THE PROCESSES OF CREATIVE THINKING

A. Newell
J. C. Shaw
H. A. Simon*

P-1320

September 16, 1958

Revised January 28, 1959

Presented at a Symposium on Creative Thinking, University of Colorado, Boulder, Colorado, May 16, 1958.

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

*Carnegie Institute of Technology

Summary

We ask first whether we need a theory of creative thinking distinct from a theory of problem solving. Subject to minor qualifications, we conclude there is no such need — that we call problem solving creative when the problems solved are relatively new and difficult. Next, we summarize what has been learned about problem solving by simulating certain human problem solving processes with digital computers. Finally, we indicate some of the differences in degree that might be observed in comparing relatively creative with relatively routine problem solving.

TABLE OF CONTENTS

THE PROCESSES OF CREATIVE THINKING

Allen Newell, J. C. Shaw, and H. A. Simon*

What is meant by an "explanation" of the creative process? In the published literature on the subject, the stages of thought in the solution of difficult problems have been described, and the processes that go on at each stage discussed. Interest has focussed particularly on the more dramatic and mysterious aspects of creativity — the unconscious processes that are supposed to occur during "incubation," the imagery employed in creative thinking and its significance for the effectiveness of the thinking, and, above all, the phenomenon of "illumination," the sudden flash of insight that reveals the solution of a problem long pursued. Experimental work — to the limited extent that it has been done — has been most concerned with directional set, including the motivational and cognitive conditions that produce set and that alter set, and interpersonal differences in "inappropriate" persistence of set (stereotypy).

All of the topics we have mentioned are interesting enough, and are appropriate parts of a theory of creative thinking. In our own orientation to creativity, however, we have felt the need for a clearer idea of the overall requirements and aims of such a theory. We propose that a theory of creative thinking should consist in:

*Carnegie Institute of Technology

1. Completely operational specifications* for the behavior of mechanisms (or organisms) that, with appropriate initial conditions, would in fact think creatively;

2. a demonstration that mechanisms behaving as specified (by these programs) would exhibit the phenomena that commonly accompany creative thinking (e.g., incubation, illumination, formation and change in set, etc.);

3. a set of statements — verbal or mathematical — about the characteristics of the class of specifications (programs) that includes the particular examples specified.

Stated otherwise, we would have a satisfactory theory of creative thought if we could design and build some mechanisms that could think creatively (exhibit behavior just like that of a human carrying on creative activity), and if we could state the general principles on which the mechanisms were built and operated.

When it is put in this bald way, these aims sound utopian. How utopian they are — or rather, how imminent their realization — depends on how broadly or narrowly we interpret the term "creative." If we are willing to regard all human complex problem solving as creative, then — as we shall point out — successful programs for problem-solving mechanisms that simulate human problem solvers already exist, and a number of their general characteristics are

---

*As we shall explain later, we propose that such a set of specifications take the form of a program, as that term is used in the digital computer field, and we will henceforth refer to them as "programs."

known. If we reserve the term "creative" for activities like discovery of the special theory of relativity or composition of Beethoven's Seventh Symphony, then no example of a creative mechanism exists at the present time.

However, the success already achieved in synthesizing mechanisms that solve difficult problems in the same manner as humans is beginning to provide a theory of problem solving that is highly specific and operational. The purpose of this paper is to draw out some of the implications of this theory for creative thinking. To do so is to assume that creative thinking is simply a special kind of problem-solving behavior. This seems to us a useful working hypothesis.

We start by discussing the relation of creative thinking to problem solving in general, and by inquiring to what extent existing problem-solving programs may be considered creative. Next we sketch the theory of problem solving that underlies these programs, and then use the theory to analyze the programs, and to compare them with some human problem-solving behavior exhibited in thinking-aloud protocols of subjects in the laboratory. Finally, we consider some topics that have been prominent in discussions of creativity to see what this analysis of problem solving has to say about them.

Problem Solving and Creativity

In the psychological literature, "creative thinking" designates a special class of activities, with somewhat vague

and indefinite boundaries (see, e.g., Johnson, pp. 166-167). Problem solving is called creative to the extent that one or more of the following conditions are satisfied:

1. The product of the thinking has novelty and value (either for the thinker or for his culture).

2. The thinking is unconventional, in the sense that it requires modification or rejection of previously-accepted ideas.

3. The thinking requires high motivation and persistence: either taking place over a considerable span of time (continuously or intermittently), or occurring at high intensity.

4. The problem as initially posed was vague and ill-defined, so that part of the task was to formulate the problem itself.

Vagueness of the Distinction

A problem-solving process can exhibit all of these characteristics to a greater or lesser degree, but we are unable to find any more specific criteria separating creative from non-creative thought processes. Moreover, the data currently available about the processes involved in creative and non-creative thinking show no particular differences between the two. We may cite, as example, the data of Patrick (11,12) on the processes involved (for both professionals and amateurs) in drawing a picture or writing a poem, or the data of de Groot (1) on the thought processes of chess players. Not only do the processes appear to be remarkably similar from one task to another — agreeing well with Wallas' (16) account

of the stages in problem solving — but it is impossible to distinguish, by looking solely at the statistics describing the processes, the highly skilled practitioner from the rank amateur.

Similarly, there is a high correlation between creativity (at least in the sciences) and proficiency in the more routine intellective tasks that are commonly used to measure intelligence. There is little doubt that virtually all the persons who have made major creative advances in science and technology in historic times have possessed very great general problem-solving powers (4, pp. 431-432).

Thus, creative activity appears simply to be a special class of problem-solving activity characterized by novelty, unconventionality, persistence, and difficulty in problem formulation.

Simulation of Problem Solving

As we indicated earlier, the theory of problem solving we are putting forth derives from mechanisms that solve problems in the same manner as humans, and whose behavior can be observed, modified, and analyzed. The only available technique for constructing problem solvers is to write programs for digital computers; no other physical mechanisms are complex enough.

