal protocols, and 21 human-to-computer protocols, containing over 80,000 words. (See Thompson[12] for a complete report.)

It was found that in task-oriented situations the syntax of interaction is influenced in all modes by this context in the direction of simplification, resulting in short, simple sentences (averaging in all three modes about seven words). Users seek to maximize efficiency in solving the problem. When given a chance, in the human-to-computer mode, to use special devices facilitating the solution of the problem, they all resort to them.

In reporting on the analysis of these protocols, the term "message" refers to an utterance of one speaker; a "sentence" was required to contain both a noun phrase and a verb phrase, be confined to a single message and have substantial semantic cohesiveness. Parts of messages not also parts of sentences were identified as either "phatics" or "fragments." A phatic is any string whose function was to keep the channels of communication open, e.g., "okay," "I see," and—to the computer—"You lie." The following table presents some of the results of analysis (F-F = face to face, T-T = terminal to terminal, H-C = human to computer).

	F-F	T-T	H-C
Sentence length	6.8	6.1	7.8
Message length	9.5	10.3	7.
Fragment length	2.7	2.8	2.8
% of words in sentences	68.8	72.8	89.3
% of words in fragments	17.2	21.1	10.7
sentences per message	.96	1.22	.81
fragments per message	.59	.74	.19
phatics per message	1.1	.59	.04

The types of sentences used in human to computer communication is of considerable interest:

	Total	Percent
All sentences	882	100
Simple sentences, e.g., "List the decks of the Alamo."	651	73.8
Wh-type questions, e.g., "What are ships?"	658	75.0
Sentences with pronouns, e.g., "What is its length?"	30	3.4

	Total	Percent
Sentences with quantifiers, e.g., "List the class of each cargo."	101	11.4
Sentences with conjunctions, e.g., "List hatch width and hatch length of each deck of Alamo."	113	12.8
Sentences with relative clause, e.g., "List the ships that have water."	16	1.9

The dominance of simple sentences is striking. The reason is certainly not the lack of availability of complex sentences. We think that several factors account for this. The problem-solving situation influences the subject to work in a simple manner, often employing what we have termed special strategies, e.g., repetition of the same type of requests. Another reason is definitions. Once subjects introduce a definition whose right hand side is complex, they use it in subsequent messages, which are therefore short and simple. Another reason may be that subjects tend to be more formal in conversation with a computer. On the whole, one is forced to conclude that monotony of structure is the rule rather than the exception in human-computer communication.

Fragments compose a significant part of communication in all three modes. In an earlier paper on human dialogue in problem-solving situations, it was noted by Chapanis[1] on the basis of extensive experiments that "people do not naturally speak in sentences" and that "in general great unruliness characterizes communication." At first sight of the protocols one tends to confirm the impression. But a closer look and careful analysis reveals a considerable orderliness. Over the course of analysis of these protocols we have been led to classify fragments into eleven categories,[12] each suggesting corresponding procedures for processing them. We will briefly comment on three of these categories here.

Terse Question: e.g., "How about pyrotechnics?" "How many?", "Which ones?" Elliptical questions of this type should be handleable by computational means. Note that in handling pronouns, one looks in the preceding part of the protocol for a referent that can be substituted for the pronoun. The same techniques appear applicable in this case, when one seeks a referent that can be replaced or modified by the body of the elliptical expression.

Added Information: e.g., "What are the destinations of ships?... Soviet ships," "It doesn't say anything about weight.... Except for the crushables." The frequency of such added information suggests that the terminal keyboard should be kept open and additional input before start of output should be incorporated into message processing.

False Start: e.g., "Do ships What Ships carry ammunition?" Although computer terminals usually provide a convenient means for deleting an input and starting over, they are not always used. Ways of intercepting these occurrences should not be difficult to incorporate.

