DOCUMENT RESUME

ED 146 577 CS 003 792

AUTHOR Bond, Nicholas A., Jr.; And Others
TITLE Studies of Verbal Problem Solving: I. Two Performance-Aiding Programs. Technical Report No. 83.
INSTITUTION University of Southern California, Los Angeles. Behavioral Technology Labs.
SPONS AGENCY Advanced Research Projects Agency (DOD), Washington, D.C.; Office of Naval Research, Arlington, Va. Personnel and Training Research Programs Office.
PUB DATE Aug 77
CONTRACT N00014-75-C-0838
NOTE 43p.

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS College Students; Computer Assisted Instruction; *Computer Programs; *Logical Thinking; Man Machine Systems; *Problem Solving

ABSTRACT

This booklet describes two computer programs that were written to provide on-line aid to problem solvers. Both programs were designed for "membership" problems, or those in which there are several English sentences and implicit relationships. The task was to infer a membership structure that is compatible with all the logical constraints. Membership problems may be cast in various settings (for example, a murder mystery, where a culprit is to be identified). One program (FIRST) was based on Findler's Universal Puzzle Solver; the other (GABE) used Wang's theorem-prover logic. Of the two programs, FIRST appeared to be the most feasible for use with college-level subjects. It accepts logical inputs in a near-English format and shows the current logical status of a problem through a tabular array. The program's structure suggests a "depth of inference" measurement technique. When all possible logical paths of a logical problem are known, the "depth" of any given node in the path can be obtained from probability-of-success numbers at that node. A subject's logical progress along a path can also be computed and displayed. (Author/AA)

DEPARTMENT OF PSYCHOLOGY

UNIVERSITY OF SOUTHERN CALIFORNIA

Technical Report No. 83

STUDIES OF VERBAL PROBLEM SOLVING:

I. TWO PERFORMANCE-AIDING PROGRAMS

August 1977

Nicholas A. Bond, Jr.
William T. Gabrielli
Joseph W. Rigney

Sponsored by

Personnel and Training Research Programs
Psychological Sciences Division
Office of Naval Research

and

Advanced Research Projects Agency
Under Contract No. N00014-75-C-0838

ARPA TECHNICAL REPORT

1.	ARPA Order Number	: 2284
2.	ONR NR Number	: 154-355
3.	Program Code Number	: 1 B 729
4.	Name of Contractor	: University of Southern California
5.	Effective Date of Contract	: January 1, 1977
6.	Contract Expiration Date	: 30 September 1977
7.	Amount of Contract	: $150,000
8.	Contract Number	: N00014-75-C-0838
9.	Principal Investigator	: Joseph W. Rigney (213) 741-7327
10.	Scientific Officer	: Marshall Farr
11.	Short Title	: Studies of Verbal Problem-Solving

This Research Was Supported

by

The Advanced Research Projects Agency

and by

The Office of Naval Research

And Was Monitored by

The Office of Naval Research

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report #83	2. GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle) STUDIES OF VERBAL PROBLEM SOLVING: TWO PERFORMANCE-AIDING PROGRAMS		5 TYPE OF REPORT & PERIOD COVERED 1 Jul. - 30 Sept. 1977 6 PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) N. A. Bond, J. W. Rigney & W. F. Gabrielli		8 CONTRACT OR GRANT NUMBER(s) N00014-75-C-0838
9 PERFORMING ORGANIZATION NAME AND ADDRESS Behavioral Technology Laboratories University of Southern California Los Angeles, California 90007		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element: 61153N Project: RR042-06 Task Area: RR042-06-01 Work Unit: 154-355
11 CONTROLLING OFFICE NAME AND ADDRESS Personnel and Training Research Programs Office of Naval Research (Code 458) Arlington, Virginia 22217		12. REPORT DATE September 1977 13. NUMBER OF PAGES 34 + iv
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15 SECURITY CLASS. (of this report) Unclassified 15a DECLASSIFICATION/DOWNGRADING SCHEDULE
16 DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18 SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Verbal Problem Solving, Intersentence Processing, Logical Inference, Man-Computer Symbiosis		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Two computer programs were written to provide on-line aiding to human problem solvers. Both programs were written in time-shared BASIC, and were designed for "membership" problems. In this kind of problem, there are several English sentences and implicit in the sentences are various relations; the task is to infer a membership structure that is compatible with all the logical constraints. Membership problems may be cast in various settings, such as a murder mystery where a culprit is to

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

be identified.

One program (FIRST) was based on Findler's "Universal Puzzle Solver" concept; the other (GABE) used Wang's theorem-prover logic. In both programs, the human operator converted English problem sentences to logical membership relations. The programs kept track of all relations entered, indicated when more data inputs were needed, and scored whether a correct answer was achieved.

Of the two programs, FIRST appears to be most feasible with ordinary college subjects. It accepts logical inputs in a near-English format, and shows current logical status of a problem via tabular arrays of X's and O's. The present version of GABE used a strict "p, q, r" logical notation; college subjects find this difficult and unsatisfactory.

The structure of the FIRST program suggests a "depth-of-inference" measurement technique. When all possible logical paths in a membership problem are known, the "depth" of any given node in the path can be obtained from probability-of-success numbers at that node; also it appears that a subject's logical progress along a path can be computed and displayed. Further empirical work will explore the usefulness of such depth measures for scoring individual performances, and for teaching problem-solving heuristics in technical materials.

SUMMARY

Two computer programs were written to provide on-line aiding to human problem solvers. Both programs were written in time-shared BASIC, and were designed for "membership" problems. In this kind of problem, there are several English sentences and implicit in the sentences are various relations; the task is to infer a membership structure that is compatible with all the logical constraints. Membership problems may be cast in various settings, such as a murder mystery where a culprit is to be identified.

One program (FIRST) was based on Findler's "Universal Puzzle Solver" concept; the other (GABE) used Wang's theorem-prover logic. In both programs, the human operator converted English problem sentences to logical membership relations. The programs kept track of all relations entered, indicated when more data inputs were needed, and scored whether a correct answer was achieved.

Of the two programs, FIRST appears to be most feasible with ordinary college subjects. It accepts logical inputs in a near-English format, and shows current logical status of a problem via tabular arrays of X's and O's. The present version of GABE used a strict "p, q, r" logical notation; college subjects find this difficult and unsatisfactory.

The structure of the FIRST program suggests a "depth-of-inference" measurement technique. When all possible logical paths in a membership problem are known, the "depth" of any given node in the path can be obtained from probability-of-success numbers at that node; also it appears that a subject's logical progress along a path can be computed and displayed. Further empirical work will explore the usefulness of such depth measures for scoring individual performances, and for teaching problem-solving heuristics in technical materials.