The material in the present paper rests mostly on several programs that we have constructed. These are:

1. The Logic Theorist. The Logic Theorist is a computer program that is capable of discovering proofs for theorems in

elementary symbolic logic, using heuristic techniques similar to those used by humans. Several versions of the Logic Theorist have been coded for a computer, and a substantial amount of experience has been accumulated with one of these versions and some of its variants (6, 7, 8, 9).

2. The Chess Player. We have written a program that plays chess. It is just now being checked out on the computer, but we have done a good deal of hand simulation with the program so that we know some of its more immediate characteristics (10).

When we say that these programs are simulations of human problem solving, we do not mean merely that they solve problems that had previously been solved only by humans — although they do that also. We mean that they solve these problems by using techniques and processes that resemble more or less closely the techniques and processes used by humans. The most recent version of the Logic Theorist was designed explicitly as a simulation of a (particular) human problem solver whose behavior had been recorded under laboratory conditions.

Although the RAND-Carnegie group is the only one to our knowledge that has been trying explicitly to construct programs that simulate human higher mental processes, a number of workers have been exploring the capabilities of computer programs to solve complex and difficult problems. Many of these programs provide additional information about the nature of the problem-solving process. Some of the more relevant are:

3. Musical Composition. A computer program has been written and run on the ILLIAC that composes music employing Palestrina's rules of counterpoint. Some of its music has been performed by a string quartet and tape-recorded, but as far as we are aware, no description of the program has been published. Other experiments in musical composition have also been made.

4. Chess Playing. Two programs besides ours have been written that play chess. Although both of these proceed in a manner that is fundamentally different from the ways humans play chess, some of their features provide illuminating comparisons (10).

5. Design of Electric Motors. At least two, and probably more, computer programs have been written, and are now being used by industrial concerns, that design electric motors. These programs take as their inputs the customers' design specifications, and produce as their outputs the manufacturing specifications that are sent to the factory floor. The programs do not simply make calculations needed in the design process, but actually carry out the analysis itself and make the decisions that were formerly the province of the design engineers.

The main objective of these motor design programs, of course, is to provide effective problem-solving routines that are economical substitutes for engineers. Hence, these programs simulate human processes only to the extent that such processes

are believed to enhance the problem-solving capabilities and efficiency of the programs.

6. Visual Pattern Recognition. A program has been written that attempts to learn a two-dimensional pattern — like an "A" — from examples. The program was developed by Selfridge and Dineen (2, 15). Although only partly successful, it was a pioneering attempt to use computer simulation as a technique for investigating an area of human mental functioning.

Is the Logic Theorist Creative?

The activities carried on by these problem-solving computer programs lie in areas not far from what is usually regarded as "creative." Discovering proofs for mathematical theorems, composing music, designing engineering structures, and playing chess would ordinarily be thought creative if the product were of high quality and original. Hence, the relevance of these programs to the theory of creativity is clear — even if the present programs fall short of exact simulation of human processes and produce a fairly mundane product.

Let us consider more specifically whether we should regard the Logic Theorist as creative. When the Logic Theorist is presented with a purported theorem in elementary symbolic logic, it attempts to find a proof. In the problems we have actually posed it, which were theorems drawn from Chapter 2 of Whitehead and Russell's Principia Mathematica (17), it has found the proof about three times out of four. The Logic

Theorist does not pose its own problems — it must be given these — although in the course of seeking a proof for a theorem it will derive the theorem from other expressions and then attempt to prove the latter. Hence, in proving one theorem, the Theorist is capable of conjecturing other theorems and then trying to prove these.

Now no one would deny that Whitehead and Russell were creative when they wrote Principia Mathematica (17). Their book is one of the most significant intellectual products of the twentieth century. If it was creative for Whitehead and Russell to write these volumes, it is possibly creative for the Logic Theorist to reinvent large portions of Chapter 2 — rediscovering in many cases the very same proofs that Whitehead and Russell discovered originally. Of course the Logic Theorist will not receive much acclaim for its discoveries, since these have been anticipated, but, subjectively although not culturally, its product is novel and original. In at least one case, moreover, the Logic Theorist has discovered a proof for a theorem in Chapter 2 that is far shorter and more elegant than the one published by Whitehead and Russell.[1/]

1/ Perhaps even this is not creative. The Journal of Symbolic Logic has thus far declined to publish an article, co-authored by the Logic Theorist, describing this proof. The principal objection offered by the editor is that the same theorem could today be proved (using certain meta-theorems that were available neither to Whitehead and Russell nor the Logic Theorist) in a much simpler way.

If we wish to object seriously to calling the Logic Theorist creative, we must rest our case on the way it gets the problems it tackles, and not on its activity in tackling them. Perhaps the program is a mathematical hack, since it relies on Whitehead and Russell to provide it with significant problems, and then merely finds the answers to these; perhaps the real creativity lies in the problem selection. This certainly is the point of the fourth characteristic we listed for creativity. But we have already indicated that the Theorist has some powers of problem selection. In working backwards from the goal of proving one theorem, it can conjecture new theorems -- or supposed theorems -- and set up the subgoal of proving these. Historically, albeit on a much broader scale, this is exactly the process whereby Whitehead and Russell generated the theorems that they then undertook to prove. For the task they originally set themselves was to take the basic postulates of arithmetic (as set forth by Peano and his students), and to derive these as theorems from the axioms of logic. The theorems of Chapter 2 of Principia were generated, as nearly as we can determine the history of the matter, in the same way that subproblems are generated by the Logic Theorist -- as subproblems whose solution would lead to the solution of the problem originally posed.

We do not wish to exaggerate the extent to which the Logic Theorist is capable of matching the higher flights of the human mind. We wish only to indicate that the boundary

between its problem-solving activities and activities that are important examples of human creativity is not simple or obvious.

