American Journal of Computational Linguistics *Microfiche 21 : 2*

RECENT COMPUTER SCIENCE RESEARCH IN NATURAL LANGUAGE PROCESSING

ALLEN KLINGER
Computer Science Department
School of Engineering and Applied Science
University of California, Los Angeles

ABSTRACT

The machine translation problem has recently been replaced by much narrower goals and computer processing of language has become part of artificial intelligence (AI), speech recognition, and structural pattern recognition. These are each specialized computer science research fields with distinct objectives and assumptions. The narrower goals involve making it possible for a computer user to employ a near natural-language mode for problem-solving, information retrieval, and other applications. Natural computer responses have also been created and a special term, "understanding", has been used to describe the resulting computer-human dialogues. The purpose of this paper is to survey these recent developments to make the AI literature accessible to researchers mainly interested in computation on written text or spoken language.

1. INTRODUCTION

The computer literature discussed in this paper uses several linguistic terms in special ways, when there is a possibility of confusion, quotation marks will be used to identify technical terms in computer science. The term "understanding" is frequently used as a synonym for "the addition of logical relationships or semantics to syntactic processing". This use is substantially narrower than the word's implicit association with "human behavior implemented by computer" the narrower use is introduced as a neutral reference point. The question of whether a computer porgram can operate in a human-like way is central to artificial intelligence. "Do current 'understanding' program systems show how extended human-like capability can be implemented using computers?" is a related pragmatic question Initially this investigation sought to examine whether programs which "understand" language in the stipulated narrow sense are prototypes which could lead to expanded capability. Unfortunately, "language understanding" and its special subtopic "speeeh understanding" are insufficiently developed to permit profitable discussion of the original question Hence an operational approach to the recent literature is taken here. This paper outlines how "language understanding" research has evolved and identifies key elements of program organization used to achieve limited computer "understanding".

2. LEVEL AND DOMAIN

Current AI programs for language processing are organized by level and restricted to specified domains. This section presents those ideas and comments on the limitations that they entail.

Three principal levels of language-processing software are

1. "Lexical" (allowed vocabulary)
2. "Syntactic" (allowed phrases or sentences)
3 "Semantic" (allowed meanings)

In practice all these levels must operate many times for the computer to interpret even a small portion, say two words, of restricted natural-language input. Programs that perform operations on each level are, respectively,

1. Word in a table?
2. Word string acceptable grammatically?
3. Word string acceptable logically?

A program to detect "meaning" (logical consequences of word interpretations) must also perform grammatical operations for certain words to determine a part of speech (noun, verb, adjective, etc.) One method makes a tentative assignment, parses, then tests for plausibility via consistency with known facts. To reduce the complexity of this task, the designer limits the subset of language allower or the "world" (i.e. the subject) discussed. The word "domain" sums up this concept, other terms for "restricted domain" are "limited scope of discourse", "narrow problem domain", and "restricted English framework"

The limitation of vocabulary or context constrains the lexicon and semantics of the "language". The trend in the design of software for "natural-language understanding" is to deal with (a) a specialized vocabulary, and (b) a particular context or set of allowed interpretations in order to reduce processing time. Although computing results for several highly specialized problems [e.g. 7, 23] are impressive examples of language processing in restricted domains, they do not answer several key concerns.

1. Do specialized vocabularies have sufficient complexity to warrant comparison with true natural language?
2. Are current "understanding" programs, organized by level and using domain restriction, extendable to true natural language?

The realities are severe. Syntactic processing is interdependent with meaning and involves the allowed logical relationships among words in the lexicon. Most natural-language software is highly developed at the "syntactic" level However, the number of times the "syntactic" level must be entered can grow explosively as the "naturalness" of the language to be processed increases. Success on artificial domains cannot imply a great deal about processing truly natural language.

3. PROGRAM SYSTEMS

The systems cited in this section answer questions, perform commands, or conduct dialogues.