ACKNOWLEDGEMENTS

This research was sponsored by ONR Contract N00014-75-C-0838. The support and encouragement of Marshall Farr and Henry Halff, Personnel and Training Research Programs, Office of Naval Research, and of Harry F. O'Neil, Jr., Program Manager, Cybernetics Technology Office, Defense Advanced Research Projects Agency, is gratefully acknowledged.

We should like to thank Professor Nicholas V. Findler of SUNY-Buffalo for his courtesy in providing a listing of his "Universal Puzzle Solver" program. Don McGregor and Dale Terra helped to formulate and tryout the programs.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Certain intellectual tasks, such as estimating and combining probability information, controlling several aircraft, or troubleshooting equipment, may be aided by a special type of computer program. This type of program has in it a representation of the real-world setting, and can quickly perform the library, bookkeeping, and calculating chores; the controlling human remains on line, contributing inputs and judgments. It is possible to achieve a genuine man-computer interaction in this way; and the output may be appreciably better than either man or computer could produce alone. This report describes some preliminary investigations of computer-program aids for humans who are attempting to solve verbal problems and verbal puzzles.

Motivation for selecting verbal problems for our attention came from several places. A practical reason for building such aids derives from the fact that some of the hardest problems facing humans are cast in part-verbal form. A familiar example here is the technician who must operate, calibrate, or troubleshoot a complicated electronic or mechanical device. His tech manuals, diagrams, and previous training may sometimes provide adequate information for him. But his performance must be a mixture of hypothesis-formation-and-test behaviors, combined with inferences about the meaning of observed events. When he talks and thinks about his actions, the technician is apt to use qualitative verbal models of the physical actions in the equipment. His sequences of checks may be remembered in verbal form. Furthermore, his attempts to validate his interpretations may be confirmed, controlled, contradicted, or frustrated by verbal sentences in tech manuals. Conceivably, a general software aid could assist

the human in "understanding" technical information, in "keeping things straight," deciding the "best thing to do now, knowing what tests I have done so far," and in avoiding "doing the same thing over and over again." Many studies show that technicians are usually redundant and non-optimal in their search behavior, even though the correct information may reside in documentary sources. Even though "it is all in the tech manual," the search process may be quite ineffective. People must be taught, and taught specifically, on ways to extract information from complex sentences.

We expect that a systematic investigation of verbal problem-solving processes would serve to pinpoint just where the psychological difficulties are. Verbal problems usually rest upon a definite underlying structure. This structure has to be inferred from English words, and elements of the structure are then operated upon by the application of logical processes. If a computer program requires the human solver to enter the essential relations in a problem, then the program would always know just which of these relations are not yet realized by the person attempting the problem. The program could show the solver what his present solution status is, and just where the remaining logical gaps are. In fact, as we indicate later, this approach leads to a way of measuring the depth of inference required in a given verbal-problem.

Finally, the investigation of verbal problem-solving relates to other research at the Behavioral Technology Laboratories, concerned with the analysis of text processing. Verbally stated problems, of the sort studied here, are useful for studying intersentence processing, in distinction to the intrasentence processing that has been the almost exclusive concern of traditional reading research. The importance of understanding more about

intersentence processing lies in its contributions to the comprehension of text passages; (1) its saliency for understanding different types of texts, a topic that deserves more attention from the theorists, who have tended to restrict their studies to simple narrative forms; (2) its potential as a rich source of information about higher level cognitive processes, and (3) the relevance of the information-processing skills it requires to effective reading. Many of these issues are discussed in greater detail by Rigney (1977).

There is reason to believe that integration of information across sentences may require different kinds of cognitive processes than those required by intrasentence processing. This assumption is based on observations in our laboratory of two forms of what we call decoupled reading. In one form, the reader decides to read the passage, but not to read for comprehension. In the other form, the reader's intention is to read for comprehension, but somewhere in the passage he realizes he does not remember the meaning of any of the last few sentences. He had been reading at a word-by-word level, and had the feeling that he understood what he read, but he suddenly realizes that the focus of his attention was occupied with something else.

Word problems should be useful for investigating these higher level, integrative processes, since these problems are easily read sentence-by-sentence, but cannot be solved without a large amount of more difficult intersentence processing that entails deeper levels of inference. The whole question of what intersentence relationships influence cognitive processes mediating comprehension and memory, deserves intensive investigation. Some initial work has been done on story grammars for narrative

forms (Rumelhart, 1975; Thorndyke, 1977; Mandler and Johnson, 1977), but this is just a beginning. Using this research as a point of departure, work is in process at our laboratory on a second generation text grammar that will encompass forms other than the narrative form (Gordon, Munro, and Rigney, in press).

One way to characterize the text structure problem is as follows. Suppose that exactly the same words were arranged in five different ways; (1) as a random string of characters, (2) as a random string of words, (3) as a random list of sentences, (4) as a conventional paragraph with topic sentence and amplifying sentences, and, finally, (5) as a word puzzle. If these five different arrangements of characters were given to subjects to read, clearly each arrangement would evoke different kinds of cognitive processing, under the same objective. If subjects were given the objective of memorizing the passages, there would be differences among them in time to completion, errors in protocols, and length of retention. If subjects were told to read the passages for comprehension, there also would be differences among the dependent variables. It would, in fact, be difficult to find common measures of comprehension. The meaning of each passage would be quite different. Why?

Our interest is in the different answers to this question required for different text forms above (3), the random list of sentences. We do not know, at this point, how many meta-sentence level forms exist. Possibly there are many classes and many variations within each class. Rigney (1976) speculated that there are at least four; narrative, explanation, description, and prescription. It remains to be seen whether this will be a useful classification. We are reminded of some interesting

variations in text forms. For example, in the Bransford and Johnson (1973) passage on washing clothes, the meaning of the entire passage depends on the information that it is about washing clothes. Each sentence in the passage relates to washing clothes. This seems to be the crucial intersentence relationship. Other intersentence relations seem to be primarily those found in prescriptions; various objects are manipulated in temporal sequence, determined at least partly by causal relationships. But the sentences in the passage are so worded that the prescription might be for any of a number of tasks, which leaves the reader confused until the information is given him that the passage is about washing clothes. Bransford and Johnson demonstrated that subjects given this information before the passage was read had higher comprehension and recall scores than subjects who did not have this prior information.