An Abstract Model of Problem-Solving Behavior

We turn next to the general theory of problem solving, only returning later to issues that are specific to the "creative" end of the problem-solving spectrum.

Definition of "Problem"

The maze provides a suitable abstract model for most kinds of problem-solving activity. A maze is a set of paths (possibly partly overlapping) some subset of which are distinguished from the others by having rewards at their termini (see Fig. 1). These latter are the "correct" paths; to discover one of them is to solve the problem of running the maze.

We can abstract one stage further, and characterize problem-solving by the following rubric: Given a set, P, of elements, to find a member of a subset, S, of P having specified properties. Here are some examples:

a. Solving a crossword puzzle. Take as P all possible combinations of letters of the English alphabet that will fill the white squares of the puzzle. The subset S comprises those combinations in which all consecutive linear horizontal and vertical sequences are words that satisfy specified definitions.

b. Finding the combination of a safe. Take as P all possible settings of the dials of the safe; and as S those

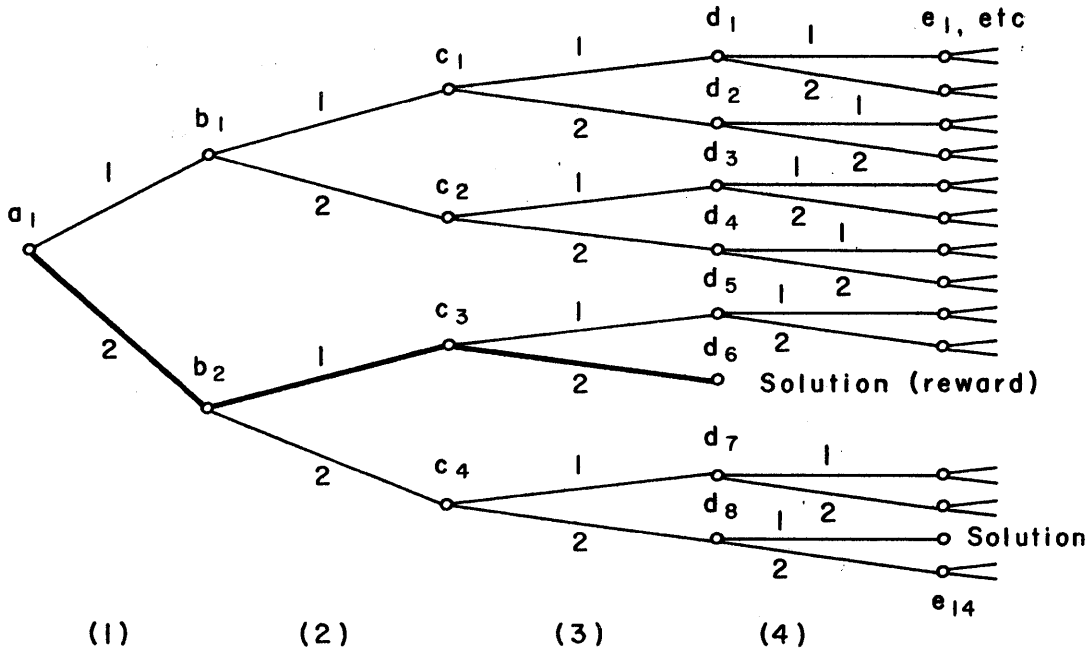


Fig. 1 — A problem maze, alternatives at choice points, $m = 2$; minimum length of path to solution, $k = 3$. The shortest path to a solution is given by the choices 2—1—2; it runs from choice point a_1, through b_2 and c_3 to d_6, the solution.

particular settings that open the safe. As safes are usually constructed, S consists of a single element.

c. Making a move in chess. Take as P the set of all possible (legal) moves; as S, the set of "good" moves, where the term "good" reflects some set of criteria.

d. Proving a theorem in logic or geometry. Take as P the set of all possible sequences of expressions in a formal language for logic (or geometry, respectively); and as S the subset of sequences that: (a) are valid proofs, and (b) terminate in the specified theorem.

e. Programming a computer to invert a matrix. Take as P the set of all possible sequences of computer instructions; and as S a particular sequence that will perform the specified matrix inversion.

f. Translating a German article into English. Take as P the set of all possible sequences of English words (of length, say, less than L); take as S the subset of sequences that: (a) satisfy certain criteria of English syntax and style, and (b) have the same meaning as the German original.

g. Designing a machine. Take as P the set of all possible parameter values for a machine design; take as S the subset of parameters values that: (a) satisfy the design specifications, (b) meet certain criteria of cost minimization.

In examples (d), (e), (f) the interpretation in terms of the maze model can be carried a step further by identifying the elements of the sequences mentioned there with the successive segments of the maze that constitute a path.

A Preliminary View of Problem-Solving Processes

There are any number of ways of classifying the processes that are used by humans to solve problems. One useful distinction differentiates processes for finding possible solutions (generating members of P that may belong to S), from processes for determining whether a solution proposal is in fact a solution (verifying that an element of P that has been generated does belong to S). This is a distinction that is often made in the literature, in one set of terms or another. Johnson (4), for example, distinguishes "production" processes from "judgment" processes in a way that corresponds closely to the distinction we have just made (4, pp. 50-52). We prefer to call the first class of processes solution-generating processes, and the second class verifying processes.

Solution generators range all the way from exceedingly "primitive" trial-and-error searches that take up the elements of P in a fairly arbitrary order to extensive calculations that select an appropriate solution at the first try or to elaborate analytic processes that construct a solution by some kind of "working backwards" from the known properties of solutions. In spite of the primitive character of trial-and-error processes, they bulk very large in highly creative problem solving; in fact, at the upper end of the range of problem difficulty there is likely to be a positive correlation between creativity and the use of trial-and-error generators.

How Large is the Maze?

