U. S. AIR FORCE

PROJECT RAND

RESEARCH MEMORANDUM

THE SIMULATION OF HUMAN THOUGHT

A. Newell
H. A. Simon

RM-2506

December 28, 1959

Assigned to._____

This research is sponsored by the United States Air Force under contract No. AF 49(368)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Development, Hq USAF.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

SUMMARY

This Research Memorandum describes a method of studying human problem solving, gives an example of the application of the method, and indicates the theory of problem solving that emerges. The method consists in constructing a theory of central processes in the form of a program, demonstrating the sufficiency of the theory to produce problem solving behavior by realizing it in a computer, and testing the theory against human processes by comparing the trace generated by the program with the protocol of a human subject. The application consists in a general problem solving program, capable of solving problems in logic and other domains. The theory of human problem solving consists in a program for reasoning in terms of goals and methods for attaining those goals. The program selects paths for exploration by first determining the functions to be performed, and then finding courses of action relevant to those functions. It reflects the "insightfulness" and "directedness" that has often been observed as a salient characteristic of human problem solving. The theory constitutes a rigorous, detailed explanation of a significant area of human symbolic behavior.

CONTENTS

THE SIMULATION OF HUMAN THOUGHT

1. INTRODUCTION

Behaviorism and the acceptance of the norms of the natural sciences in psychology greatly restricted for a generation or more the range of behavioral phenomena with which the psychologist, as scientist, was willing to concern himself. Unless an aspect of behavior could be examined in the laboratory and could be recorded and measured in an entirely objective fashion it was not, in the prevailing view, a proper subject of study.

There has been considerable relaxation of this austerity in the past decade, although not without misgiving and apology. A leading text on experimental psychology, for example, in introducing the topics of problem-solving, thinking, and language behavior, observes [15]:

> "These topics have often been omitted from textbooks in the past--perhaps because of some subtle aura of 'mentalism.' Historically, of course, thought and meaning were the central problems for psychology. The wide circle that American psychologists have been making through behaviorism seems to be bringing them back again to the same core of the science, but perhaps they are returning with more precise techniques and a more objective point of view than would otherwise have been attained."

There is no need to document in detail this resurgence in psychology of concern with the central topic of thinking. This subject had been kept alive during the heyday of behaviorism by a number of outstanding men--the names of Köhler, Tolman,

Wertheimer, Bartlett, Duncker, and Maier, come at once to mind--who refused to allow fixed canons of rigor to bar them from studying the relevant and the significant. Their work provided, in turn, foundations on which the more recent investigators--Luchins, Heidbreder, Harlow, de Groot, Guetzkow, and Bruner, to mention some examples--have built.

Those who regard thinking as the core of psychological inquiry, and who urge a return to concern with it, don't want to turn the clock back. The behaviorists and operationalists are, of course, right in demanding objectivity, clarity, and rigor. Relatively few psychologists are satisfied with the vagueness of Gestalt language and with explanation at the level it permits. Few are satisfied with the eclectic language of James, and many find excessive vagueness and ambiguity in the "mediation" hypotheses of his middle-of-the-road descendants.

The task is not simply to restore thinking to the center of the psychological stage; it is to study thinking with as much methodological sophistication as we demand for simpler phenomena. Consequently, the increasing attention to thought processes does not merely reaffirm their importance; it reflects a growing belief that the techniques of psychological inquiry have become adequate, at least to some degree, to the subject matter. Whether the belief was fully justified a decade ago, when the resurgence began, is a moot question. The main thesis of this paper is that the belief is justified

now--that the technological advances that are necessary to permit a theory of thinking to be formulated and tested have occurred.

To understand complex phenomena we must have powerful tools of inquiry--tools for observing facts and tools for reasoning from complicated premises to their consequences. The invention of the telescope and of the calculus played crucial roles at one stage in the history of physics, and the invention of the cyclotron and of quantum mechanics at a later stage. New observing instruments made visible the previously invisible phenomena that had to be known if the theory was to advance. New analytic instruments made comprehensible the facts revealed by the telescope and cyclotron, which otherwise would have been inscrutable. A science of complex phenomena needs powerful machines for observing and powerful tools for reasoning.

The phenomena of human thinking are more complex than the phenomena that physics studies. In some respects the former are easier to observe than the latter--human verbal behavior is present and audible, neither submicroscopic nor as distant as the stars. (To be sure, we do not have instruments for observing cerebral events that are nearly as revealing as the instruments of physics.) But observable or not, human behavior has not been easy to interpret. We have had great difficulties in building successful theories to explain it.

Until a decade ago, the only instruments we had for building theories about human behavior were the tools we

borrowed and adapted from the natural sciences: operationalism and classical mathematics. And so inadequate are these tools to the task that a highly respected psychologist offered in earnest the doctrine that we must build a science without a theory--surely a doctrine of desperation.

With the advent of the modern digital computer and the emergence of the concept of a program the situation is altered radically. The computer was invented as a machine to do arithmetic rapidly. But as matters turned out, a machine to do arithmetic was a machine that could manipulate symbols. It was natural to ask whether such a machine could perform some of the more general symbol-manipulating processes required for thinking and problem solving as well as the very specialized processes required for arithmetic. The answer, as we shall see, is "yes." There is now substantial evidence, which we shall review, that a digital computer, appropriately programmed, can carry out complex patterns of processes that parallel exceedingly closely the processes observable in human subjects who are thinking.

But the significance of the computer does not lie solely in its ability to exhibit humanoid behavior. It lies even more in the fact that we can specify with complete rigor the system of processes that make the computer exhibit this behavior--we can write a program that constitutes a theory of the computer's behavior in literally the same sense that the equations of Newtonian dynamics constitute a theory of the motions of the

solar system. The genuinely new analytic instrument available for explaining human behavior is the program.

Thinking is to be explained by writing a program for a thinking process. If the program is complicated--as it usually is--so that it is hard to predict what behavior it will produce, we code the program for a computer. Then we compare the behavior of the computer so programmed with the behavior of a human subject performing the same tasks. Thus, the programming language provides a precise language for expressing theories of mental processes; the computer provides a powerful machine for grinding out the specific behavioral consequences of the theories, and for comparing these consequences in detail--sentence by sentence--with the verbal behavior of human subjects.