Word problems embody a different text form. The first sentence establishes a cast of characters and some of their attributes. The following sentences describe relationships among some of these attributes without identifying which characters are involved. The last sentence is a question requiring the identification of the character with a specified attribute. This requires the reader to (1) do deeper processing of his prior knowledge, (2) to make inferences about which character could possess which attribute, and (3) to hold a large amount of information in temporary store. An example of a word problem is:

Mr. Scott, his sister, his son, and his daughter
are tennis players.
The best player's twin and the worst player are
of the opposite sex.

The best player and the worst player are

the same age.

Who is the best player?

Solving this problem requires deeper processing of a kinship schema (Munro and Rigney, 1977) to retrieve the information that Scott's sister could be the same age as his children, and to make inferences that can be formalized in the propositional calculus. These inferences also are deeper than a reader ordinarily would indulge in if the last question was omitted.

We view this kind of text form as being useful for learning more about how people do deeper processing and deeper inference during intersentence processing, and for a measure of current information processing capacity, Hunt's (1977) CIP capacity, using text processing skills, rather than the simple tasks of the verbal learning laboratory that theorists presume to be involved in text processing but that have not been demonstrated to underly the tasks of intersentence processing.

The principal thrusts of the research described here were an exploratory investigation of the difficulty of word problems for students, and an investigation of how students interact with a computer program designed to accept their inferences during intersentence processing and to give them feedback that would assist them in solving word problems.

To date, we have tried out two interactive computer programs that might be expected to serve as problem-solving aids. These programs have not yet been fully evaluated; but they are now working, they do "solve" verbal problems, and we have gained some experience with college students using them on line. In this report, we give a simple example of a word

problem and its solution. Then we describe the computer programs themselves. The program listings along with sample problem solutions are printed in full in the Appendix. The last part of the report recapitulates our experiences with the programs so far, and offers some suggestions for extending these investigations.

II. AN EXAMPLE OF A WORD PROBLEM AND ITS SOLUTION

To fix the present setting, let us turn to the following reference problem, which was originated some decades ago by the English puzzle expert Henry Dudeney, and is presented here in Americanized form.

1. Smith, Jones and Robinson are the engineer, brakeman and fireman on a train, but not necessarily in that order. Riding the train are three passengers with the same three surnames, to be identified in the following premises by a "Mr," before their names.

2. Mr. Robinson lives in Los Angeles.

3. The brakeman lives in Omaha.

4. Mr. Jones long ago forgot all the algebra he learned in high school.

5. The passenger whose name is the same as the brakeman's lives in Chicago.

6. The brakeman and one of the passengers, a distinguished mathematical physicist, attend the same church.

7. Smith beat the fireman at billiards.

Who is the engineer?

This is a class-membership problem. When well-formed, such problems have a unique solution, the reasoning can be followed by ordinary people, and the special information demands are not excessive. Thus we suppose that everybody knows that Chicago, Omaha and Los Angeles are cities; and everybody also knows that if Smith beat the fireman at billiards, as stated in premise 7, then Smith cannot <u>be</u> the fireman.

When educated adults are given this problem without aids or without any special training, they get the right answer within 15 minutes or so (about 80% of one large psychology class solved it). A few, perhaps

five percent, will not seriously attempt to solve it ("I'm not good at this sort of thing"); some will approach the problem in a proper spirit, but will make mistakes and come up with a wrong answer; a very few will propose answers to the problem on some non-logical grounds ("Physicists just don't live in Omaha, they'd be more likely to live in L.A. or Chicago"). Successful solvers show marked individual differences in their solution time (some get it in less than two minutes); the subjects will also differ in their confidence about their reasoning processes. After being shown a logically sound path to a solution, however, very few educated adults will doubt the answer.

It may be helpful to set up a tabular representation of the problem; in Figure 1, the matrix on the left has to do with the railroad employees, and the right-hand matrix concerns the passengers. When a logical possibility is eliminated, we put an "X" in a cell; when a cell is true, we insert a small dot.

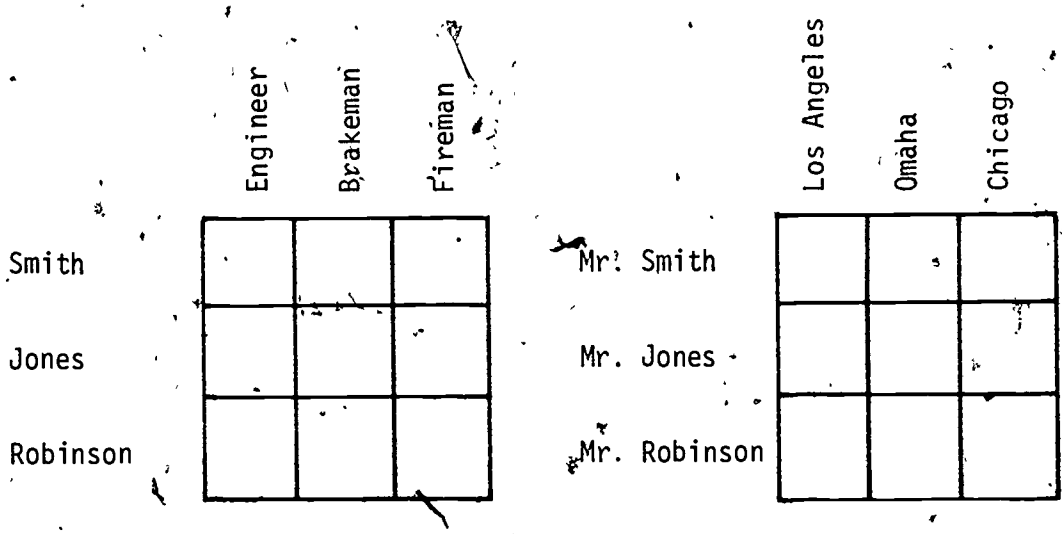

Figure 1. Two Matrices for the "Smith-Jones-Robinson" Problem.