Programs that enable a user to execute a task via computer in an on-line mode are generally called "interactive" Some systems are so rich in their language-processing capability that they are called "conversational" Systems that have complicated capabilities and can reply with a sophisticated response to an inquiry are called "question answering". The survey [1] discusses two "conversational" programs ELIZA [2, 3] and STUDENT [4], which answers questions regarding algebraic word problems. SIR [5] answers questions about logic. Both [4] and [5] appear in [6], the introduction there provides a general discussion of "semantic information and computer programs involving "semantics"

The "question-answering" program systems described in [2-5] were sophisticated mainly in methods of solving a problem or determining a response to a statement. Other systems have emphasized the retrieval of facts encoded in English. The "blocks-world" system described in [7] contrasts with these in that it has sophisticated language-processing capability It infers antecedents of pronouns and resolves ambiguities in input word strings regarding blocks on a table. The distinction between "interactive", "conversational", and "question-answering" is less important when the blocks-world is the domain. The computer-science contribution is a program to interact with the domain as if it could "understand" the input, in the sense that it takes the proper action even when the input is somewhat ambiguous. To resolve ambiguities the program refers to existing relationships among the blocks.

The effect of [7] was to provide a sophisticated example of computer "understanding" which led to attempts to apply similar principles to speech inputs. (More detail on parallel developments in speech processing is presented later.)

The early "language-understanding" systems, BASEBALL [9], ELIZA, and STUDENT, were based on two special formats: one to represent the knowledge they store and one to find meaning in the English input. They discard all input information which cannot be transformed for internal storage. The comparison of ELIZA and STUDENT in [1] is with regard to the degree of "understanding" ELIZA responds either by transforming the input sentence (essentially mimicry) following isolation of a key word or by using a prestored content-free remark. STUDENT translates natural-language "descriptions of algebraic equations, ... proceeds to identify the unknowns involved and the relationships which hold between them, and (obtains and solves) a set of equations" [1, p 85]. Hence ELIZA "understands" only a few key words; it transforms these words via a sentence-reassembly rule, discards other parts of the sentence, and adds stock phrases to create the response. STUDENT solves the underlying algebraic problem--it "understands" in that it "answers questions based on information contained in the input" [4, p. 135]. ELIZA responds but does not "understand", since the reply has little to do with the information in the input sentence, but rather serves to keep the person in a dialogue.

Programs with an ability to spout back similar to ELIZA's usually store a body of text and an indexing scheme to it. This approach has obvious limitations and was replaced by systems that use a formal representation to store limited logical concepts associated with the text. One of them is SIR, which can deduce set relationships among objects described by natural language. SIR is designed to meet the requirement that "in addition to echoing, upon request, the facts it has been given, a machine which 'understands' must be able to recognize the logical implications of those facts. It also must be able to identify (from a large data store) facts which are relevant to a particular question" [5].

Limited-logic systems are important because they provide methods to represent complex facts encoded in English-language statements so that the facts can be used by computer programs or accessed by a person who did not input the original textual statement of the fact. Such a second user may employ a completely different form of language encoding. Programs of this sort include DEACON [10, 11] and the early version of CONVERSE [12]. The former could "handle time questions" and used

> a bottom-up analysis method which allowed questions to be nested. For example, the question "Who is the commander of the battalion at Fort Fubar?" was handled by first internally answering the question "What battalion is at Fort Fubar?" The answer was then substituted directly into the original question to make it "Who is the commander of the 69th battalion?" which the system then answered. [7, p. 37]

CONVERSE contained provisions for allwing even more complex forms of input questions (Recent versions are described in [13-15].)