Important insights into ways to improve the habitability of human-computer systems was gained by analysis of the errors that occurred in the protocols. In the human to computer protocols there were 446 errors. A breakdown into eight categories is given by the following table:

	Total	Percent
Vocabulary	161	36.1
Punctuation	72	16.1
Syntax	62	13.9
Spelling	61	13.6
Transmission	32	7.2
Definition format	30	6.7
Improper response to prompt	16	3.6
"Bug," error in system	12	2.7

THE EXTENDED CAPABILITIES OF POL

Although the core of the POL System is similar to REL, it also embodies several significant extensions. In the first place, in the development of REL, very little attention was given to problems of habitability, a property of human-computer systems sometimes referred to as "friendliness." For example, there was no spelling corrector and substantive diagnostics were particularly weak. Analysis of the experimental protocols has indicated concrete directions for improvements. To illustrate, you will notice in the table immediately above that the greatest source of errors was the use of words that were not in the vocabulary of the particular application. REL did not identify those words nor even indicate the nature of the problem. The user would often try paraphrases using the same missing word before sensing the source of his or her difficulty. POL immediately identifies such words, as in the following example:

User: What is the usual use of the super deck of the Alamo?
REL: Input error: please re-enter request.
POL: The following word is not in the vocabulary: usual

Other illustrations will be given below.

In REL, we were preoccupied with the core of the system and with REL English and its relational database system. Capabilities beyond these core concerns are, however, also important, in particular those facilitating inter database communication, the distributed database problem, and also capabilities to facilitate the building of specialized user applications. We have had two fine doctoral dissertations which have introduced significant improvements in these two areas. These have added new dimensions to POL beyond REL, as described below. Finally, we are incorporating several methods for augmenting natural language for more efficient human-computer communications. These topics will be discussed in the following four subsections.

All of these topics, however, share a common theme. Problems of natural language processing and efficient database processing are now well in hand and quite adequate for implementation of practical systems with good response times. The REL System gives quite convincing evidence of this. The next stage in improving human-computer communication will be through a better understanding of how users will actually behave as they productively interact with the computer in ways that are natural for them. The systems that we can now provide are unfriendly, they are too sensitive to trivial errors, too pedantic in the messages they are willing to understand, they make inadequate provision for building the user's knowledge base and vocabulary into the system, they do not offer ade-

quate protection from catastrophic mishap, and they do not provide for normal kinds of inter knowledge base communication. The user is in a position where s/he realizes s/he is doing something wrong but has no idea how to proceed. The thrust of our work in extending REL to POL is to relieve these unfriendly aspects of using the computer and to replace them by a habitable working environment.

Diagnostics

The user inadvertently makes mistakes. Even when the user's message is quite correct and the computer formulates a correct response, that response may still be confusing. In these cases, the computer needs to provide a more useful response. Diagnostics are of two kinds: semantic, which must be accounted for in the interpretive routines of the specific language involved; and syntactic, which can largely be taken care of in the language processor. We give several illustrations of POL diagnostic techniques.

The first step in correcting the diagnostic deficiencies in REL was to maintain with each phrase recognized by the parser its underlying literal string so that this string is available to both the system and the interpretive routines for framing meaningful responses. Thus for example: "San Diego ships" may be found to be ambiguous, referring both to those ships located in San Diego or to those ships whose home port is San Diego. REL provides these two interpretations but does not provide the user with any indication of the source of the ambiguity. In POL, the interpretation routine that is looking for San Diego ships discovers the ambiguity and, having the string "San Diego ships" available, is able to tag them. Thus:

```
›What are the destinations of San Diego ships?
 Ambiguous:
(1) San Diego (location) ships
      New York
      Tokyo
(2) San Diego (home port) ships
      Naples
      San Diego
```

Using this technique we have been able to incorporate many of the ideas concerning diagnostics expressed by S.J. Kaplan,[5] as illustrated by the following example:

```
›What is the square foot capacity of the super deck of the
 Alamo?
 The Alamo does not have a super deck.
```

A technique we are widely employing in POL, facilitated by an "evaluate" procedure that can be called from an interpretive routine, is to check out possible corrections by calling the parser and semantic processor, and using these evaluations in deciding on a response. When corrections are made, we signal the interpretation by echoing. If the number of potential corrections exceeds some small number at any given point, then any attempt at correction is discontinued and a less informative diagnostic is given. Thus the query:

```
›Who is the commander of the Alemo?
```

might, as the case may be, yield any one of the following responses:

- Capt H. Smith
- The Alemo does not have a commander.
- Spelling corrected: "Alamo" for "Alemo" Capt H. Smith
- Ambiguous:
  (1) Spelling corrected: "Alamo" for "Alemo" Capt H. Smith
  (2) Spelling corrected: "Alimo" for "Alemo" Capt K. Jones
- The following word is not in the vocabulary: Alemo

We are augmenting the POL English grammar with many rules which recognize what are, strictly speaking, ungrammatical forms. The interpretive routines associated with these rules will have checks and safeguards built into them, again with use of evaluate, and will echo the corrected form before giving the response. For example:

```
›width, length of M74 truck
 Width and length of M74 truck?
 96 180
›What is the destination of the Alamo.
 What is the destination of the Alamo?
 London
```

These methods do not, however, handle the problem where the input message does not completely parse because it does not strictly adhere to the grammar. How to frame truly helpful responses in these cases is a difficult research question. The work of Sondheimer and Weischedel[10] provides important directions.

Fragments and Pronouns

There has been some excellent work by Grosz on identifying and following the focus of a dialogue[2] and by Sidner on using this notion of focus in identifying the referent for pronouns and anaphoric expressions.[9] Using these and other linguistic analyses of pronouns, Roach has developed a considerably improved method for handling pronouns in POL.[8] We hope to be able to apply these same techniques to the processing of terse questions and added information. The discussion above of the experimental protocols indicates other areas where we plan to strengthen POL's ability to handle a variety of sentence fragments.

Inter-Knowledge-Base Relationships

A research or management staff has not one but many knowledge bases. For example, in a manufacturing firm there would be a personnel base, an inventory control base, a purchasing base, a customer base, etc. Each of these has its own practices for updating and deleting, and each is the responsibility of a different part of the firm. In the past, attempts have been made to consolidate all of these into a single corporate database, but this has been found to be unsatisfactory. Similarly, in a research staff or project team, there may be a variety of knowledge bases, each documenting a

variation of a basic experimental design. In these rapidly changing environments, where members of the staff are involved in modifying and extending their knowledge bases, backup copies must be maintained, contingencies examined, alternative plans evaluated—requiring several working copies of a common knowledge base.

POL provides the capability for creating and maintaining many knowledge bases within the same knowledge base system. Further, these several knowledge bases may be interrelated in a variety of different ways. One such relationship is "basing." One knowledge base, say B, can be "based" upon another, say A. Once this basing operation has taken place, all of the information available in A is automatically available in B, and any subsequent changes in A are automatically reflected in B. Changes in B, on the other hand, will not affect A at all. A knowledge base may be based upon several other knowledge bases, and many knowledge bases may be based upon a single one.

To see how these capabilities might be used, suppose the accounts in a firm were divided into three groups, aa, bb and cc, each the responsibility of a separate desk in the accounting section. Then three knowledge bases, AA, BB and CC would be created, each owned by its respective desk which would preserve the rights to modify it. The general accounting base, say GG, would be based on all three. The higher management base MM and the home office base HH would be based on GG. A staff office, studying the effects of a change in pricing policy, could also base their study base SS on GG, making what ever changes they were interested in in SS but not affecting GG at all. These capabilities reflect the doctoral dissertation of Yu.[15]

Metalanguage

POL incorporates a significant new notion of metalanguage, the knowledge-base environment for the application programmer. The bottom "knowledge base" of the POL System is POL English, containing the syntax of a subset of natural English, a mathematics package and the function words of English, e.g., "have," "and," "which" etc. All other knowledge bases for users are based upon POL English in the sense described in the last paragraph. There is another basic "knowledge base," namely MetaEnglish. It contains a variety of constructions, including (1) all of Pascal, (2) the capability of writing in succinct form new grammar rules for a given target language and their associated interpretive routines, (3) a similar capability to extend the metalanguage itself with either new procedures or macros, (4) a variety of useful utility procedures for dealing with the relational database, effectively handling input and output, literals, and the evaluate function, and (5) a compiler/linker that is able to relate this self-extended Pascal and its associated syntax. POL English and MetaEnglish are associated with each other.