Right off, we can enter a dot for the lower left-hand corner of the second matrix: from premise 1, Mr. Robinson lives in Los Angeles, and not in the other two cities. So there must also be X's in the Robinson cells for Omaha and Chicago; and X's in the Los Angeles column for Mr. Smith and Mr. Jones. As we have already noticed, premise 7 plainly indicates that Smith is not the fireman, so we enter an X in the appropriate place in the left-hand matrix. Now the table looks like this:

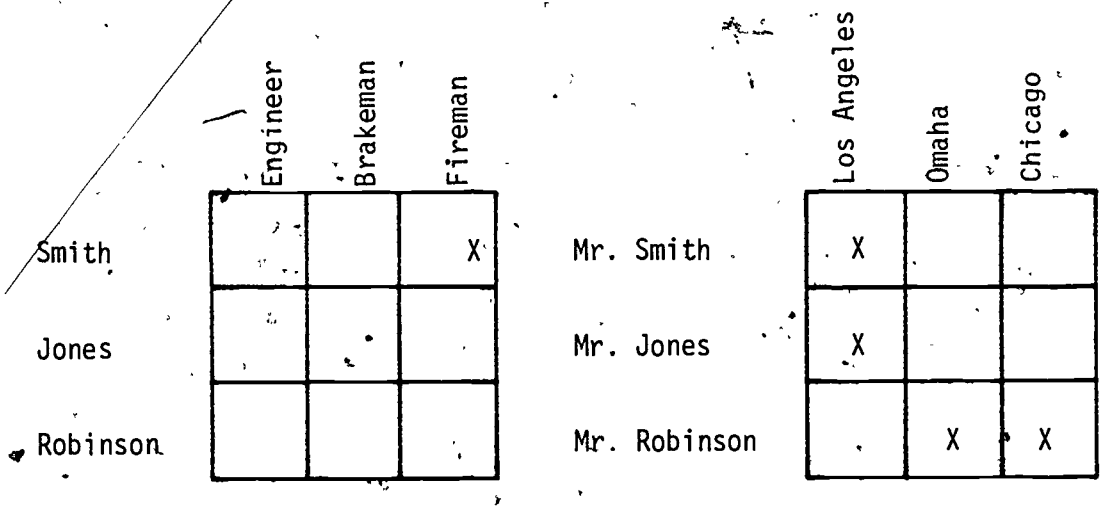

Figure 2. "Smith-Jones-Robinson" Matrices, after Data are Entered from Premises 1 and 7.

There are still a dozen indeterminate cells in the two tables, so we must now begin to <u>combine</u> information from two or more sentences. Scanning the set, we see that premises 3 and 6 imply that the physicist lives in Omaha; and since we already know he cannot be Mr. Robinson, then he must be either Mr. Jones or Mr. Smith. But from premise 4, the physicist cannot be Mr. Jones (because you cannot be a physicist and still have forgotten all your high school algebra). Hence, when you take 3, 4, and 6 together, you see that the physicist must be Mr. Smith. This effectively

fills in the second table, because there is nowhere left for Mr. Jones to live except in Chicago. Now we can go back to the first table, and see that from premise 5, Jones must be the brakeman; and so the final answer is that Smith is the engineer. The problem is solved. Of course, this particular problem can be solved without any computers; or perhaps without any graphs or recording techniques. For problems that are longer or that are more complicated, though, the potential usefulness of computer-aiding increases. It might even be possible to teach subjects, via computer-aiding, to become champion solvers of this kind of problem.

III. THE COMPUTER PROGRAMS

To our knowledge, there are two published reports on computerized systems for solving problems like the Smith-Jones-Robinson example. In 1956, John G. Kemeny programmed a gigantic twelve-premise problem which Lewis Carroll had posed about 80 years earlier. Twenty years ago, his solution took four minutes on an IBM 704 (Kemeny, 1956), a complete printing of the "truth table" of the problem would have taken 13 hours! (With present technology, the computations would have taken several seconds, and the printing some few minutes).

Seventeen years after Kemeny's tour de force at RAND, Nicholas Findler (1973) described a "Universal Puzzle Solver" program. Findler's program, which was written in SNOBOL, operated via a membership logic structure and a recursive search subroutine. English words for set members and relations have to be entered, along with absolute and conditional membership statements; these logical statements are derived, by a human, from the original English-language problem sentences. Once all the problem and solution conditions are entered, the program sets up appropriate arrays, and then searches these for a solution. The search is systematic but brute force. On medium-fast processors, a problem runs in a second or two. Answers are printed in constrained English sentences. The program has elegant provisions for multi-stage problems, for conditional relations between variables, and for output of results. A most unusual feature of Findler's work concerns the generality of the program; at the end of his paper describing the program, he says he cannot see any way, or any need, to extend its capabilities further (Findler, 1973). Because of this generality, and the ingenuity of Findler's search routine, we decided to adapt

Findler's concept for one of our aiding programs; we called our version FIRST (Findler Interactive Routine for Subject Training). Because time-share SNOBOL was not available on our mini-computer, we recoded parts of Findler's program into extended BASIC. Also, several features were added to suit our purposes better: for instance, provision was made for identification and correction of errors in entering problem information; the subject's information state was tracked at every step; tabular graphs of logical inclusions and exclusions already achieved were available on demand.

The first important inputs to FIRST from the human subject are dimension and set specifications. In the San Francisco problem shown in the Appendix, there are five people who get to work in five different ways; there are then, five members in set 1 (Al, Bill, Chuck, Dave and Ed), and five in set 2 (bike, car, BART, bus, walk). Nineteen sentences are listed, and these sentences contain enough information to allow each person to be associated with a mode of transportation. Membership relations from these sentences are written in "CON" or "NOT-CON" form. "CON" means a strict logical connective is established; "NOT-CON" is a logical exclusion. So when sentence 14 says "Dave greets his driver with 'good morning' everyday," a solver might enter the following FIRST statements:

DAVE, NOT-CON, WALKING

DAVE, NOT-CON, BIKE

DATE, NOT-CON, BART

Some local information is needed in this problem. The logical assertion about BART is less obvious than the other two; you have to know that BART is a rail transit system, and also that a BART driver is inaccessible, and cannot be spoken to (the driver doesn't "drive" the vehicle, a computer drives it; the driver is there for override purposes).

When the subject working a problem wants a "present-status" printout, he hits a control key, and a tabular presentation appears on the terminal; this table shows "O" for membership and "X" for non-membership. In a five-variable problem, if there were three X's in a given column, then the solver might focus on that variable, and go over the problem sentences again, in order to find a fourth exclusion and thus pin down the identity of the column member.

We selected Wang's theorem-prover system, called GABE in the Appendix, as the model for the second program. As in the Findler approach, to use the system a human has to accomplish some translation of complex English sentences into logical relations, using only the "and," "or," and "not" operators. But instead of a Findler-style recursive search for one or a few right answers, in Wang's system, after you have inserted the premises; you then must ask the program whether a given outcome statement is valid or not. Thus, after coding the Smith-Jones-Robinson problem into Wang notation, you would have to suggest to the program the following three "theorems:"

Smith is the engineer.