Deductive systems can be divided into general systems which add a first-order predicate-calculus theorem-proving capability to limited-logic systems to improve the complexity of the facts they can "infer", and procedural systems which enable other computations to obtain complex information The theorem-proving capability is designed to work from a group of logical statements given as input (or statements consistent with these input statements) However, facts INCONSISTENT with the original statements cannot always be detected and deductive systems quickly become impractical as the number of input statements (elementary facts, axioms) becomes large [6, 7, 16], since the time to obtain a proof grows to an impractical length. Special programming languages (e.g. QA4 [17, 18], PLANNER [20, 21]), have added strategy capabilities and better methods of problem representation to reduce computing time to practical values

> QA4 (seeks) to develop natural, intuitive representations of problems and problem-solving programs. (The user can) blend ... procedural and declarative information that includes explicit instructions, intuitive advice, and semantic definitions. [17]

However, there is currently no body of evidence regarding the effectiveness of the programs written in this programming language or related ones on problem-solving tasks in general

or "language understanding" in particular. There is a need for experimental evaluation of the strategies that the programming language permits for "language understanding" problems.

Procedural deductive systems facilitate the augmentation of an existing store of complex information. Usually systems require a new set of subprograms to deal with new data:

> each change in a subprogram may affect more of the other subprograms. The structure grows more awkward and difficult to generalize. ... Finally, the system may become too unwieldy for further experimentation. [5, p. 91]

In procedural systems the software is somewhat modular In 19 "semantic primitives" were assumed to exist as LISP subroutines. PLANNER [20] allows complex information to be expressed as procedures without requiring user involvement with the details of interaction among procedures (but [21] reports some second thoughts).

The work of many other groups could be added to this survey. Recent work on REL, building on on [10, 11] is reported in [36, 37]; [24, 25] are relevant collections; and [26] is a survey paper.

4. DEDUCTION

In all of the program systems described thus far, "language understanding" depends on the "deductive capabilities" of the

---

*Some experiments on problem-solving effectiveness of special programming languages in another context appear in [22].

program, that is, its ability to "infer" facts and relationships from given statements. In some cases deduction involves discerning structure in a set of facts and relationships. This section describes how "understanding" programs themselves are structured and how that structure limits their capability for general deduction.

Theorem-proving programs use an inference rule illustrated in [23 p. 61] to deduce new knowledge. A formal succession of logical steps called resolutions leads to the new fact. The example there begins with P1 - P4 given:

P1 if x is part of v, and if v is part of y, then x is part of y;
P2 a finger is part of a hand;
P3 a hand is part of an arm;
P4 an arm is part of a man

A proof that

P9 a finger is part of a man

is derived by steps, such as combining P1 and P2 to get

P6 if a hand is part of y, then a finger is part of y

Unfortunately, it is easy to move outside the domain where the computer can make useful deductions, and the formal resolution process is extremely lengthy and thus prohibitively costly in computer time. In [31, 32] it is shown that some statements ("who did not write ---?") are unanswerable and that there is no algorithm which can detect whether a question stated in a zero-one logical form can be answered. Hence

theorem proving is not e-sential to "deduction" and "understanding" systems, natural or artificial, must rely on other techniques, e.g., outside information such as knowledge about the domain.

In most "understanding" programs, information on a primitive level of processing can be inaccurate; for example, the identification of a sound string "blew" can be inaccurately "blue" Subsequent processing levels combine identified primitives. If parts of speech are concerned, the level is syntactic; if meaning is involved, "semantic"; if domain is involved, the level is that of the "world". Each level can be an aid in a deductive process, leading to "understanding" an input segment of language. Programs NOW EXIST which operationally satisfy most of the following points concerning "understanding" in narrow domains (emphasis has been added)

> Perhaps the most important criterion for understanding a language is the ability TO RELATE THE INFORMATION CONTAINED in a sentence TO KNOWLEDGE PREVIOUSLY ACQUIRED. This IMPLIES HAVING SOME KIND OF MEMORY STRUCTURE IN WHICH THE INTERRELATIONSHIPS OF VARIOUS PIECES OF KNOWLEDGE ARE STORED AND INTO WHICH NEW INFORMATION MAY BE FITTED... The memory structure in these programs may be regarded as semantic, cognitive, or conceptual structures...these programs can make statements or answer questions based not only on the individual statements they were previously told, but also on THOSE INTERRELATIONSHIPS BETWEEN CONCEPTS that were built up from separate sentences as information was incorporated into the structure...