The methodology provides a powerful test of the sufficiency of the theories. For if a program is vague or incomplete, the computer will not operate--it will not do what we assert it will do. Conversely, if we are able to write a program that, realized on a computer, simulates human behavior closely, we can assert that we have discovered a set of mechanisms at least <u>sufficient</u> to account for the behavior. No dark corners are left in which vitalism or mysticism can lurk--nor even the vagueness of "mediational" hypotheses.

These are large claims. It is time to present the evidence for them. We can only do this, however, after stating a little more fully what we mean by an explanation of

behavior and how a computer program can constitute an explanation of the processes of human thinking and problem solving.

2. WHAT IS AN EXPLANATION?

To explain a phenomenon means to show how it inevitably results from the actions and interactions of precisely specified mechanisms that are in some sense "simpler" than the phenomenon itself. Thus, a chemical reaction is explained by reducing it to the interactions of atoms having specified properties. A spinal reflex is explained by reducing it to a sequence of neural and synaptic processes.

For complex phenomena there may be, and usually are, several levels of explanation; we do not explain the phenomena at once in terms of the simplest mechanisms, but reduce them to these simplest mechanisms through several stages of explanation. We explain digestion by reducing it to chemical events; we explain the chemical reactions in terms of atomic processes; we explain the atomic processes in terms of the interactions of subatomic particles. Every flea has its little fleas, and the scientist's view accepts no level of explanation as "ultimate."

Programs explain behavior in terms of an intermediate level of mechanism, simpler than the behavior itself but more complex than neural events. The intermediate mechanisms provide a theory of the behavior, and provide also a starting point for the next stage of reduction--either to neural events or

to still another level of mechanism above the neurological.

Concretely, human thinking is to be explained in terms of precisely specified simple mechanisms called elementary information processes. Elementary information processes are organized into complex processes--thinking, problem solving, verbal behavior--by programs. Programs are long, branching sequences of elementary processes. In the course of behavior, at each branch point a particular continuation is selected and followed conditionally on the outcome of a simple test (itself an elementary information process) of the identity or difference of a pair of symbols.

In summary, the study and explanation of complex human behavior is to proceed as follows:

Behavior is to be explained by specifying programs that will, in fact, produce the behavior. These programs consist of systems of elementary information processes.

Elementary information processes are to be explained by showing how they can be reduced to known physiological processes in the central nervous system and its appendages.

Since we are concerned here with only the first of these two tasks of explanation--the reduction of behavior to information processes--what guarantee exists against introducing vitalism by the back door? What is to prevent one of the elementary processes from harboring some kind of elan vital? Since explanation at the second level has not been carried out, we cannot guarantee directly that the human nervous

system contains mechanisms capable of performing each of the elementary information processes. But we can insist that there exists some mechanism--a mechanism that can be explained completely at the level of physics--capable of performing all these processes. We can demand that the processes and the programs constructed of them be realized in a digital computer. If the computer executes the processes and, in executing them, simulates human thinking, then no vitalistic mystery can be hidden in the postulates.

We are not talking of a crude analogy between the nervous system and computer "hardware." The inside of a computer does not look like a brain any more than it looks like a missile when it is calculating its trajectory. There is every reason to suppose that simple information processes are performed by quite different mechanisms in computer and brain, and we shall sometime want to know what the brain mechanisms are as we now know the mechanisms of the computer. However, once we have devised mechanisms in a computer for performing elementary information processes that appear very similar to those performed by the brain (albeit by quite different mechanisms at the next lower level), we can construct an explanation of thinking in terms of these information processes that is equally valid for a computer so programmed and for the brain.

3. PROGRAMS AS EXPLANATIONS

We have described a program as a conditionally branching sequence of elementary information processes. To explain a behavior path by a program is quite analagous to explaining the path of a planetary system by a system of differential equations. The differential equations determine what will happen next (during the next "infinitesimal" interval of time) as a function of the exact state of the system at the beginning of the interval. The program determines what the mechanism will do next as a function of its exact state at the moment--this state being dependent, in turn, on the previous history of the system and its current environment.

How is the "right" program discovered--the one that explains the behavior? Proceed the same way that you would to find correct theory for any phenomena. One recipe is this: tape-record some human subjects who are thinking aloud while solving problems (make observations of the phenomena); try to write a computer program that you think will simulate the human protocols (formulate some differential equations); realize the program on a computer, and determine what behavior path it could follow when confronted with the same problems as the human subjects (integrate the equations numerically); compare the simulated with the actual behavior (compare the predictions with the data); modify the program on the basis of the discrepancies that are discovered (modify the equations). Repeat until you are satisfied with the fit.

A number of investigators have independently proposed this general path to the explanation of higher mental processes, and its origins can be traced back at least to Ach and the Wurzburg School. In recent times, perhaps the most explicit examples are to be found in de Groot's investigation of the thought processes of chess players [4] and in A Study of Thinking by Bruner and his associates [3]. Bruner uses the term strategy, borrowed from the mathematical theory of games, for what we have called a "program." What the digital computer and the techniques of programming add is the machinery that gives us hope of following this path, not merely in principle and in general, but in fact and in detail.

4. COMPUTER SIMULATION OF A PROGRAM

The methodology outlined above requires that a computer simulate the sequence of verbal utterances of a human subject (or other symbolic behavior, such as button pushing). It is easy to understand how this can be done once we recognize that computers fundamentally have nothing to do with numbers. It is only by historical accident that we perceive computers as mechanisms for manipulating numerical symbols. They are, in fact, extremely general devices for manipulating symbols of any kind; and the elementary processes required to simulate human thinking could be performed by a computer that had no special capacity for rapid arithmetic--that could do no more than simple counting. The programs we shall

describe make no essential use of the computer's arithmetic processes.

What processes can a general-purpose computer perform? Some of the crucial ones are these:

It can read a symbol--transform a symbol presented to its input mechanisms into a different representation of that symbol in its internal storage (transform a pattern of holes on a punched card into a pattern of magnetism in core storage). The relation between the external and internal representation is quite flexible, almost arbitrary, and can be altered by the program.

It can move a symbol--reproduce in a storage location a symbol that is present in another storage location, with or without a change in the form of representation.

It can generate a symbol--create and store a pattern in one of its modes of internal representation.

It can compare two symbols--executing one program step if they are identical, but a different one if they are not.

It can associate two symbols, allowing access to one symbol (the associated symbol) when the other is given.

Programs can be written that combine these simple processes into processes that are slightly more complex. For example, a computer can be programmed to manipulate a series of symbols as a list, so that it can perform such operations as: "Put this symbol at the end of that list," or "Find a symbol on this list which is identical with that symbol." Such

list structures, and the processes for operating on them, have many resemblances to human memory and association.