Jones is the engineer.

Robinson is the engineer.

All three theorems would be "tested," via the Wang algorithm; of course, only the first would turn out to be valid, if the problem relations were properly entered. As it is now set up, the system does not list all possible valid statements from a set of premises; you have to ask it about specific ones that are of interest. In fact, from a small but fairly rich set of premises, an enormous number of valid "theorems" can be derived, and it would often be impractical to print the whole list.

Putting a verbal problem into the Wang process is more abstract to the subject than is FIRST. English problem statements are converted into bare representations; then, terms in these representations are given a symbolic translation into logical operators. As an illustration, we take the "murder" problem from Raphael (1976).

Wang's algorithm works by following a staged reduction routine. The procedure writes down a series of logical lines, each simpler than the preceding one. The simplification continues until the same logical expression occurs on both sides of a central arrow, or until a mismatch occurs. The Appendix shows this line-shortening process as it worked in one problem. We originally hoped that human subjects could learn, by imitating the algorithm, how to process logical terms; or at least, we thought that some subjects would become intrigued with Wang's reduction and proof scheme. This view was naive, as it turned out; the details of the Wang operation are totally mysterious, and also totally uninteresting, to the ordinary adult.

THE PROBLEM

The Facts

The maid said that she saw the butler in the living room. The living room adjoins the kitchen. The shot was fired in the kitchen and could be heard in all nearby rooms. The butler, who has good hearing, said he did not hear the shot.

To Prove

If the maid told the truth, the butler lied.

THE REPRESENTATION

p = The maid told the truth
q = The butler was in the living room
r = The butler was near the kitchen
s = The butler heard the shot
u = The butler told the truth

ORIGINAL STATEMENT	EQUIVALENT FORM	MEANING
PREMISES		
$p \supset q$	$\mu p \vee q$	(If the maid told the truth, the butler was in the living room).
$q \supset r$	$\mu q \vee r$	(If the butler was in the living room, he was near the kitchen).
$r \supset s$	$\mu r \vee s$	(If he was near the kitchen, he heard the shot).
$u \supset \mu s$	$\mu u \vee - s$	(If he told the truth, he did not hear the shot).
THEOREM		
$p \supset \mu u$	$\mu p \vee - u$	(If the maid told the truth, the butler did not).

Figure 3. A Mystery Solved by Propositional Calculus. The problem and its representation.

IV. TRYOUT OF PROGRAMS WITH HUMAN SUBJECTS

When materials like the two aiding programs described above are prepared, there are some questions which can be answered only by experimentation. For instance, do the programs teach problem-solving more effectively than does simple undirected practice; will subjects attempt to imitate the computer way of doing things? After a few problem sessions on the computer terminal, what are the transfer effects from one problem to another? Another class of issues concerns feasibility of the software concept. Can ordinary people use the program, and will they readily use it? Are the materials self-administering and easy to run? Without standby programmer staff; do subjects seem comfortable in the situation? Does performance seem to improve? What kind of performance model do the subjects appear to follow, etc. It is to this second class of questions that this part of the report is addressed; the experiences reported here are based on a grab sample of California State University undergraduates. Our impressions so far can be imparted quickly, under half a dozen headings.

1. General feasibility. Both programs run at present on a time-shared PDP-11. They probably will run on any medium-capacity time-shared system. No remarkable operating problems arose in ordinary program use, though we often wished we had a better restart procedure. Neither program had any provision for referring to a library of problems, so to start each problem, a staff member usually handed the subject the problem sentences on a separate piece of paper, and stayed nearby while the subject worked. Because of the large amount of text material, and because the

subjects often wanted to refer to some previous logic table or data entry, it was necessary to employ hard-copy teletype terminals; video terminals could be used only for small problems. Some large problems took two or three feet of paper to reach a solution. One incidental result; with an assistant nearby, subjects were often tempted to engage the assistant in conversation about the problem sentences, and to seek some immediate confirmation of the logical expressions being entered into the terminal.

For Wang's reduction program, GABE, it was not feasible for ordinary students to convert English sentences into logical symbols. This was probably due to the general lack of fluency with the logical operator notation: p, q, μ , ∧ , ∨ , ⊃ , etc. Also, there were often two stages of "stripping" the English sentence down into symbols. We tried to give a "short course" in the notation to several people, but there was general and specific resistance: generally against <u>any</u> logical symbolism, and specifically against the (μ p ∨ q) representation of if-then or implication. We conclude that any serious use of the Wang reduction concept as an aid would require considerable pre-requisite training in logical notation and in translation. We suppose, too, that people who are fluent in artificial languages, such as computer programmers, would find the system more acceptable.

2. <u>Data Input</u>. There is no doubt that subjects find the teletype format to be a "slowdown," and somewhat frustrating. The presentation is "all words," and constrained words at that; everything has to be typed in; and on a fairly large problem; the subject cannot be sure whether or not he/she has enough data to reach a solution. He then must request the

computer to print out a current state table (or the program itself decides to print one); and at ordinary teletype speeds this takes some time and interrupts the solution process. So the clanking terminal may be a real distraction to the solver. There are input-output devices on the horizon that could help to alleviate this problem.

3. Individual Differences. For entering logic from plain, descriptive sentences, the FIRST aiding program reduces individual differences to near zero. In the first sweep through the problem sentences, when each sentence is taken separately, the human solver simply converts the sentence meaning into a "CON" (membership) statements, or a "NOT-CON" (exclusion) statement. "Mr. Robinson CON: Los Angeles," would be one example from our reference problem.

When the subject has to combine information from two or more sentences, or has to realize some "deeper" aspect of the facts presented, then the variation between people can be quite marked. If facts from two or more sentences are processed in such a way as to provide a new, non-trivial inference, then the subject first has to select the sentences to be considered together; this means that a dimensional scanning operation must be performed. Next the subject has to do further processing to reach a new inference.

The combining processes can be illustrated with one of our favorite problems, "The Murderer," taken from Summers (1968):

> Murder occurred one evening in the home of a married couple and their son and daughter. One member of the family murdered another member, the third member witnessed the crime, and the fourth member was an accessory after the fact.
>
> 1. The accessory and the witness were of opposite sex.
> 2. The oldest member and the witness were of opposite sex.
> 3. The youngest member and the victim were of opposite sex.

4. The accessory was older than the victim.
5. The father was the oldest member.
6. The killer was not the youngest member.

Who was what?