> THE MEANINGS OF THE TERMS STORED IN MEMORY ARE PRECISELY THE TOTALITY OF THE RELATIONSHIPS THEY HAVE WITH OTHER TERMS IN THE MEMORY. [28 pp. 3-4]

This has been accomplished through clever (and lengthy) computer programming, and by taking advantage of structure inherent in special problem domains such as stacking blocks on a table, moving chess pieces, and retrieving facts about a large naval organization.

Program systems for understanding begin with a "front end": a portion designed to transform language input into a computer representation. The representation may be as simple as a character-by-character encoding of alphabetic, space marker, and punctuation elements. However, a complex "front end" could involve word and phrase detection and encoding. The usual computer science term for a computer representation is "data structure" [27] and there are many types. The language processing program DEACON used ring structures [11], a representation frequently used to store queues. In principle a data structure can represent involved associations, but in practice simple order or ancestor relationships predominate Completely different and far more complex types of structure are inherent in natural language. For example, from [28]

> "The professors signed a petition." is not true.

has for valid interpretations:

(a) The professors DIDN'T sign a petition.
(b) THE PROFESSORS didn't sign a petition.
(c) The professors didn't sign a PETITION.
(d) The professors didn't SIGN a petition.

Iterative substitution of alternatives to deduce overall meaning yields cumbersome processing, especially when there are nested uncertainties. The recursive properties associated with the data structure term "list" [27] are not easily adapted to multiple meanings. Hence, representing linguistic data for computation is an open and fundamental research problem. Nevertheless, the programs which deduce facts from language do so without a clear best technique for computer representation. To do this, restrictions on the language implicit in the input domain are used, and repeated processing by level (lexical, syntactic, semantic) is used in the absence of an efficient representation language. Data structures that facilitate following the language structure are needed Existing programs provide special solutions to the problems of deductive processing in narrow language domains While these programs are not a general breakthrough in representing language data for computation, they demonstrate that current programming techniques enable a useful "understanding" capability Furthermore, there is a real potential for use of the "understanding" in an interactive mode to facilitate use of computers by nonspecialists and to tap the more sophisticated human understanding capabilities

5 INTERACTION

Research and computer program development designed to store multitudes of facts so that they can be accessed [29] or combined [30] and "understood (see pp. 3-10 in [30]) in

linguistic form (see pp. 11-17 of [30]) is highly relevant to recent research programs in text and speech understanding. When such a system is used a user might fail to get a fact or relationship because the natural-language subset chosen to represent his question was too righ--i.e., it includes a complex set of logical relationships not in the computer. Thus a block could result in a human-computer dialogue if the program has no logical connection between "garage" and "car" but only between "garage" and "house" (the program replies "OK" or "???" to user input sentences)

```
I LIKE CHEVROLETS.
OK
CHEVROLETS ARE ECONOMICAL.
OK
MY HOUSE HAS A LARGE GARAGE.
OK
I CAN GET TWO IN
???
```

The computer failed to "understand" that there was no change of discourse subject. This is an example of a "semantic" failure which could be overcome by interaction. That is, the human user would need to input one more meaning or association of a valid word so that computer "understanding" may be achieved. Syntactic blocks may also occur. M. Denicoff pointed out that in [7] 172 different syntactic features were used for a situation where there are no statements with psychological content and no use of simile. If the psychological meanings are added as in [38], these features would not be

enough to describe all the possible meanings of a text drawn from a less artificial source. Indeed, a key problem which formal grammars seem ill-suited for is the reality that many contexts may be simultaneously valid: multiple meanings give natural-language communication the richness of overtones and subtleties--poetry carries this to an extreme.