Finally, still more complex programs can be composed that enable a computer to respond to instructions like: "Solve the problems on the following list," "Print out the steps of the proof, giving the justification for each step," "Print out the processes used at each step of the problem-solving process: the methods, what is being noticed and attended to, what plans are formed, what subproblems are created." When this last stage has been reached, the trace that the computer prints out while it is attempting to solve the problem can be compared, line by line, with the tape recording of the human thinking-aloud protocol. If the stream of words produced by the two processes is almost the same, then the computer program that produced the trace is an explanation of the thought process of the human subject in every significant sense of the word.

5. TESTING PROGRAMS AS THEORIES

The phrase "almost the same" glides over the whole problem of goodness of fit. Unfortunately, existing statistical theory offers no solution to the problem in the situation we have described, nor can we propose any simple answer. A rough and ready answer is that the evidence provided by five minutes of thinking aloud and the corresponding trace is so voluminous as to scarcely leave room for doubt whether a first approximation has been achieved or not. This is a subjective answer, and we

should like to discuss a slightly more objective, though weaker one:

6. TURING'S TEST

No two human subjects solving the same problem will have the same program or produce the same protocol. Hence, any single program can only be a precise theory of the behavior of a single subject. There must be, however, close qualitative similarities among the programs and protocols of appropriately selected classes of subjects--if not, then it is meaningless to speak of a theory of human problem solving. Suppose that we mix ten traces of computer programs and ten human protocols in an urn. Suppose that a properly qualified human observer is unable to separate, with more than chance success, the protocols produced by the computer programs from those produced by the humans. Then we shall say that the programs which produced the computer traces pass Turing's test*, and provide a satisfactory explanation of the human protocols.

Turing's test can be applied in stronger or weaker forms. Comparison of the move chosen by a chess program with the moves chosen by human players in the same position would be

*A test of this sort was first proposed by A. M. Turing [17] in a discussion of whether a machine could think. Given two communication channels (say teletypes), one connected to a human, the other to a machine, a human interrogator was to identify which channel was the machine's. Active questioning was allowed, and the machine's problem was to fool the interrogator, despite the best efforts of the human on the other channel (who it was assumed would side with the interrogator) to reveal his identity.

a weak test. The program might have chosen its move by quite a different process from that used by the humans. For the task environment itself defines what are appropriate behaviors, and any mechanism capable of behaving adaptively in the environment might be expected to exhibit about the same external behavior. Similarity of function does not guarantee similarity of structure, or of process.

If data are gathered, however, by the thinking-aloud technique or by other means, that indicate the processes used to select the behavior, it may, and usually will be possible to distinguish different ways of arriving at the result. If the program makes the same analysis as the humans, notices the same features of the board, overlooks the same traps, then we will infer, and properly, that down to some level of detail, the program provides an explanation of the human processes. The more minute and detailed the comparison between program and behavior, the greater will be the opportunities for detecting differences between the predicted and actual behaviors.

This method of theory building and testing meets the problem of induction no better and no worse than other methods. There never is, and can never be a guarantee that some other theory will not explain the data equally well or better. As in other sciences, it will be time to face this problem when someone actually proposes an alternative theory that explains the data equally well and in comparable detail. Meanwhile, the validity of programs as theories can be tested in stronger

and stronger form by pushing the level of detail of matching down toward the level of elementary information processes.

7. THE EXISTING STATE OF THE ART

Several computer programs in existence at the present time can more or less lay claim to being theories of certain kinds of human problem-solving behavior--either by Turing's test or by more stringent criteria. A brief and partial inventory of such programs will give a better picture of where matters stand.

Not every computer program for performing complex tasks constitutes a theory of human problem solving. A program that solves large problems by relying substantially on the arithmetic speed and 'brute force' of the computer in performing systematic routine calculations is certainly not a simulation of the program that humans use in solving similar problems. Most of the programs, for example, that have been written for solving operations research problems, and which incorporate such mathematical techniques as linear programming, solution of differential equations, and the like, fall in this category. We will have nothing to say about these, for they were not constructed as simulations of human problem solving, and make no claims as theories of human thinking.

In general, computational routines of the kinds just mentioned tend to use systematic, arithmetic procedures (impracticable without the rapid "inhuman" arithmetic processes in modern

computers), which are usually called algorithms. The programs that simulate human thinking tend to rely on less systematic, more selective search for paths to the solution. Their selectivity is based on relatively unsystematic rules of thumb, which seldom guarantee a solution to the problem, but which frequently yield solutions with relatively little processing. They make no use of the fast arithmetic processes available to the computer. We call such procedures heuristics. There is no hard-and-fast line between algorithms and heuristics, but it is easy to point to clearcut examples of each.

We will comment briefly on some examples of types of problem-solving programs that are relevant for the simulation of thinking. The first three that are mentioned--programs for musical composition, for playing games, and for making business and engineering decisions--have generally been constructed with the goal of performing the task and without any special concern for human simulation. The same may be said, to a lesser extent, of the logic and geometry programs. However, many of these programs have a strong heuristic, rather than algorithmic, flavor, because it happens to be easier to solve the problems in question by imitating human tricks than by using the arithmetic speed of the computer.

The other two categories of programs: the general problem solver, and programs for "simple" processes, were written as direct attempts to simulate human processes. Most of our discussion will center on them.

A Program for Musical Composition. In 1956, Hiller and Isaacson at the University of Illinois programmed the ILLIAC to compose music [6]. The music has been judged tolerable by some modern ears, but could not pass Turing's test before an audience of musicians. Its interest lies in the fact that its rules of counterpoint approximate those of Palestrina.

The Logic Theorist. In 1956, J. C. Shaw and the authors programmed JOHNNIAC at The RAND Corporation to discover proofs for theorems in the Principia Mathematica of Whitehead and Russell [9,14]. The program, in several variants, was able to prove about 70% of the theorems in Chapter 2 of Principia. The program, distinctly heuristic in character, was derived largely from the introspections of its authors. Some learning programs were later incorporated in it that definitely modified its behavior, usually in an adaptive way. It could certainly not pass Turing's test if compared with thinking-aloud protocols, although many of its qualitative features match those of human problem solvers [10].