Each of the first three sentences in this problem contains an easy conditional relation: for instance, (1) implies that if the accessory is female, then the witness is male, and vice versa. Anybody who can read English will be able to enter these relations into the program. Some of the combinations between sentences are easy, too. Look at premises (1) and (2). The last seven words of these two premises are identical, and the sentences are right next to each other; so the circumstances favor a comparison between the two. It then quickly appears that the oldest member and the accessory are of the same sex. Other combinations may not be quite so easy, but are still likely to be achieved. For example, from premise (5) we know that the father was the oldest; so we could already infer, at this stage in the search for a solution, that the witness was female.

A more difficult, but also more intellectually satisfying, inference chain goes as follows. Suppose we explore the identity of the "youngest member," and start working across sentences. From (6) the youngest member cannot be the killer; from (3) the youngest member cannot be the victim; so the youngest member must be either the accessory or witness. But from (4) we see that the accessory is older than somebody, and hence also cannot be the youngest. Therefore the youngest must be the witness since all other possibilities have been eliminated. The difficulty in attaining this chain of reasoning stems mostly from the (4) inference about the accessory. Scanning premises (6) and (3) was relatively easy and direct, because both have

straight-forward language mentioning "youngest member." But in premise (4), youngest member does not appear as a term per se; we have to deduce something about youngest member from the "older than" relation.

The problem is now easily solved; the witness is the daughter, the accessory is the father, and so on. There are several other logical paths that can reach a correct solution; or, all possible role-membership combinations (24 in this particular problem) could be tried and tested against the original problem sentences until an acceptable set of assignments met the conditions (the original Findler program would actually proceed in this manner).

We have seen enough solution attempts to believe that multi-sentence scanning, selection, and combining skills may be the key to successful problem-solving of this type. The basic identification and negation logic is apparently easy enough, once the appropriate meaning sources are put together in a small package of critical phrases, and examined closely for their logical implications. If this view proves to be correct, then effective training methods will focus heavily on the cognitive processing of several temporarily-combined sentences or long phrases, and not on the strictly logical processing of identification, negation, and conditionality relations. To put it another way: once you are looking at the right phrases and relations to combine, and confine your attention to just one or two main inclusions or conditionalities, then the logic itself is easy.

4. Depth of Inference.* It appears, then, that subjects who use our FIRST version of Findler's concept are performing a complex translation task. English sentences are read, and the logical gist of the sentence(s) is typed into the terminal using the "CON" or "NOT-CON" entry conventions. Variable names remain in English, and part of the solution output appears as a simple English sentence. The computer always knows, then, the exact logical relations that the subject has put into the machine, and the order in which these were entered. It is perhaps useful to define depth of inference in terms of (1) the probability that a given inference is ever achieved, in a reference sample of subjects, and (2) the primacy with which a logical relation is deduced. Both probability and primacy values can be extracted from computer records of problem attempts. The Murderer example given above permitted an easy and convincing decision that the youngest member was not the victim, and not the killer; it was much harder, as we saw immediately to perceive that the youngest member could not be the accessory either. The performance of subjects could be easily checked, by counting the frequency and time order of the following three entries into the FIRST logical arrays:

YOUNGEST MEMBER: NOT-CON: VICTIM

YOUNGEST MEMBER: NOT-CON: KILLER

YOUNGEST MEMBER: NOT-CON: ACCESSORY

---

* We use this phrase instead of the overworked "depth of processing" of Craik and Lockhart (1972), which they defined as the deployment of a flexible processor over any of several stages of processing, presumed to intervene between sensory inputs and semantic processing in LTM. Depth of inference could be considered to be a form of the latter, and thus might be one of many kinds of deep processing.

Depth-of-inference indexes, then, can readily be determined in the computer-aided problem-situation. These could be useful at the individual level (what is this subject's average depth-of-inference in the first five minutes of some set of reference problems?), or at the group performance level (which inferences in this particular problem are deepest?). Obviously, depth-of-inference indicators could be used to check the effectiveness of a training program, or of some other intervention. When properly standardized, problem inferences could be scored for depth, and individuals ranked according to their performance.

The logical inference task, we expect, requires some elaborative processes that are not often found in word-memory tasks commonly used to test Craik's and Lockhart's (1972) depth-of-processing concept. We can see some parallels between these two areas. Craik and Tulving (1975) found that subjects do not remember "... what was 'out there' but rather what they did during encoding." We predict that aided problem-solvers will remember best (and perhaps enjoy most) the difficult but productive inferences. Another point of possible agreement with the depth-of-processing idea concerns the "number of features checked." Assuming some analogy with the problem-solving case, a deeper inference is one requiring recognition, selection, and scanning, of several phrases across several sentences. A membership problem with several variables will elucidate the point.

The five events in the annual Boys' High intramural swimming meet--one was a butterfly race--were won by five different "Animal League" teams, which then competed against one another to determine the teams' overall ranking. From the following clues, can you find the event each team won, the name of its captain (one was Ned), and the final ranking of the teams?

1. Will was not the captain of the backstroke winners or the diving champions.

2. The Bears did not win the freestyle race.

3. The team that won the breastroke event finished ahead of the Leopards, but behind both Will's team and the one that won the freestyle event.

4. Tom's team was not the Tigers or the Leopards.

5. The Bears finished ahead of the Lions.

6. The Panthers did not win the breastroke event, nor did the Leopards triumph in diving.

7. The Panthers did not finish last, but they were behind Paul's team.

8. The backstroke winners and the Tigers and Steve's team all finished behind the Lions.

Within ten minutes, many adults attempting this problem will see that the Bears were in first place, and the Lions second; also, since Will's team is not either backstroke, diving (clue 1), breastroke, or freestyle (clue 3), Will's team has to be the butterfly swimmers. It is easy to peg the Leopards, too; the Leopards cannot be breastroke, freestyle, or butterfly (clue 3); and they cannot be the diving team (clue 6); so they must be the backstroke team, and they also can have finished no higher than fourth (clue 3). We now have a good start on the problem; to finish it, we will probably have to realize that Tom and Will are on the top two teams; and only a few solvers will realize this, even if given half an hour or more to work on the problem (some subjects may eliminate some of the possibilities, "permute" the rest, and thus reach a correct assignment without going through all the 60 assignment possibilities; when they proceed in this way, they would be imitating the FIRST program). The psychological difficulty is that, to infer the Tom-Will placement, a lot of preliminary information has to be developed;

and then a rank-order-of-finish table processor has to operate simultaneously on several bits of verbal data. (From (8), Steve must be the leader of the Panthers; the Panthers must have finished fourth; and also from (7), Paul's Tigers must be third). Thus, there are many aspects to "hold" at the same time, and these must be appreciated firmly enough to be converted into computer-acceptable statements. As far as the computer can tell, one "CON" or "NOT-CON" assertion is as good as another, but the data demands on the human for realizing the different relations are usually quite disparate.