The above dialogue on "Chevrolets" is an example of what Carbonell [39, p. 194] called "mixed-initiative discourse". This important aspect of interaction is considered in the LISP program DWIM ("Do What I Mean"), which is a useful working tool for text-input error correction precisely because it "understands" the user's characteristics. (For example, typical spelling errors.) This is discussed by Teitelman [40, 41, 42]

> A great deal of effort has been put into making DWIM "smart". Experience with perhaps a dozen different users indicates we have been very successful: DWIM seldom fails to correct an error the user feels it should have, and almost never mistakenly corrects an error. [40, p. 11]

Another limited-discourse interactive program [43] facilitates introduction of expert knowledge on chess. The program uses search with a maximum look-ahead depth of 20 and has backtracking capability; both syntactic and semantic knowledge is incorporated. By grouping similar board positions (i.e., all involving a piece on cell 1, all involving a queen move), it imposes semantic organization on the vast files to be searched and improves syntactic processing speed

6. SPEECH

Publication of [44], which coined the term "speech understanding", initiated the natural next step toward use of the computer's "understanding" capability. The goal of easy interaction with the computer becomes more exciting with speech as input medium. Systems to recognize both text and speech have used syntax and context [45, 46], but [47] added a comprehensive approach using multiple processing levels to resolve ambiguities. In the direct successors of this work [8, 49], the same process of partial acceptance of primitive elements (phonemic candidates from digitized acoustic data) followed by lexical, syntactic, and semantic processing to rank alternatives has shown significant success. Reddy (in a Carnegie-Mellon University film on the Hearsay System) states hat on 144 connected utterances, involving 676 words, obtained from 5 speakers, performing 4 tasks (chess, news retrieval, medical diagnosis, and desk calculator use), requiring 28 to 76-word vocabularies, the computer program recognition, in terms of words spotted and identified correctly, was

a. 89% with all sources of knowledge
b. 67% without use of semantic knowledge
c. 44% without use of syntactic or semantic knowledge

These results were obtained in October 1973, and have been improved since [50]. However, a key limitation of this form of computer speech "understanding" is response rate. Reddy

estimated that the third word-accuracy figure (without use of syntactic and semantic knowledge) would have to be in excess of 90% to allow the program to achieve a near-human response speed.

The nature of computer "understanding" programs leads to problems of combinatoric explosion in number of alternatives and this lessens the usefulness of multilevel program organization (acoustic-phonetic, lexical, syntactic, semantic, domain, and user interactions) as much in speech processing as in text processing. Prototype speech "understanding" systems have been build [49, 50] and newer acoustic-phonetic and syntactic techniques have been incorporated into this work [49, 51, 52], yet it seems clear that the development of theory in prosody and grammar cannot provide a breakthrough to escape the combinatoric explosion. The reason is that the search of parse trees and the use of semantics (look up related words) depend on a single context--both take geometrically increasing amounts of computing time as the number of contexts grows. Furthermore, this increase in time is added onto that which occurs when the size of lexicon is expanded. As words are added, the number of trees that can be produced by the grammar's rewriting rules in an attempt to "recognize" a string expands rapidly. Hence in speech as in text processing, "understanding" exists via computer yet it is not likely to lead to machine processing of truly natural language. Indeed the artificiality of speech "understanding" by computer is

even greater than that of text input. The "moon rocks" text system [33, 35] used a vocabulary of 3500 words, while the speech "understanding" version based on it [51] used only 250 words.

The COMMERCIAL AVAILABILITY of systems that recognize isolated words with 98.5% accuracy [53]* and the need for a rapid human-computer input interface [54] promise that the last word has not been spoken on "understanding". Research and development on language handling systems is continuing in the hope of achieving useful "understanding". Indeed, Stanford Research Institute's Artificial Intelligence Center is basing its current work on the just-mentioned isolated-word recognizer. It is likely that useful developments will occur where language, and probably spoken-language, "understanding" will be exhibited. These developments will occur through careful design of tasks and use of advances in computer technology However, the general problem of machine "understanding" of natural language--whether text or speech--is not likely to be aided by these developments.