Game-Playing Programs. A program for checkers, written by A. Samuel, plays a strong game and incorporates some powerful learning programs, but cannot be considered a simulation of human checker-playing--it explores far too many continuations. Chess programs written by M. Wells and his associates at Los Alamos [7], and by A. Bernstein and his associates at IBM [1], can beat novices but not passable amateurs. These programs

explore large numbers of continuations (800,000 per move in the Los Alamos program, and about 2,500 per move in the IBM program), although the latter also incorporates certain heuristics for finding plausible moves that resemble those used by human chess players.

A third chess player has been programmed by J. C. Shaw and the authors [11], incorporating about the same amount of selectivity as the program of a strong human player. The first aim has been to write a program that would play good chess, with simulation as a secondary objective. The program plays a game about as strong as the other two, with much more selectivity in its explorations, but could not yet pass Turing's test. (Parenthetically, the attraction of chess for these researches lies in the fact that it is a game of sufficient complexity and irregularity that a heuristic rather than an algorithmic approach is almost certainly required for strong play.)

Business and Engineering Decisions. Programs have been written, with purely pragmatic aims, for preparing payrolls, maintaining inventories and making purchases, designing electric motors and transformers, and the like, which incorporate many heuristic elements and which are probably not very dissimilar from the human processes they replace [2]. None of these programs has been systematically investigated for the light it might throw on human processes; probably none could pass Turing's test. Currently, an explicitly heuristic

program is being written by F. Tonge [16] to handle an assembly line balancing problem that has proved too complex for existing algorithms.

Geometry. A theorem prover for geometry, developed by Gelernter and Rochester of IBM [5], produced its first complete proof in April 1959. It probably will compete in skill with high school geometry students, and incorporates a number of heuristics used by humans--although fewer than are used by a skillful student. It will therefore not pass Turing's test.

The General Problem Solver. The Logic Theorist was superseded in 1957-58 by the General Problem Solver, which has been hand simulated but not yet fully tested on the computer. This new program will be described more fully in the next section. Its heuristics are based on general ideas for reasoning about problems in terms of means and ends, and are not specific to logic or any other subject matter. It was derived largely from analysis of human protocols, and probably can pass Turing's test for a limited range of tasks.

"Simple" Human Behaviors. The work on the Logic Theorist led to conjectures that certain "simple" psychological tasks, like responding in a partial reinforcement experiment or memorizing nonsense syllables actually involve a great deal of problem-solving behavior. These conjectures are now being explored by J. Feldman, in a program for a binary choice

experiment, and by E. Feigenbaum in an EPAM (Elementary Perceiver and Memorizer) program. These programs are being designed to simulate the behavior of subjects, in response to experimental instructions, over a range of standard psychological experiments. It is too early to evaluate them beyond saying that they show great promise of explaining the processes involved in recognizing patterns in sequences, distinguishing stimuli and responses of high and low similarity, serial and paired-associate memorizing, and the like. Both programs would have a good chance to pass Turing's test.

8. THE GENERAL PROBLEM SOLVER

To give substance to these generalities, we shall conclude by examining more closely what is perhaps the most advanced of these programs at the present time: The General Problem Solver [12,13]. The General Problem Solver was devised to simulate the behavior of some specific human subjects solving problems in symbolic logic in a task situation devised by O. K. Moore and Scarvia Anderson [8]. Some thirty thinking-aloud protocols for these tasks have been recorded in the laboratory at Carnegie Institute of Technology. Comparison of these data obtained for 64 subjects by Moore without requiring the subjects to think aloud indicates that there are no substantial differences in process under the two conditions.*

GPS, as we shall call the program, is called "general" because it is not limited to the task for which it was originally

* Professor Moore kindly provided us with the full data on his subjects prior to publication.

devised. Hand simulation indicates it can also solve the Whitehead and Russell logic problems, do trigonometric identities, perform formal integration and differentiation, and, with a small extension to the program, solve algebraic equations. As will be seen, there is reason to hope that it can be extended to an even wider range of tasks.

Before comparing GPS in detail with human behavior, we should like to observe that it does solve problems. Hence its program constitutes a system of mechanisms, constructed from elementary information processes, that is a sufficient system for performing certain tasks that humans perform. However much it may prove necessary to modify the details of the program for close human simulation, in its present form it constitutes an unequivocal demonstration that a mechanism can solve problems by functional reasoning.

In simplest terms, GPS is a program for reasoning about means and ends. It grew out of our observation that the protocols of laboratory subjects contained many statements of the following sorts: "So in all these now I have notes as to exactly what I can do with them." (Paraphrase: "Here are some means at my disposal, what ends will they serve?") "I'm looking at the idea of reversing these two things now . . . then I'd have a similar group at the beginning." (Paraphrase: "If I use means X, I will achieve Y.") "I'm looking for a way, now, to get rid of that symbol." ("What is the means to achieve this end?") "And now I'd use Rule 1." ("I'll apply means X.")

Closer scrutiny of the protocols reveals that the vast majority of the statements in them fall within this general framework. Simulating the behavior of these subjects requires a program that can handle problems in this kind of functional language. Further, the functional language makes no reference to the specific subject matter of the problem--in this instance symbolic logic. The program must be organized to separate its general problem-solving procedures from the application of these to a specific task. GPS is such a program.

The adjective "general" does not imply that GPS can reason about all or most kinds of problems; or that it will simulate all or most human problem-solving activity. It simply means that the program contains no reference to the task content, and hence is usable for tasks other than the one for which it was devised.

GPS operates on problems that can be formulated in terms of objects and operators. An operator is something that can be applied to certain objects to produce different objects (as a saw applied to logs produces boards). The objects can be described by the features they possess (boards have flat parallel sides), and by the differences between pairs of objects (a 2x4 is thicker than a 1x4). Operators may be restricted to apply to only certain kinds of objects (nails are used on wood, not steel); and there may be operators that are applied to several objects as inputs, producing one or more objects as output (joining four beams to produce a frame).

Various problems can be formulated in a task environment containing objects and operators: to find how to transform one object into another; to find an object with specified features; to modify an object so that a specified operator can be applied to it; and so on. In the task environment confronting our laboratory subjects, the objects were symbolic logic expressions (which the subjects were told were messages in code). The operators were twelve rules of logic for transforming one or two input expressions into an output expression. Figure 1 gives the first eight of these rules, enough for our illustrative purposes. For example, by Rule 1 an expression of the form (A.B) could be transformed into (B.A), thus reversing the order of the symbols. The problems given the subjects were to "recode," using the rules, one or more given logic expressions into a different logic expression. One problem, for example, was to transform $R\cdot(-P\supset Q)$ into $(Q\vee P)\cdot R$.