How might depth-of-processing concepts be used in teaching people to be good problem-solvers? One possibility is to teach problems with easier or "shallower" inferences first, up to a strict performance criterion, and then gradually to increase the depth of inferences via controlled practice. A program of this sort might be designed to be adaptive, in the sense it would adjust the practice to the "best expected gain" per unit time at the terminal. There are several empirical matters to investigate: the bases for ordering the problems in the training set, expected transfer effects across problems, the proportion of variance due to aptitude or knowledge differences, the extent to which processing tricks and gimmicks can be taught, and so on.

Another project could focus on the extremely "deep" or difficult inferences. Here the research strategy would be that, if the subject's performance on these hardest parts could be improved, then the easier tasks would take care of themselves. To teach the deeper inferences, specific training analyses would be done for each difficult inference, and the student would walk through these examples.

5. Keeping Score. We noticed that, when using the FIRST routine, a student will often tend to make rather too many "status checks;" that is, he/she will frequently ask for a printout "to see if I've solved it yet." This feature was provided in the program as an informational aid, but in some cases it may actually serve to obscure the logical process. Perhaps the student gets involved with "getting the answer," and is visibly disappointed or exhilarated when the table is filled in. This makes it more of a game, all right, but does not necessarily instruct the players. Perhaps future program versions should not permit so many table checks.

6. Differences between Analyzing Logic Tables and Human Inference Behavior. Our programs that operate upon decision tables are necessarily "clean," with nice 1's and 0's in the cells. Also, there are definite evaluation rules in the program which decide whether the problem is solved or not; the processing is aimed directly at getting a clear resolution of the set-membership relations. Actually, of course, human inference behavior is often far less than certain, and it may not know just "where it is going." As Schank (1975) put it:

> "....the (real) process of generating conceptual inferences is inherently a computationally wasteful process, because its intent is to discover what is interesting in a particular context."

This means that we should expect much elaboration behavior as subjects work on a problem. It may be possible, through directed practice, to facilitate a certain "directness" in the elaborative activities of subjects. Certainly many verbal puzzles have common dimensions; often the problem rests on variables like age, parent-child kinship relations, the days of the week, rank placement on some criterion, such as money or

other countable outcomes, or physical contiguity of events. Suppose that these standard dimensions could be listed, and specific elaborative operations be planned for each dimension. Then it should be a direct task to teach the necessary elaborative behaviors in a set of problems; perhaps a computerized scratch-pad could be provided for each of the candidate dimensions.

When a neutral observer watches a problem attempt, a frequent occurrence is that the solver will "graze," but still "miss," a key implication of a statement. In at least some cases, the trouble appears to be that a scan of a statement, or of two or more statements, alternate between two rather different processes: (1) discovering what the dimension should be, and (2) evaluating the statement for any new inferences that may come from the dimension. Perhaps these aspects should be artificially separated, at least in a training program.

V. IMPROVEMENTS AND EXTENSIONS

While tryouts have shown the feasibility of decision-table software in verbal problems, there has been no thorough evaluation of the programs as teaching aids. Before such an evaluation is undertaken, the programs need more features and capabilities than they have now. Some major changes planned are listed in the paragraphs below; in this material, we have limited consideration to those items that seem possible right now.

1. Rank-Order Dimensional Store. A problem-solver often needs to put his membership variables in some order. In age-related problems, mothers and fathers are older than sons and daughters; in the Swimmer problem on pages 23 and 24, you probably will never get the answer unless you see that the Bears and Lions are the top two teams, and that the Leopards are on the bottom with Paul's and Steve's teams in between. Such processing is done as an intermediate step. A software aid should have a call up feature that permits order information to be collected and stored outside the usual CON, NOT-CON, and conditional tables. Probably three rank-order dimensions would be sufficient for most problems. The solver could define and use these as "working files" while he is combining information from two or more sentences; once he has a firm membership statement he can go to his regular CON table entry.

Here's how it might work. Returning to the Swimmer problem, a rank-order file might be defined as "order of finish, with five slots, 1-5." From premise 5, the solver would enter "Bears ahead of Lions;" from premise 8, "backstroke team," Tigers, and Steve's team behind Lions," The system now knows that Bears and Lions are first and second. If the

solver looks at premise 7, he sees that the last team cannot be the Panthers, so he enters "Panthers not last" or some equivalent. There are only four possible orders remaining:

BEARS	BEARS	BEARS	BEARS
LIONS	LIONS	LIONS	LIONS
TIGERS	PANTHERS	LEOPARDS	PANTHERS
PANTHERS	TIGERS	PANTHERS	LEOPARDS
LEOPARDS	LEOPARDS	TIGERS	TIGERS

Seeing this table, the solver now may focus his further search to resolving the third-fourth issue for the Panthers, or perhaps to the placement of the Leopards.

2. Storing Problems. It is a nuisance to start each problem with a separate piece of paper; this necessity also requires an attendant to stand around while the solver is working. Future versions of FIRST will allow for storage of a dozen or so problems; before a session begins, the attendant will set in the order of problems, and then leave the solver alone. With new memories offering a quarter-of-a-million words of storage, there should be no further need for manual problem starts! Another software addition will be a problem restart procedure which will be easy for the subject to use.

3. Scoring System for Depth of Inference. As a silent accompaniment to the student's work, subroutines will be installed to figure continuous "depth-of-inference" scores. First attempts at doing this will use simple probability-of-success indicators for each cell in the matrix, including whether or not the entry was achieved by inclusion or exclusion logic. There will also be rough (1-minute increment) time scores for each logic entry. Every CON, NOT-CON, and conditional entry into a basic problem matrix will be flagged for this scoring system.

Several groups of subjects have been asked to reconstruct their logic, immediately after working on such problems as the Murderer. While the main results of those studies will be given in another BTL report, we can mention here that, for some problems, it is quite feasible to determine just which logical path a given subject followed. This is possible because there are only a few paths to a (logical) solution. The Murderer has four paths, and one of these is by far the most popular.

It seems that a scoring system might try to track the logical path of each subject on each problem, and print out a final account of just where the subject got as he worked on the problem. This might be a bigger software job than it appears to be right now. We expect to explore it first with a few problems wherein we already know all the logically admissable paths, and where we have some idea of the success probabilities at each node in the path.