In a sufficiently small maze where a member of S, once discovered, can be identified easily as a solution, the task of discovering solutions is trivial (e.g., a T-maze for a rat with food in one branch). The difficulties in complex problem solving arise from some combination of two factors: the size of the set of possible solutions that must be searched, and the task of identifying whether a proposed solution actually satisfies the conditions of the problem. In any particular case, either or both of these may be the sources of problem difficulty. By using our formal model of problem solving we can often obtain meaningful measures of the size and difficulty of particular problems, and measures of the effectiveness of particular problem-solving processes and devices. Let us consider some examples.

The Logic Theorist. We have made some estimates of the size of the space of possible solutions (proofs) for the problems handled by the Logic Theorist. By a possible proof—which we shall take as the element of the set P in this case—we mean a sequence of symbolic logic expressions. If we impose no limits on the length or other characteristics of such sequences, their number, obviously, is infinite. Hence, we must suppose at the outset that we are not concerned with the whole set of possible proofs, but with some subset comprising, say, the "simpler" elements of that set. We might restrict P, for example, to proofs consisting of sequences of not more than twenty logic expressions, with each expression not more than 23

symbols in length and involving only the variables p, q, r, s, and t, and the connectives "or" and "implies." The number of possible proofs meeting these restrictions is about 10^{235}— one followed by 235 zeros!

The task is also not trivial of verifying that a particular element of the set P, as we have just defined it, is a proof of a particular problem in logic; for it is necessary to determine whether each expression in the sequence is an axiom or follows from some of the expressions preceding it by the rules of deductive inference. In addition, of course, the expression to be proved has to be contained in the sequence.

Clearly, selecting possible proofs by sheer trial and error and testing whether each element selected is actually the desired proof is not a feasible method for proving logic theorems—for either humans or machines. The set to be searched is too large and the testing of the elements selected is too difficult. How can we bring this task down to manageable proportions?

First of all, the number we have just computed—10^{235}— is not only exceedingly large but also arbitrary, for it depends entirely on the restrictions of simplicity we impose on P. By strengthening these conditions, we reduce the size of P, by weakening them we increase its size. We must look for a more meaningful way to describe the size of the set P. This we do by considering a simple solution generator that produces members of the set in a certain order, and asking how many members the generator would have to produce, on the average, to obtain

solutions to problems of a specified class. Let us generate elements of P according to the following simple scheme (which we call the British Museum Algorithm in honor of the primates who are credited with employing it) (7):

(1) We consider only sequences of logic expressions that are valid proofs — that is, whose initial expressions are axioms, and each of whose expressions is derived from prior ones by valid rules of inference. By generating only sequences that are proofs (of something), we eliminate the major part of the task of verification.

(2) We generate first those proofs that consist of a single expression (the axioms themselves), then proofs two expressions long, and so on, limiting the alphabet of symbols as before. Given all the proofs of length k, we generate those of length (k+1) by applying the rules of inference in all permissible ways to the former to generate new derived expressions that can be added to the sequences. That is, we generate a maze (see again, Fig. 1) each choice point (a_1, b_1, b_2, etc.) representing a proof, and the alleys leading from the choice point representing the legitimate ways of deriving new expressions as immediate consequences of the expressions contained in the proof. Thus, in the figure, d_4 is a proof that can be derived as an immediate consequence of c_2, using path 2.

We estimate that of the sixty-odd theorems that appear in Chapter 2 of Principia Mathematica (17) about six (all of which are among the first ten in the chapter) would be included in

the first 1,000 proofs generated by the algorithm, but that about a hundred million more proofs would have to be generated to obtain all the theorems in the chapter. (The actual number may be much greater; it is difficult to estimate with any accuracy.) That is to say, if we used this scheme to find the proof of a theorem selected at random from the theorems of Chapter 2, we would, on the average, have to generate some fifty million possible solutions before finding the one we wanted; and the chances of finding the proof among the first thousand generated would be only one in ten. One hundred million (10^8) is a large number, but a very small number compared with 10^{235}. Thus a proof has a very much higher probability of turning up in Chapter 2 of Principia if it is relatively simple than if it is complicated. On the other hand, something more effective is needed than the British Museum Algorithm in order for a man or a machine to solve problems in symbolic logic in a reasonable time.

Before leaving the Logic Theorist, we wish to mention a variant of the Whitehead and Russell (17) problems which we have also studied, and which will be the subject of some analysis later. At Yale, O. K. Moore and Scarvia Anderson (5) have studied the problem-solving behavior of subjects who were given a small set (from one to four) of logic expressions as premises and asked to derive another expression from these, using twelve specified rules of transformation. (For details see the discussion in the next section and the Appendix.) If

we again suppose derivations to be generated by working forward from the premises, we can, in the case where there is a single premise, make simple estimates of the number of possible derivations of given length—and hence characterize this particular problem maze.

Assuming (which are oversimplifications) that each rule of transformation operates on one premise, and that each such rule is applicable to any premise, this particular maze branches in twelve directions (one for each rule of transformation) at each choice point. That is, we start out with a single premise; depending on which rule of transformation we apply, we obtain one of twelve possible new expressions from each of these, and so on. Thus, the number of possible sequences of length k is 12^k. If the problem expression can be derived from the premise in a minimum of seven steps, then a trial-and-error search for the derivation would require, on the average, the construction of $1/2 \times 12^7 = 1.8 \times 10^7 = 18{,}000{,}000$ sequences.

If only four rules of transformation were actually applicable, on the average, at each stage (a more realistic assumption, since expressions must be of particular forms for particular rules to be applicable to them), the number of sequences of length 7 would still be $4^7 = 16{,}384$.