A statement like: "I'm looking for a way, now, to get rid of that horseshoe" expresses a goal. The goal in this instance is to eliminate a difference (the "horseshoe" in the original expression versus the "wedge" in the desired expression) between one object and another. The goals that the subjects mention in their protocols take a variety of forms. We have incorporated three types of goals, which account for the vast majority of goal statements, in the present version of GPS. They are:

Goal Type 1: Find a way to transform object a into

object b (i.e., a sequence of operators that will accomplish the transformation).

Goal Type 2: Apply operator q to object a (or to an object obtained from a by transformations).

Goal Type 3: Reduce the difference, d, between object a and object b by modifying a.

The problems initially given the subjects established transform goals; "getting rid of the horseshoe" expresses a reduce goal; "And now I'd use Rule 1" states an apply goal.

1. $A \vee B \iff B \vee A$
   $A \cdot B \iff B \cdot A$

2. $A \supset B \iff \neg B \supset \neg A$

3. $A \vee A \iff A$
   $A \cdot A \iff A$

4. $A \vee (B \vee C) \iff (A \vee C) \vee C$
   $A \cdot (B \cdot C) \iff (A \cdot B) \cdot C$

5. $A \vee B \iff \neg(\neg A \cdot \neg B)$
   $A \cdot B \iff \neg(\neg A \vee \neg B)$

6. $A \supset B \iff \neg A \vee B$
   $A \vee B \iff \neg A \supset B$

7. $A \vee (B \cdot C) \iff (A \vee B) \cdot (A \vee C)$
   $A \cdot (B \vee C) \iff (A \cdot B) \vee (A \cdot C)$

8. $A \cdot B \Longrightarrow A$
   $A \cdot B \Longrightarrow B$

Fig. 1 — Rules for transforming logic expressions

To attain a goal, consideration of the goal must evoke in the problem solver some idea of one or more means that might be relevant. The subject, for example, who says "I'm looking for a way, now, to get rid of that horseshoe," follows this statement with, "Ah . . . here it is, Rule 6. So I'd apply Rule 6 to the second part of what we have up there." Applying

Rule 6 (See Figure 1) has been evoked as a method for getting rid of horseshoes.

Thus, the evoking process is represented in GPS by associating with each type of goal one or more methods for attaining a goal of that type. These are shown in Figure 2. Method 1, associated with Transform goals, consists in: (a) matching the objects a and b to find a difference, d, between them; (b) setting up the Type 3 subgoal of reducing d, which if successful produced a new transformed object, c; (c) setting up the Type 1 subgoal of transforming c into b. If this last goal is achieved, the original Transform goal is achieved. The match in step (a) tests for the more important differences (in terms of some priority list) first.

Method 2, for achieving an Apply goal, consists in: (a) determining if the operator can be applied by setting up a Type 1 goal for transforming a into the input form of the operator q (which we call C(q)); (b) if successful, the output object is produced from the output form of q (P(q)).

Method 3, for achieving a Reduce goal, consists in: (a) searching for an operator that is relevant to reducing the difference, d; (b) if one is found, setting up the Type 2 goal of applying the operator, which if attained produces the modified object.

To see how GPS goes about applying these goal types and methods to the solution of problems, consider a concrete example. We will use the problem mentioned earlier: to