7 CONCLUSIONS

A large body of research in computer science is devoted to language processing. A survey of the program systems that

---

*Threshold Technology Inc. has sold such a system to several users. Their VIP-100 includes a minicomputer dedicated to the recognition task; there are other isolated-word systems [54]

have been reported shows that two main goals have emerged:

1. To enable "intelligent" processing by the computer ("artificial intelligence")
2. To produce a more useful way to access data and solve problems ("man-machine interaction")

Techniques in artificial intelligence and speech recognition have been developed to the extent that prototype computer program systems which exhibit "understanding" have been developed for highly limited contexts. To extend these programs to larger subsets of natural language poses problems, it is unlikely that any of the research directions currently being explored will of themselves "solve" the "natural language problem". (The techniques include, but are not limited to, further developments in artificial intelligence programming languages [17, 18 20, 21, 55]; refinements in theories of grammar; improved deductive ability, possibly by better theorem-proving techniques; and the introduction of stress-related features in the encoding of speech [52]. A useful collection of language models appears in [56].) Nevertheless, prototype systems for "understanding" both text and speech are useful achievements of engineering, and spoken entry of data by humans to computers is beginning to be established by isolated-word recognizers which use a minicomputer dedicated to the task. A multiplicity of purposes beyond this simple but practical task of data entry are mentioned briefly in the

foregoing discussion of "interaction". Developments along the many diverse paths indicated under that heading are likely to be rapid in the future as practical "understanding" of subsets of language becomes part of computer technology For another view of the evolution of that process, see [57].

REFERENCES

1. Nievergelt, J., and J. C. Farrar, "What Machines Can and Cannot Do," *Computing Surveys, 4*, June 1972, 81-96.

2. Weizenbaum, J., "ELIZA--A Computer Program for the Study of Natural Language Communication Between Man and Machine," *Comm. ACM 9*, January 1966, 36-45.

3. Weizenbaum, J., "Contextual Understanding by Computers," *Comm. ACM 10*, August 1967, 474-480.

4. Bobrow, D. G., "Natural Language Input for a Computer Problem Solving System," in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, Mass., 1968, 135-215

5. Raphael, B., "SIR: Semantic Information Retrieval," in M. Minsky (ed.), *Semantic Information Processing*, MIT, Press, Cambridge, Mass., 1968, 33-134, 256-266.

6. Minsky, M. (ed.), *Semantic Information Processing*, MIT Press, Cambridge, Mass., 1968.

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

8. Plath, W., "Restricted English as a User Language," IBM T. J. Watson Research Center, Yorktown Heights, New York, 1972.

9. Green, P. F., A. K. Wolf, C. Clomsky, and K. Laugherty, "BASEBALL. An Automatic Question-Answer," in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, 1963.

10. Thompson, F. B., "English for the Computer," *Proc. FJCC*, Spartan, New York, 1968, 349-356.

11. Craig, J. A., S. Berezner, H. Carney, and C. Longyear, "DEACON: Direct English Access and Control," *Proc. FJCC*, Spartan, New York, 1968, 365-380.

12. Kellogg, C., "A Natural Language Compiler for On-Line Data Management," *Proc. FJCC*, Spartan, New York, 1968, 473-492.

13. Travis, L., C. Kellogg, P. Klahr, *Inferential Question-Answering: Extending Converse*, System Development Corporation, SP-3679, January 31, 1973.

14. Kellogg, C. H., J. Burger, T. Diller, and K. Fogt, "The CONVERSE Natural Language Data Management System: Current Status and Plans," in J Minker and S. Rosenfeld (eds.), *Proc. Symp. Information Storage and Retrieval*, University of Maryland, College Park, April 1971, 33-46.