Chess Playing. Let us turn now to a second example—choosing a move in chess. On the average, a chessplayer whose turn it is to move has his choice among twenty to thirty alternatives. There is no difficulty, therefore, in "finding"

possible moves, but great difficulty in determining whether a particular legal move is a good move. The problem lies in the verifier and not in the generator. However, a principal technique for evaluating a move is to consider some of the opponent's possible replies to it, one's own replies to his, and so on, only attempting to evaluate the resulting positions after this maze of possible move sequences has been explored to some depth. The maze of move sequences is tremendously large. If we consider the number of continuations five moves deep for each player, assuming an average of 25 legal continuations at each stage, we find that P, the set of such move sequences has about 10^{14} (one hundred million million) members.

Opening a Safe. We can make similar estimates of the sizes of the set P for the other examples of problem-solving tasks we have listed. In all cases the set is so large as to foreclose a solution-generating process that makes a completely random search through the set for possible solutions.

Before we leave our estimates, it will be useful to consider one additional "synthetic" example that has a simpler structure than any we have discussed so far, and that will be helpful later in understanding how various heuristic devices cut down the amount of search required to find problem solutions. Consider a safe whose lock has ten independent dials, each with numbers running from 00 to 99 on its face. The safe will have $100^{10} = 10^{20}$, or one hundred billion billion possible settings, only one of which will unlock it. A would-be safe-cracker, twirling

the dials at random, would take on the average 50 billion billion trials to open it.

However, if the safe were defective, so that there was a faint click each time any dial was turned to its correct setting, it would take an average of only 50 trials to find the correct setting of any one dial, or 500 trials to open the safe. The ten successive clicks that told him when he was getting "warmer" would make all the difference to the person opening the safe between an impossible task and a trivial one.

Thus, if we can obtain information that tells us which solutions to try, and in particular, if we can obtain information that allows us to factor one large problem into several small ones—and to know when we have successfully solved each of the small ones—the search task can be tremendously reduced. This guidance of the solution generator by information about the problem and its solution, and this factorization of problems into more or less independent subproblems lie at the heart of effective problem-solving processes.

Heuristics for Problem Solving

We have seen that we can describe most problems abstractly in terms of a maze whose paths are possible solutions, and some small fraction of which are actual solutions. Then we can analyse the problem-solving processes into those that determine the order in which the paths shall be explored (solution generators) and those that determine whether a proposed solution is in fact a solution (solution verifiers).

Our examples show that solution generation and verification need not operate in an inflexible sequence. In the Logic Theorist, as we saw, certain of the verifying conditions are built into the generator, so that only valid proofs are generated and other sequences of logic expressions are never considered. On the other hand, in chess playing, to verify that a proposed move is satisfactory, it is necessary to consider a large maze of possible continuations, and to search some part of this maze.

In the present section we shall examine some actual examples of successful problem-solving programs to see just what is involved in solution generation and verification, and how the programs reduce the problems to manageable size. We use the term heuristic to denote any principle or device that contributes to the reduction in the average search to solution. Although no general theory of heuristics exists yet [1], we can say a good deal about some of the heuristics employed in human complex problem solving. Our data derive largely from symbolic logic and chess, problems that are formal and symbolic. This characteristic of the tasks undoubtedly limits the range of heuristics that we have observed. However, the kinds of heuristics we have found and can describe (e.g., planning and functional analysis) seem to have rather general applicability.

Efficient Generators

Even when the set P is large, as it usually is in complex

[1] See, however, the work of G. Polya (13,14) who has analysed the use of heuristics in mathematics.

problem solving, it is possible for the solution generator to consider at an early stage those parts of P that are likely to contain a solution, and to avoid the parts that are most likely to be barren. For example, many problems have the following form: S, the set of solutions, consists of all elements of P with property A, and property B, and property C. No generator is available that will generate elements having all three properties. However, generators may exist that generate elements having any two of the properties. Thus there are three possible schemes: (1) to generate elements with properties A and B until one is found that also has C; (2) to generate elements with A and C until one is found with property B; (3) to generate elements with B and C until one is found with property A. Which generator should be chosen depends on which constraints are the more difficult to satisfy, and on the relative costs of generation. If there are lots of elements satisfying A, then generating elements with B and C is reasonable, since an "A" can be expected to show up soon. If "A's" are rare, it is better to generate elements that already have property A.

The Logic Theorist provides a clear example of this type of heuristic. Recall that the problem of the Logic Theorist is to find proofs. A proof is a list of logic expressions satisfying the following properties:

A. The beginning of the list consists of known theorems (any number of them);

B. All other expressions on the list are direct and valid

consequences of prior expressions on the list;

C. The last expression on the list is the expression to be proved.

Now, although there is no generator that will turn out sequences satisfying all three of these conditions, there are generators that satisfy any two of them. It is easy to write down lists that start with theorems and end with the known expression. The difficult condition, however, is B: that the list must consist of valid inference steps. Hence, it would be obviously foolish to choose a generator that automatically satisfied A and C, and simply wait until it generated a list that also satisfied B.

It is also possible to construct a generator satisfying A and B—one that produces lists that are proofs of something. This generator could find a proof by producing such lists until one appeared containing the desired expression—condition C. The British Museum Algorithm discussed earlier is a generator of this kind. Finally, one can build a generator that satisfies conditions B and C. Fixing the last expression to be the desired one, lists are produced that consist only of valid inference steps leading to the last expression. Then the problem is solved when a list is generated that satisfies condition A, so that the expressions on the front of the list are all theorems. With this kind of generator, the list is constructed "backwards" from the desired result toward the given theorems. This is the way the Logic Theorist actually goes about discovering proofs.

How do we choose between these two generators — the one that requires a search to satisfy C, or the one that requires a search to satisfy A? In the case of logic the answer is reasonably clear. There is only one terminal expression (the theorem to be proved), but there are usually many known theorems. The difference is roughly comparable to the problem of our finding a needle in a haystack versus the problem of the needle finding its way out of the haystack. Working backwards is by far the more efficient.