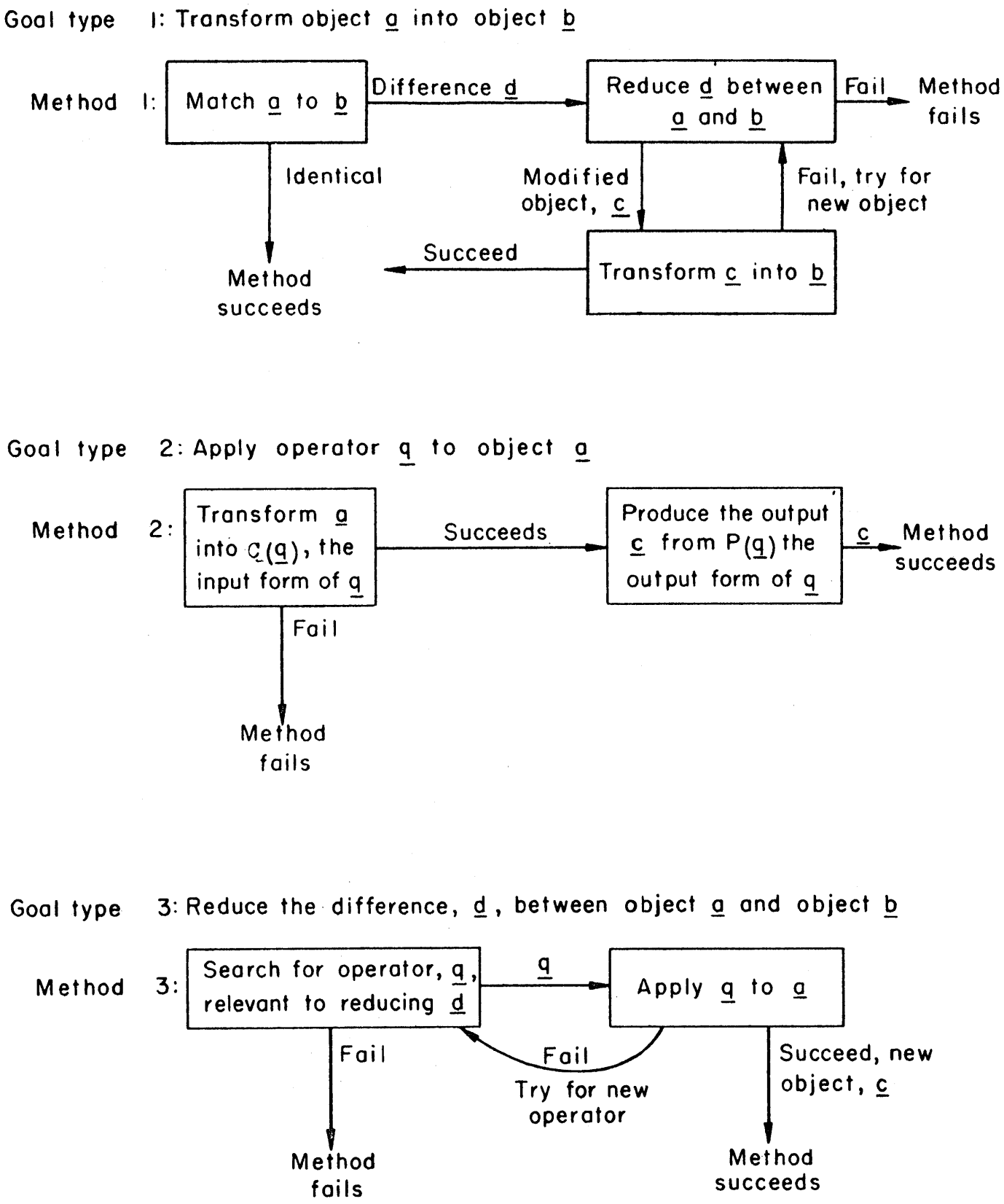

Fig. 2 — Methods for means—ends analysis

"recode" the expression L1, R•(-P⊃Q), into the expression L0, (QvP)•R. As we go along we will explain the recoding rules as far as is necessary to understand the example.

GPS begins by establishing the Type 1 goal of transforming L1 into L0. Among the information processes available to it--built up from the elementary processes, are a number of tests for the possible differences among pairs of expressions; for example:

1. A test whether the same or different variables (letters) appear in the two expressions;
2. A test whether each variable occurs the same or a different number of times;
3. A test whether a variable or group occurs in the same or in a different position;
4. A test whether a pair of connectives (•, v, ⊃) is the same or different.

And so on. The tests may be applied to whole expressions or to corresponding parts of expressions (e.g., (-P⊃Q) and (QvP)).

Method 1 applies these tests in order. It discovers a difference in position of the R's in L1 and L0, and establishes the Type 3 goal of reducing the difference. For each difference, it has available a list of operators that are possibly relevant to removing a difference of that kind. (These lists can be constructed by GPS itself by examining the set of available operators.) In this case, it discovers that Rule 1, which transforms an expression of form (A•B) or (AvB) into an expression of form (B•A) or (BvA), respectively, affects differences in position. Consequently, it establishes the

Type 2 subgoal of applying Rule 1 to L1. This can be done by identifying R with A and (-P⊃Q) with B in the rule, thus producing L2, (-P⊃Q)·R, as the output expression.

This done, the original goal now sets up the new subgoal of transforming L2 into L0. Repeating the cycle, a difference in connectives is found between the left-hand sides of L2 and L0, respectively; a rule, R6, is found that changes connectives by transforming (-A⊃B) into (AvB), hence transforms (-P⊃Q)·R into L3, (PvQ)·R.

A third repetition of the basic cycle discovers the difference in position between (PvQ) in L3 and (QvP) in L0, and applies R1 to remove the difference. Finally, GPS discovers that the product of this transformation is identical with L0, and declares the problem solved. We may summarize the steps as follows:

Step	Expression	Justification for Step
L1	R.(-P⊃Q)	Given
L2	(-P⊃Q)·R	Rule 1
L3	(PvQ)·R	Rule 6 inside parenthesis
L0	(QvP)·R	Rule 1 inside parenthesis

9. COMPARISON WITH HUMAN BEHAVIOR

Granted that GPS can solve this problem, and many that are a good deal more difficult, why do we suppose that the processes of GPS resemble in any way the processes a human would use in solving the same problem? Let us compare GPS's processes, as just narrated, with the content of the protocol of a human subject solving the same problem. We shall let

the reader judge whether the two processes are or are not closely similar. (Neither the particular problem nor the human protocol we shall examine was used in devising GPS.)

In the right-hand half of Table 1 we reproduce, word for word, the human protocol, omitting only some introductory paragraphs. In the left-hand half we reproduce the trace (hand simulated) of a program that, we believe, approximates closely the processes of the subject's thinking. This program is not identical with the one we described for GPS, but incorporates some modifications to fit it to the empirical data. Basically however, the objects, operators, differences, goal types, and methods that appear in this trace are those of GPS.

Scanning down the protocol, we see that the subject sets up the goal of transforming L1 into L0, notices the difference in order of terms in the two expressions, and considers reversing them (lines 4-9). Simultaneously, the program is establishing the same goal (Goal 0), noticing the same difference (line 4), discovering that Rule 1 reduces this difference (line 9), and fixating the idea of applying Rule 1 when the analysis is complete. (This distinction between overt and covert action is one of the modifications introduced.)

Next, the subject scans down the list of rules--there is explicit evidence that he looks at Rules 2, 3, and 4--rejecting each because it doesn't apply (Rule 3), would introduce a new difference ("That would require a sign change"), or doesn't perform the function of switching the P and Q (Rule 4). Simul-

Table 1

SIMULATION 7 OF SUBJECT 9 ON O. K. MOORE PROBLEM α1.

Line	Simulation	Protocol
1	L0: (QvP)·R	(Expression to be obtained)
2	L1: R·(-P Q)	(Expression given at start)
3	Goal 0: Transform L1 into L0	(Goal set by experimenter)
4	Match gives position difference (Δp)	I'm looking at the idea of reversing these two things now.
5	Goal 1: Reduce Δp between L1 and L0	(Thinking about reversing what?)
6	Search list of rules	The R's . . .
7	Goal 2: Apply R1 to L1	
8	Match: R1 applicable	
9	Test rule functions: reduces Δp	then I'd have a similar group at the beginning
10	no others	but that seems to be ...
11	Set to execute R1 when analysis complete	I could easily leave something like that 'til the end,
12	Goal 3: Transform right L1 into left L0	except then I'll . . .
13	Match gives position difference (Δp)	
14	Goal 4: Reduce Δ p between right L1 and left L0	
15	Search list of rules	
16	Goal 5: Apply R1 to right L1	
17	Match: R1 fails, right L1 has (Δc)	(Applying what rule?)
18	Goal 6: Apply R2 to right L1	Applying, . . . for instance, 2.
19	Match: R2 applicable	
20	Test rule functions: reduces Δp,	
21	but introduces unwanted - (Δs)	That would require a sign change.
22	Reject goal	(Try to keep talking, if you can)
23	Goal 7: Apply R3 to right L1	Well . . . then I look down at rule 3
24	Match: R3 not applicable	and that doesn't look any too practical
25	Goal 8: Apply R4 to L1	Now 4 looks interesting.
26	Match: R4 not applicable*	It's got three parts similar to that .. and ... there are dots

* But the subject mistakenly thinks R4 is applicable; therefore tests its functions.
Statements in parentheses are experimenter's statements and explanatory comments.
All other statements are the subject's.