When a knowledge base, say AA, is based upon POL English, a parallel knowledge base, MetaAA is also created, based on MetaEnglish; AA and MetaAA are associated with each other. An application programmer using MetaAA adds grammar rules and associated interpretive routines to her user's knowledge base AA. In doing this, she may expeditiously add new utilities to her own "knowledge base," namely MetaAA. Obviously she can use the utilities and procedures of MetaEnglish, since her MetaAA is based on MetaEnglish. In fact the whole basing structure of hierarchically related knowledge bases is strictly paralleled by the associated Metabasing structure.

This base/meta-base apparatus is designed to facilitate the building of specialized knowledge bases that extend and specialize more general capabilities to the needs of users. This aspect of POL reflects the doctoral thesis of Hess.[3]

IMPLEMENTATION

REL was implemented on a large IBM-370/3032 computer. POL, on the other hand, is being implemented on a desk top computer, the Hewlett-Packard HP-9845B with a 50 megabyte disk. POL is written in Pascal.

At the present time all of the central parts of the system—preparser, parser, semantic processor, output processor, definition handling, paging, list processor—have been completed. Basing has been completed and we are in the last stages of completion of the metalanguage. Much of POL English has been completed, including the optimized utilities for managing the relational data base.

Of course there is much left to be done. But the most important work on POL will start when the system is completed, response times have been brought under control and the first applications implemented. For then we can observe, in an environment much closer to being friendly and conducive to natural propensities, just how professionals with a job to do will communicate with computers. That is the exciting moment, for as we know from our REL experience, reality reveals the directions to truly more responsive systems.

REFERENCES

1. Chapanis, A., "Interactive Human Communication," *Scientific American*, April 1975, pp. 39-42.
2. Grosz, B.J., *The Representation and Use of Focus in Dialogue Understanding*, SRI International, Tech. Note 151, 1977.
3. Hess, G.D., *A Software Design System*, Ph.D. Dissertation, Calif. Inst. of Tech., 1980.
4. Kaplan, R.M., "A General Syntactic Processor," Rustin, R.(ed.), *Natural Language Processing*, Algorithmics Press, New York, 1973, pp. 193-241.
5. Kaplan, S.J., *Cooperative Responses from a Portable Natural Language Data Base Query System*, Ph.D. Dissertation, Univ. of Penn., 1979.
6. Kay, M., *Experiments with a Powerful Parser*, The Rand Corp., Santa Monica, Calif., RM-5452-PR, 1967.
7. Papachristidis, A.C., *Comparison of ATN, Kay and Related Parsing Algorithms*, Master's Thesis, Calif. Inst. of Tech., 1980.
8. Roach, K., *Pronouns*, Master's Thesis, Calif. Inst. of Tech., 1980.
9. Sidner, C., *Towards a Computational Theory of Definite Anaphor Comprehension in English Discourse*, Mass. Inst. of Tech., Tech. Report 537, 1979.
10. Sondheimer, N. and R.M. Weischedel, "A Rule-Based Approach to Ill-Formed Input," *Proc. 8th Intern. Conf. on Comp. Ling.*, Tokyo, 1980, pp. 46-53.
11. Thompson, B.H., *REL English for the User*, Calif. Inst. of Tech., 1978.
12. Thompson, B.H., "Linguistic Analysis of Natural Language Communication with Computers," *Proc. 8th Intern. Conf. on Comp. Ling.*, Tokyo, 1980, pp. 190-201.
13. Thompson, B.H. and F.B. Thompson, "Rapidly Extendable Natural Language," *Proc. 1978 Nat. Conf. of ACM*, pp. 173-182.
14. Woods, W.A., "Transition Network Grammars for Natural Language Analysis," *Comm. of ACM*, vol. 13, (1970), pp. 591-606.
15. Yu, K.I., *Communicative Databases*, Ph.D. Dissertation, Calif. Inst. of Tech., 1980.