It should be clear that there is nothing inherently superior in working backwards as opposed to working forwards. The choice between them resolves itself into a question of which constraints are the most binding. It may well be, of course, that the particular situation found here (many possible starting points versus a single end) which predisposes towards working backwards, is relatively common.[1/]

Simple Selection Heuristics

When a problem solver faces a set of alternatives, such as the branches from a choice point in the maze in Figure 1, a common heuristic procedure is to screen out possible paths initially, using a relatively inexpensive test. To see the worth of this procedure, consider a maze having m alternatives

1/ Duncker (3) calls working backwards an "organic" procedure, and distinguishes it from the "mechanical" procedures of working forwards. Our analysis shows both why the former might be generally more efficient than the latter, and why there is nothing qualitatively different between them.

at each branch point and length k. If there were a single correct path to the goal, finding that path by random search would require, on the average, $1/2 \cdot m^k$ trials. If a heuristic test were available that could immediately weed out half of the alternatives at each branch point as unprofitable, then a random search with this heuristic would require only $1/2(1/2m)^k$ trials on the average. This is a reduction in search by a ratio of 2^k, which, if the maze were only seven steps in length would amount to a factor of 128, and if the maze were ten steps in length, a factor of just over a thousand.

The Logic Theorist uses a number of such selection heuristics. For example, in working backwards as described above, it can proceed in several directions from the list of logic expressions it has already obtained. Different theorems can be used with the various expressions already generated to make new valid inferences. Thus, the Logic Theorist generates a maze of subproblems, which corresponds exactly to the abstract pictures we have been giving (7).

In Figure 2, two mazes are shown, derived from two attempts, under slightly different conditions, to prove a particular theorem from the same set of known theorems. In each maze, the desired theorem (*2.45) is represented by the top node; and each node below corresponds to a new expression generated (as a subproblem) from the node immediately above it. In both cases the Logic Theorist found the same proof, which is designated in each maze by the heavy line. When it was generating the lower maze, the Logic Theorist had available two selective heuristics it

did not have during the run that generated the upper maze. One of these heuristics weeded out new expressions that appeared "unprovable" on the basis of certain plausible criteria; the other heuristic weeded out expressions that seemed too complicated, in the sense of having too many negation signs. These two heuristics reduced the amount of search required to find the solution by a factor of 24/9 or 2.7. When the cost of the additional testing is taken into account, the net saving in total problem-solving effort after allowing for this testing was 2.3. On the other hand we have experimented with heuristics that were excellent in their performance characteristics (reducing search by factors of 10) but that required so much effort to carry out as to cancel out the gain.

A heuristic need not be foolproof – indeed most are not. Both heuristics mentioned above eliminate paths that lead to solutions. In rare cases they can even eliminate all paths to solutions. To take another example, a chess heuristic that would instantly remove from consideration any move that left the queen under attack would be an excellent rule of thumb for a novice player, but would occasionally lead him to miss a winning queen sacrifice. Occasionally, heuristics are found that are foolproof. These are usually called algorithms. The British Museum Algorithm is an example, for it will always generate a proof, given enough time. ("Enough time" may, as we have seen, sometimes be centuries or millenia.)