Simulation 7, Subject 9, O. K. Moore Problem 1. Table 1 (cont.)

27		so the connective ... seems to work easily enough,
28	Test rule functions: doesn't reduce Δp	but there's no switching of order.
29	Reject goal	
30	Goals 9 to 13: Apply R5 to R9 to L1	
31	All goals fail on match	
32	or test of function	
33	Search rules again, but don't reject	I need that P and a Q changed, so . . .
34	without attacking subproblem	
35	Goal 14: Apply R1 to right L1	
36	Match: R1 fails, right L1 has (Δc)	I've got a horseshoe there. That doesn't seem
37	Test rule functions: reduce Δp, no others	practical any place through here.
38	Set to execute R1, if applicable	
39	Goal 15: Reduce Δc between right L1 and R1	I'm looking for a way now, to get rid of that horseshoe.
40	Search list of rules, for rule with	
41	that reduces Δc.	
42	Goal 16: Apply R6 to right L1	Ah . . . here it is, Rule 6.
43	Match: R6 applicable	
44	Test rule functions: reduces Δc	
45	reduces Δs	
46	Set to execute R6 when analysis complete	
47	Goal 17: Transform right L1, after reducing	
48	Δp, Δc, Δs, into left L0	
49	Match: transformed right L1 identical with	
50	left L0	
51	Goal achieved, and analysis complete	
52	Execute R6 on right L1 (from line 46)	So I'd apply Rule 6 to the second part of what we
53		have up there.
54		(Want to do that?)

Simulation 7, Subject 9, O. K. Moore Problem α1. Table 1 (cont.)

55		Yeah.
56	L2: R·(PvQ)	(OK, to line 1 you apply R6. Line 2 is R·(PvQ)
57	Execute R1 on right L2 (from line 38)	And now I'd use Rule 1.
58		(Rule 1 on what part? You can use it with the
59		entire expression or with the right part.)
60		I'd use it in both places.
61		(Well, we'll do them one at a time ... which do
62		you want first?)
63		Well, do it with P and Q.
64	L3: R·(QvP)	(R·(QvP). Now the entire expression?)
65	Execute R1 on L3 (from line 11)	Yeah.
66	L4: (QvP)·R	(On line 3, Rule 1 ... you'd get (QvP)·R)
67	Match: L4 identical with L0	And ... that's it.
		(That's it all right, OK ... that wasn't too
		hard.)

taneously, the program is establishing the goal or reducing the difference in order of P and Q (Goal 4), scanning the list of rules, and rejecting them for the same reasons.

Next, the subject observes that the horseshoe creates difficulty in changing the P and Q (lines 36 to 37), and erects the goal of eliminating the horseshoe, discovering that Rule 6 will do this (line 42). He applied Rule 6, then Rule 1 to the right-hand subexpression, then Rule 1 to the whole expression, and observed that he has solved the problem. Simultaneously, the program undertakes a second search for a rule that will reverse P and Q (lines 33-34), but now tackles as a subproblem getting an otherwise relevant rule to be applicable. Considering Rule 1, it establishes the subgoal (Goal 15) of changing the horseshoe to a wedge, finds Rule 6, checks whether this will solve the problem, then executes Rules 6, 1, and 1 (on the whole expression) in that order.

There are a number of interesting points of fine structure in the comparison of program with protocol that we cannot go into here. For example, the program searched all the rules the first time through, while the protocol of the subject gives evidence for only the first few. The cause of the discrepancy is not at all clear. The reader can use Table 1 to discover and examine a number of such features.

In testing whether this program provides a good theory, or explanation, of the behavior of the human subject, we can raise two kinds of questions:

(a) How much did we have to modify GPS to construct a program that would fit this subject's protocol?

(b) How good is the fit of the modified program to the protocol?

In the next paragraphs we will discuss some of the methodological issues that are imbedded in these questions. We are, of course, considering a test of simulation that is much stronger than Turing's test, and perhaps the reader can be persuaded from the evidence presented thus far that GPS and its variants will pass Turing's test.

Fitting a General Program to Specific Behavior. The form the theory takes is a program--in this case, GPS. But, as in the natural sciences, the theory is more appropriately expressed as a class of programs to be particularized and applied to concrete situations by specifying parameter values, initial conditions, and boundary conditions. For example, one subject may attach higher priorities to a difference in variables, another to a difference in connectives. GPS is fitted to these subjects' behaviors by modifying the program to represent the difference in priorities--this is precisely what we did in constructing the example shown in Table 1.

As in all fitting of theory to data, we must watch our degrees of freedom. If we are allowed to introduce a parameter change or a new mechanism for each bit of behavior to be explained, we will have explained nothing. The program must be a parsimonious description of the mechanisms generating the

behavior. Let us point out--to be sure the reader understands--that the trace is the output of the program, and not modifiable at will. Any change in the program affects the trace in a number of places. A chance to reduce one discrepancy--say the apparently more exhaustive initial search of the list of rules mentioned above--is likely to introduce new discrepancies. In the case in point, we were unable to find any simple change in the program that would remove this particular discrepancy, and yet leave the rest of the fit as good as it is.

Comparing a Trace with Behavior. The computer does not yet speak fluent idiomatic English; hence we cannot compare the trace with the subject's protocol literally word for word. The trace says: "Goal 1: Reduce Δp between L1 and LO." The subject says: "I'm looking at the idea of reversing these two things now." The trace says: "Goal 6: Apply R3 to L1." The subject says: " ... then I look down at Rule 3." Instead of having the computer speak English, we could hope for a code (in the psychologist's sense of the word) that would reduce the human conversation to "problem-solving content." Again, techniques are currently lacking for doing this, but perhaps we can agree that a large amount of the content of the subject's remarks is captured in the computer's "phases."

Further, the trace describes the information processes uniformly down to a specified level of detail, while the protocol fluctuates greatly in its explicitness--sometimes providing more detail, but usually much less. Thus in our search example, it is certainly possible that the subject scanned the entire set of rules, but simply failed to mention them after the

first few. This kind of mismatch is probably inevitable, at least in the present state of our knowledge. The most we can aim for is a trace that avoids sins of omission and contradiction, although it may sometimes speak when the subject remains silent.

We do not mean that the level of detail in the protocol is arbitrary--in fact, we suspect that it is very much related to the mechanisms and functions of consciousness in problem solving and learning. The distinction between conscious and unconscious does not appear in the present GPS mechanisms, and hence cannot be reflected in a non-arbitrary way in the trace.