15. Kellogg, C. A., *Question-Answering in the Converse System*, System Development Corporation, TM 5015, October 1971.

16. Nilsson, N. J., *Problem-Solving Methods in Artificial Intelligence* McGraw-Hill, New York, 1971.

17. Rulifson, J. F., R. J. Waldinger, and J. A. Derksen, "A Language for Writing Problem-Solving Programs," *Proc. IFIP Congr. 1971* (presented at Ljubljana, Yugoslavia, August 1971).

18. Rulifson, J. F., *QA4 Programming Concepts*, Stanford Research Institute, Artificial Intelligence Group, Technical Note 60, August 1971.

19. Woods, W. A., "Procedural Semantics for a Question-Answering Machine," *Proc. FJCC*, Spartan, New York, 1968, 457-471.

20. Hewitt, C., "A Language for Theorems in Robots," *Proc. Int. Joint Conf. Artificial Intelligence*, Washington, D.C., 1969, 295-301.

21. Sussman, G. J., and D. V. McDermott, "From PLANNER to CONNIVER--A Genetic Approach" ("Why Conniving is Better Than Planning"), *Proc. 1972 FJCC*, AFIPS, Vol. 41, Part II, 1171-1179.

22. Fikes, R. E. "Monitored Execution of Robot Plans Produced by STRIPS," *Proc. IFIP Congr. 1971* (presented at Ljubljana, Yugoslavia, August 1971). Also see R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence, 2*, 1971, 189-208.

23. Slagle, J. R., *Artificial Intelligence: The Heuristic Programming Approach*, McGraw-Hill, New York, 1971.

24. Garvin, P. L. (ed.), *Natural Language and the Computer*, McGraw-Hill, New York, 1963.

25. Sass, M. A., and W. D. Wilkinson (eds.), *Computer Augmentation of Human Reasoning*, Spartan, Washington, D.C., 1965.

26. Martins, G. R., "Dimensions of Text Processing," *Proc. 1972 FJCC*, AFIPS, Vol. 41, Part II, 801-810.

27. Knuth, D. *The Art of Computer Programming*: Vol. I *Fundamental Algorithms*, Chap. 2 "Information Structure" Addison-Wesley, Reading, Mass. 1968.

28. Shapiro, S.C., *The MIND System: A Data Structure for Semantic Information Processing*, The Rand Corporation, R-837-PR, August 1971.

29. Levien, R. E., and M. E. Maron, "A Computer System for Inference Execution and Data Retrieval, *Comm. ACM, 10*, 11, November 1967, 715-721.

30. Kochen, M., D. M. MacKay, M. E. Maron, M. Scriven, and L. Uhr, *Computers and Comprehension*, The Rand Corporation, RM-4065-PR, April 1964.

31. Kuhns, J. L., *Answering Questions by Computers: A Logical Study*, The Rand Corporation, RM-5428-PR, December 1967.

32. DiPaola, R., "The Solvability of the Decision Problem for Classes of Proper Formulas and Related Results," *J ACM, 20*, January 1973, 112-126.

33. Woods, W. A., and R. M. Kaplan, *The Lunar Sciences Natural Language Information System*, BBN Report 2265, Cambridge, Mass., September 1971.

34. Woods, W. A., *An Experimental Parsing System for Transition Network Grammars*, BBN Report 2362, Cambridge, Mass., May 1972.

35. Woods, W. A., R. M. Kaplan, and B. Nash-Webber, *The Lunar Sciences Natural Language Information System: Final Report*, BBN Report 2378, Cambridge, Mass., June 1972.

36. Dostert, B. H., and F. B. Thompson, *The System of REL English*, California Institute of Technology, REL Report 1, September 1971.

37. Dostert, B. H. "REL--An Information System for a Dynamic Environment," *REL Report No. 3*, California Institute of Technology, December 1971.

38. Charniak, E., "Jack and Janet in Search of a Theory of Knowledge," *Proc. Int. Joint Conf. Artificial Intelligence*, Stanford, Calif., 1973.