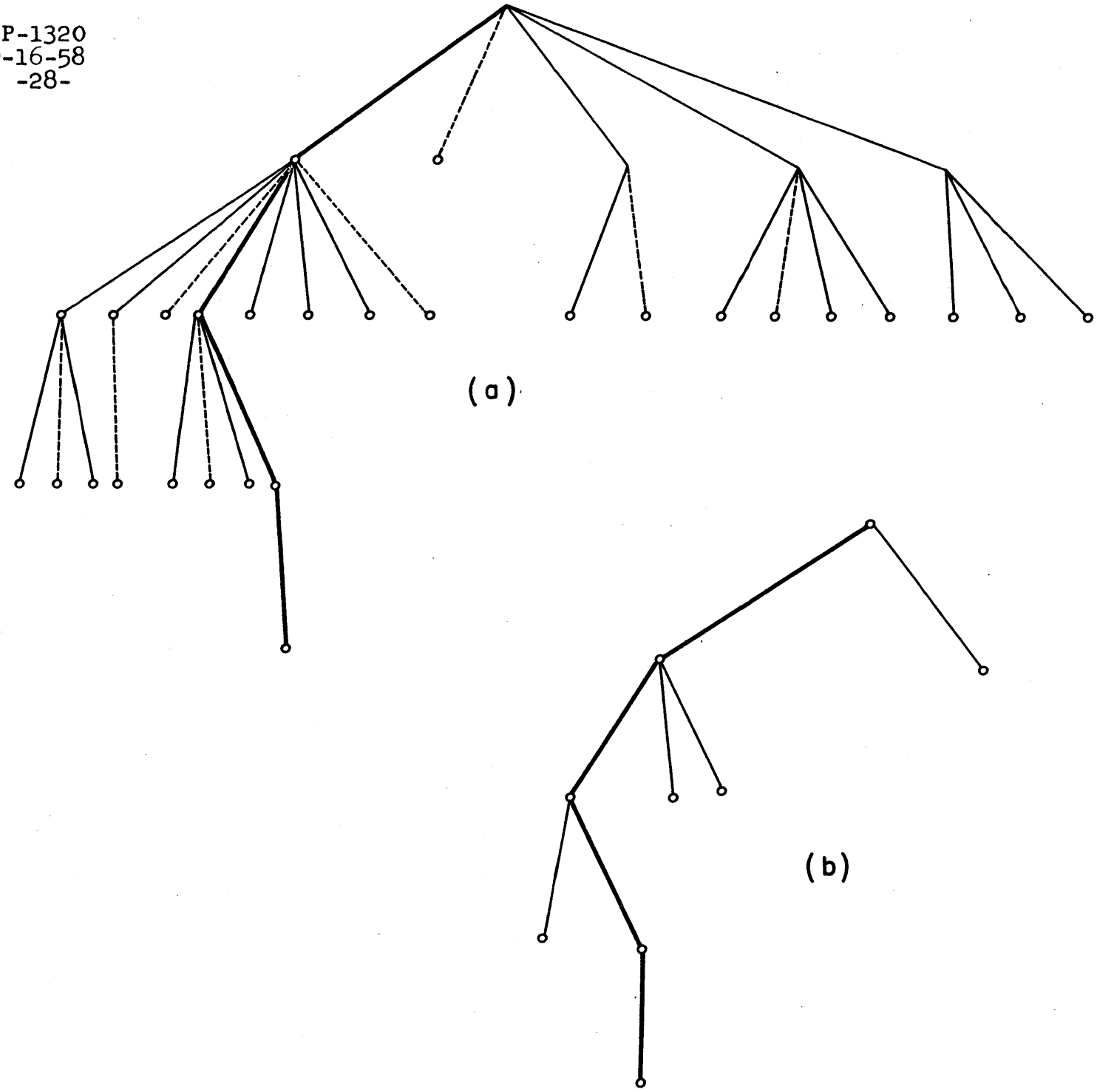


Fig. 2 — Mazes of two proofs of Theorem 2.45 from same initial theorems. Identical programs generated the two mazes, except that the program of maze b had two additional selective heuristics. The heavy line is the proof. Dotted lines show additional branches eliminated by selective heuristics already in the first program.

Strategies in Solution Generation

Usually the information needed to select promising paths becomes available only as the search proceeds. Examination of paths produces clues of the "warmer-cooler" variety that guide the further conduct of the search. We have already given a simple, but striking, example of this in the clicking safe. As any particular dial is turned to the correct setting, the person opening the safe is informed by the click that he should stop manipulating that dial and go to the next. As a result he need attempt, on the average, only 500 of the enormous total number of possible settings of the dials.

The sequential availability of cues derives from a deep property of problem-solving tasks that we must examine closely. There are, in general, two distinct ways to describe any particular choice point in a problem-solving maze. In chess, for example, a particular position can be specified by stating (verbally or with a diagram) what piece occupies each square on the board. Alternatively, the position can be specified by giving a sequence of moves that leads to it from the opening position.

Similarly, in logic an expression can be specified by writing it out explicitly in the usual way, or equally well by giving a sequence of operations on the axioms (a proof) that will produce it. (In Figure 1, the solution may be described as d_6 or as 2-1-2.) Or, in arithmetic, if we use the symbol x' to mean the integer that follows x, we can write the

number five as 5 or as 0'''''. Or, as a more familiar example, we can designate a house by an address—5936 Phillips Avenue—or by the sequence of turns (go two blocks, turn right, go nine blocks, turn right, go one block) necessary to get there from a given starting point.

In all these cases, we will call the first method of specifying an element of P specification by state description; the second method, specification by process description. Often problems are set by providing the problem solver with a partial or complete state description of the solution, the state descriptions of one or more starting points, and a list of allowable processes. The task, in these terms, is to find a sequence of processes that, operating on the initial state, will produce the final state.

We can now see how cues become available sequentially, and why, consequently, strategies of search that use the cues are possible. Each time a process is applied to an initial state, a new state with a new description is produced. If there are relations (known to or learnable by the problem solver) between characteristics of the state description and distance from the goal—from the final description that represents the solution, these relations can be used to tell when the problem solver is getting "warmer" or "colder," hence, whether or not he should continue along a path defined by some sequence of processes. If, for example, in Figure 1, the state description corresponding to b_2 indicates that it is closer to the solution than the

description corresponding to b_1, then the problem solver at a_1 will take path 2 instead of path 1, and will be relieved of the necessity of exploring the entire upper half of the maze.

Let us examine a concrete example, for which we have data. Consider the following sequence of logic expressions that was written down by a subject solving one of O. K. Moore's problems in our laboratory. (The reader does not need to know what the symbols mean to understand the example. The task involved in these problems is described briefly in the Appendix.)

Step	Expression	Justification for Step
(1)	$R\cdot(-P\supset Q)$	Given
(2)	$R\cdot(\ PvQ)$	Rule 6
(3)	$R\cdot(\ QvP)$	Rule 1 (inside parenthesis)
(4)	$(QvP)\cdot R$	Rule 1 (outside parenthesis)

The first line is the expression given to the subject as the starting point of the "maze." The last line is the expression he was instructed to produce by applying the allowable operations. The number at the right of each line is the number of the rule he applied to obtain that line from the previous one. In this example, the state description of the solution is the expression: "$(QvP)\cdot R$." The process description is: "The expression obtained by applying rules 6, 1, 1 to expression '$R\cdot(-P\supset Q)$'." It is, of course, not at all obvious, except by hindsight, that these two descriptions refer to the same logic expression—if it were obvious, the problem would be no problem.

How can the problem solver in this instance obtain new information as he proceeds each step down the maze? If we compare

the final expression with the intermediate ones, we note that at each stage the newly derived expression resembles the final expression more closely than did the previous ones. For example, expression (1) contains two symbols that do not appear in the final expression; these have disappeared from expression (2). The next step rectifies the order of the symbols within parentheses, while the final step rectifies the order of the symbols in the expression as a whole.

A simple heuristic to follow in such cases is to apply an operator if the result of its application is to produce a new expression that resembles the final expression more nearly than did the previous one. To apply the heuristic, the problem solver needs some criteria of similarity, but it is easy to see what they might be in the present case. These criteria provide the "clicks" that reduce the amount of search required.

We can test this explanation further by comparing it with the thinking-aloud protocol that the subject produced as he performed the task. We produce here an excerpt from his protocol, slightly edited to make it more comprehensible to the reader:

E. What are you looking at?

S. I'm looking at the idea of reversing the R's. Then I'd have a similar group at the beginning. But I can easily leave something like that until the end . . . .