The exhibit presented here provides a sample of the work that has been done in comparing GPS, and variants of it, with the problem-solving behavior of human subjects. The protocol shown here covers only four minutes of behavior. Relatively little simulation has been carried out to date--it is a painstaking activity. Our most intensive empirical study thus far has been the simulation of the behavior of a subject solving a considerably more difficult problem over a period of 30 minutes. We shall report on our findings in more detail elsewhere, but the following conclusions, from this intensive study, and from less detailed examination of about twenty other protocols seem reasonably certain:

Measured in terms of time and numbers of words, virtually all the behavior of subjects falls within the general framework of means-end analysis. The three goal types we have described account for at least three fourths of the subjects' goals, and

the additional goal types for which we find evidence are close relatives to those we have mentioned.

The three methods we have described represent the vast majority of the methods applied to these problems. One additional exceedingly important method--planning--has been incorporated in GPS [13], but limits on time prevent us from discussing it here. Planning appears in several different forms in the protocols, but in all of them it serves the function of temporarily omitting details in order to see if the main line of reasoning will yield a solution.

There are evidences that the programs of the subjects change--that they learn--in the course of problem solving. For example, initially they have to scan the rules, one by one, to find an applicable rule; later, once they create the goal of reducing a specified difference, they choose almost instantaneously a rule that is relevant to that particular difference. No clear distinction between learning and problem solving appears. Some of the learning takes place--and is used--in the course of attempting a single problem (one-half hour). In fact, the GPS network of goals and subgoals constitutes a "learning about the problem," so that on successive phases of the solution, the subject behaves very differently. Conversely, some of the learning occurs as the result of specific problem-solving activity devoted to learning.

10. CONCLUSION

In this Research Memorandum we have described a method for the study of human problem solving and other higher mental processes, have given an example of an application of the method, and have indicated the theory of human problem solving that emerges. We have drawn our examples largely from the work of ourselves and our associates, because it represents the only current undertaking with which we are familiar that includes detailed simulation of human protocols. There are, of course, other investigators who are exploring human problem solving, on the one hand, and simulation techniques, on the other.

The method consists in constructing a theory of central processes in the form of a program, or class of programs, demonstrating the sufficiency of the theory to produce problem-solving behavior by realizing it in a computer, and testing the theory against human processes by comparing the trace generated by the program with the protocol of a human subject.

The application consisted in constructing a general problem-solving program, capable of solving problems in logic and other domains, demonstrating that a computer so programmed could solve problems, and comparing its processes with those of human subjects in a problem situation designed by O. K. Moore.

The theory of human problem solving consists in a program, constructed of elementary information processes, for reasoning in terms of goals and methods for attaining those goals. Perhaps the most striking characteristic of this program is

that it selects the paths it explores by first determining the functions that have to be performed, and then finding courses of action relevant to those functions. In this and other ways it reflects (and incorporates in determinate mechanisms) the "insightfulness" and "directedness" that has so often been observed as a salient characteristic of human problem solving. In terms of today's nomenclature in psychology, one could describe it as a "mediational" theory that encompasses "Gestalt" processes. Its novelty is that it is definite and that, at least in one problem area, it works.

It is easy to point to difficulties and unfinished tasks. Systematic methods for fitting programs to protocols and testing goodness of fit are nonexistent. The "General" Problem Solver is still highly specific compared with the humans it simulates. The construction and testing of learning programs has hardly begun. Only the most rudimentary programs for simulating "simple" human processes have been written, and these have not been tested. There is little information for selecting the correct set of elementary processes; and even less for connecting them with neural mechanisms.

In spite of this imposing agenda of unfinished business, we wish to record our conviction that it is no longer necessary to talk about the theory of higher mental processes in the future tense. There now exist tools sharp enough to cut into the tough skin of the problem, and these tools have already produced a rigorous, detailed explanation of a significant area of human symbolic behavior.

REFERENCES

1. Bernstein, A. and M. deV. Roberts, Computer vs. Chess Player, Scientific American, 198, 6, June 1958.

2. Brenner, W. C., R. Schinzinger and R. M. Suarez, Application of High Speed Electronic Computers to Generator Design Problems, AIEE Conference Paper 56-940, 1956.

3. Bruner, J. S., J. J. Goodnow and C. A. Austin, A Study in Thinking, Wiley, 1956.

4. deGroot, A. D. Het Denken van den Schaker, Amsterdam, 1946.

5. Gelernter, H. L. and N. Rochester, Intelligent Behavior in Problem-Solving Machines, IBM Journal of Research and Development, 2, 4, October, 1958.

6. Hiller, L. A. Jr., and L. M. Isaacson, Experimental Music, McGraw-Hill, 1959.

7. Kister, J., et al., Experiments in Chess, J. Assoc. for Computing Machinery, 4, 2, April, 1957.

8. Moore, O. K. and S. B. Anderson, Modern Logic and Tasks for Experiments on Problem-Solving Behavior, J. of Psychology, 38, 151-160, 1954.

9. Newell, A., J. C. Shaw and H. A. Simon, Empirical Explorations of the Logic Theory Machine, Proceedings of the Western Joint Computer Conference, IRE, February, 1957.

10. Newell, A., J. C. Shaw and H. A. Simon, The Elements of a Theory of Human Problem Solving, Psych. Rev., 65, March 1958.

11. Newell, A., J. C. Shaw and H. A. Simon, Chess-playing Programs and the Problem of Complexity, IBM Journal of Research and Development, 2, 4, October, 1958.

12. Newell, A., J. C. Shaw and H. A. Simon, The Processes of Creative Thinking, The RAND Corporation Paper, P-1320, August, 1958.

13. Newell, A., J. C. Shaw and H. A. Simon, Report on a General Problem-Solving Program, The RAND Corporation Paper, P-1584, January, 1959.

14. Newell, A., and H. A. Simon, The Logic Theory Machine, Trans. on Information Theory, Vol. IT-2, No. 3, September, 195

15. Osgood, C. E., Method and Theory in Experimental Psychology, Oxford, 1953.

16. Tonge, F. M., Development of a Heuristic Program for an Assembly Line Balancing Problem, February, 1958, unpublished.

17. Turing, A. M., "Can a Machine Think?," in J. R. Newman's The World of Mathematics, Vol. 4, Simon and Schuster, 1956.