39. Carbonell, J. R., "AT in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," *IEEE Trans. Man-Machine Systems* MMS-11, December 1970, 190-202.

40. Teitelman, W., "Do What I Mean: The Programmer's Assistant," *Computers and Automation*, April 1972, 8-11.

41. -----, "Toward a Programming Laboratory," *Proc. Int. Joint Conf. Artificial Intelligence*, Washington, D. C., 1969, 8-11

42. Teitelman, W., D. G. Bobrow, A. K. Hartley, and D. L. Murphy, *BBN-LISP TENEX Reference Manual*, Bolt Beranek and Newman, Cambridge, Mass., 1972.

43. Zobrist, A. L. and F. R. Carlson, Jr., "An Advice-Taking Chess Computer," *Scientific American*, 228, June 1973, 92-105.

44. Newell, A., et. al., *Speech-Understanding Systems: Final Report of a Study Group*, National Technical Information Service, Springfield, Virginia.

45. Duda, R. O., and P. E. Hart, "Experiments in the Recognition of Hand-Printed Text: Part II-Context Analysis," *Proc. FJCC*, Spartan, New York, 1968, 1139-1149.

46. Alter, R., "Utilization of Contextual Constraints in Automatic Speech Recognition," *IEEE Trans. Audio Electroacoustics*, AU-16, March 6-11, 1968.

47. Vicens, P., "Aspects of Speech Recognition by Computer," Ph.D. Dissertation, Stanford University, April 1969. (Also available U. S. Dept. of Commerce Clearinghouse for Federal Scientific and Technical Information, AD687720)

48. D. R. Reddy, L. D. Erman, and R. B. Neely, "A Model and a System for Machine Recognition of Speech," *IEEE Trans. Audio Electroacoustics, Special Issue on 1972 Conference on Speech Communication and Processing, Vol. AU-21*, pp. 229-238, June 1973.

49. V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy, "Organization of Hearsay II Speech Understanding System," *IEEE Transactions on Acoustics, Speech, and Signal Processing, (Special Issue on IEEE Symposium on Speech Recognition), Vol. ASSP-23*, pp. 11-24 February, 1975.

50. *IEEE Transactions on Acoustics, Speech, and Signal Processing, (Special Issue on IEEE Symposium on Speech Recognition) Vol. ASSP-23.* February 1975.

51. W. A. Woods, "Motivation and Overview of SPEECHLIS: An Experimental Prototype for Speech Understanding Research," *IEEE Transactions on Acoustics, Speech, and Signal Processing, (Special Issue on IEEE Symposium on Speech Recognition)*, Vol. ASSP-23, pp. 2-10, February, 1975.

52. W. A. Lea, M. F. Medress, and T. E. Skinner, "A Prosodically Guided Speech Understanding Strategy," *IEEE Transactions on Acoustics, Speech and Signal Processing (Special Issue on IEEE Symposium on Speech Recognition)*, Vol. ASSP-23, pp. 30-38, February 1975.

53. T. B. Martin, "Applications of Limited Vocabulary Recognition Systems," *IEEE Transactions on Acoustics, Speech, and Signal Processing, (Special Issue on IEEE Symposium on Speech Recognition)*, Vol. ASSP-23 February 1975.

54. Turn, R. A., S. Hoffman, T. Lippiatt, *Potential Military Applications of Speech Understanding Systems*, The Rand Corporation, R-1434, June 1974.

55. Feldman, J. A., J. R. Low, D. C. Swinehart, and R. H. Taylor, "Recent Developments in SAIL--An Algol-Based Language for Artificial Intelligence," *Proc. 1972 FJCC*, AFIPS, Vol. 41, Part II, 1193-1202.

56. Schank, R. C. and K. M. Colby, eds., *Computer Models of Thought and Language*, W. H. Freeman and Company, San Francisco, 1973.

57. Wilks, Y., "Do Machines Understand More Than They Did?", *Nature*, Vol. 252, 22 November, 1974, pp. 275-278.