Automatic display of the logical path achieved by a subject might be a helpful teaching aid in itself. Suppose that a subject has completed all but one or two inferences in a path; the display might be a good way for him to review his performance. A major challenge here to the software designer will be to provide a useful, but not overly complex, printout. For instance, should little remediation sentences, elements, and advices be put on the logical-path review, at those points where the solver missed something?

4. Intersentence Processing. If the critical relations in a problem flow from the combination of data from several sentences, then a software aid should do something definite about this part of the solution attempt. So far, we can formulate several heuristics which might be generally useful. The first of these would urge the solver to ask for a

status printout after his first descriptive pass through the sentences, and then to look closely at those variables which appear to be the nearest to being locked up, or totally defined in problem terms. Then these particular variables are scanned <u>across</u> sentences, to see if any more CON or NOT-CON relations can be found.

A second heuristic would recommend that, once a solid CON is achieved in the problem table, the possibility of further NOT-CON's can be made by rereading pairs of sentences containing the element which has just been "CON'ed."

As a third technique, the most informative sentences are apt to be those with a lot of words and exclusions in them. Taking two of these high-information statements together might be a good thing to do, if a solver is temporarily stuck. Sometimes, too, a key sentence will have data on two or more dimensions in it; in the Swimmer problem, premise 3 separates Will and Leopards from three other problem-elements, and also gives the indication that the Leopards cannot be better than fourth. In fact, about nine definite logical statements can be obtained from that one premise.

It is a question whether such heuristics can be suitably defined over a broad problem set; and there is a further question whether such heuristics can be utilized to advantage in new problems. We are optimistic at the moment, partly because heuristics are eminently teachable in other logical domains (such as setting up integration problems in calculus), and partly because although the words in verbal problems are complex, they aren't so complex that most terms cannot be dimensionally analyzed. Even a <u>partial</u> system for rolling over the dimensions may be enough to promote a key inference.

On long and involved problems like the Swimmers, the solution is bound to take some minutes, and there is an interesting point when the solver begins to think that he/she has just about broken the problem, and that everything will soon fall into place. Sometimes it can even happen that the solver already has enough logic to fill in the answers, if the information is just collected from all the tabular arrays. A small aid here might be a computer subroutine which would provide a running "logic score;" when this score is, say, between 0 and 1, then the solver should continue to derive new logical inclusions and exclusions. When the score goes over 1.00, then the solver knows that he can easily solve for remaining unknowns, with the inferences he has already achieved. Thus, if your score is 1.08, then your main task is to collect, from the several arrays and tables, all the facts you now have. As yet, there seems to be no completely general way to do this calculation; but it can certainly be programmed for each problem separately. It would certainly be a shame for a solver to have enough data, and not know it!

5. Automatic Composition of Logic Tables. Experienced problem-solvers may prefer to set up their own logic tables, trees, and other bookkeeping devices; the authors, for instance, often find themselves scribbling little bits of ordering data or exclusion logic, when working on a verbal problem. These notes are usually incomplete and rather hit-and-run; as in the Schank quote earlier, we are looking for something that is logically interesting. We believe, however, that most subjects like to have the computer provide to them a clear (empty) table to start with. In the Swimmer, there would be four main dimensions (team name, place, Captain's name, style of stroke) with five rows or columns on

each dimension. Future runs of the FIRST program will immediately print out a table like this, and encourage the subject to tear it off and use it as a starter recording device. At any time, the program will also be capable of printing out an up-to-date marked version, if the instructional circumstances demand it.

6. Time and Rate Indexes. Several investigators have postulated that individuals differ radically in their basic information processing capacities. Hunt (1977), for example, was able to rank-order several groups of people according to their response latencies in some simple discrimination tasks. A computer-aided system operating on logical material should be able to yield a similar "basic inference rate" over a series of standard sentences, and to tabulate this for each subject. In the next series of trials, we plan to explore this possibility in some detail. Of special interest here will be the correlation of performance on single-sentence logical processing, with a score on inter-sentence derivations. We will also be looking at the parametric and distributional features of rate measures in this domain, just as Hunt examined intercept and slope features of his speed measures. It is probably over-optimistic to think that one or two basic logical-processing parameters can really describe performance in difficult verbal problems; but it is reasonable to think that they can tell more about the processes than most other kinds of predictors.

REFERENCES

Craik, F.I.M., E. Lockhart, R.S. Levels of processing: A framework for memory research. Journal of Verbal Learning and Verbal Behavior, 1972, 11, 671-684.

Findler, N.V. A Universal Puzzle Solver. International Journal of Man-Machine Studies, 1973, 5, No. 4.

Gordon, L., Munro, A., and Rigney, J. Summaries and Recalls of Texts of Three Types, (Tech. Rep. No. 84), Los Angeles: University of Southern California, Behavioral Technology Laboratories, in press.

Hunt, E. B. Qualitative Sources of Individual Differences in Complex Problem-Solving. (Paper given at NATO Advanced Study Institute, Banff, Canada, June 1977).

Kemeny, J. G. An Experiment in Symbolic Work on the IBM 704. Santa Monica, CA.: Rand Corporation Paper 966, September 1956.

Mandler, J. M. and Johnson, N. S. Rememberances of things parsed: Story structure and recall. Cognitive Psychology, 1977, 9, 111-151.

Munro, A., and Rigney, J. W. A Schema Theory Account of Some Cognitive Processes in Complex Learning. (Tech. Rep. No. 81). Los Angeles: University of Southern California. Behavioral Technology Laboratories, July, 1977.

Raphael, B. The Thinking Computer. San Francisco: Freeman, 1976.

Rigney, J. W. On cognitive strategies for facilitating acquisition, retention, and retrieval in training and education. (Tech. Rep. No. 78), Los Angeles: University of Southern California, Behavioral Technology Laboratories, May 1976.

Rigney, J. W. Teaching Text Processing Strategies: A Research Prospectus. Los Angeles, CA., Behavioral Technology Laboratories (internal paper) 1977.

Rumelhart, D. E. Notes on a schema for stories. In D. G. Bobrow and A. Collins (Eds.), Representation and Understanding: Studies in Cognitive Science. New York: Academic Press, 1975.

Schank, R. C. Conceptual Information Processing. Amsterdam: North Holland, 1975

Summers, G. J. Test Your Logic. New York: Dover, 1972.

Thorndyke, P. W. Cognitive structures in comprehension and memory of narrative discourse. Cognitive Psychology, 1977, 9, 77-110.