Now I'm trying to see what operation I might apply to expression (1) . . . . [He goes down the list of operations.] Rule 4 looks interesting . . . . but there's no switching of order. I need that P and Q changed . . . That doesn't seem practical with any of these rules . . . I'm looking for a way, now, to get rid of that horseshoe [⊃]. A . . . here it is,

Rule 6. So I'd apply Rule 6 to the second part of what we have up there.

E. That gives you [writing] (2) R·(PvQ).

S. Okay. And now I'd use Rule 1 on P and Q, and then with the entire expression.

E. We'll do them one at a time [writing] (3) R·(QvP). Now the total expression?

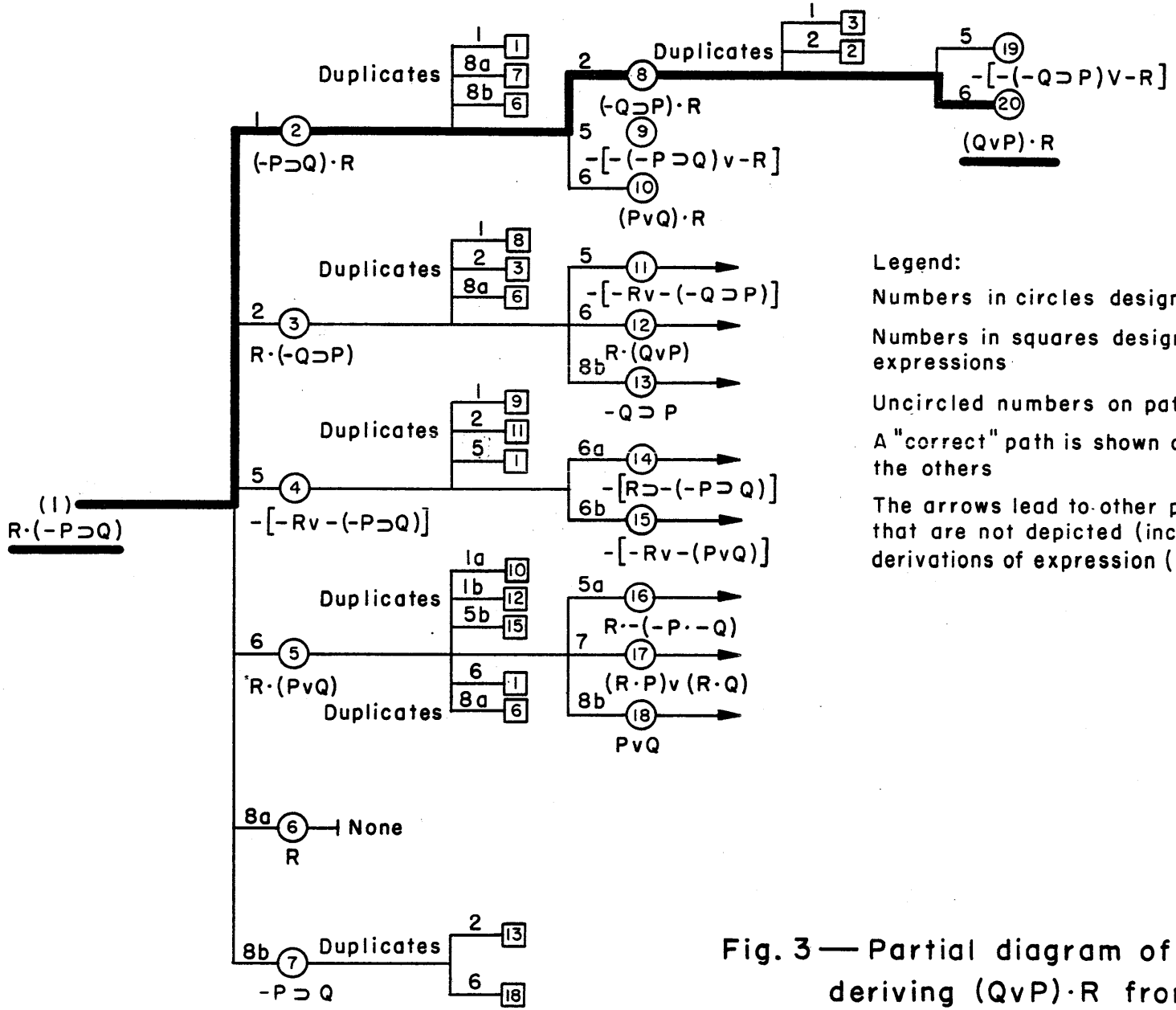
S. Yeah.

E. You get [writing] (4) (QvP)·R, and .... that's it.

S. That's it all right. Okay . . . that wasn't too hard.

It will be observed that the subject thought through the successive changes (bringing the R to the left side, interchanging P and Q, eliminating the horseshoe) in the order opposite to that in which he actually carried them out, but that his process—that of making the expression at hand more and more like the final expression—is precisely the one we have described.

To give a picture of the selectivity involved in this particular piece of problem solving, we show in Figure 3 a somewhat simplified picture of a portion of the problem maze, including only those branches that would actually be explored if the problem were solved by a systematic search without selectivity. Note that the path to the goal (1–2–6) discovered through systematic search is different from the one (6–1–1) discovered by the subject's selective processes, and that the systematic search generated many expressions that—from the protocol evidence—did not even enter the subject's awareness.



Legend:

Numbers in circles designate derived expressions

Numbers in squares designate duplicate derived expressions

Uncircled numbers on paths designate operations

A "correct" path is shown as a heavier line than the others

The arrows lead to other portions of the maze that are not depicted (including alternative derivations of expression (20))

Fig. 3 — Partial diagram of problem maze for deriving (QvP)·R from R·(-P⊃Q)

The same kind of alternation between state description and process description is involved in choosing a move in chess. Because of the tremendous size of the maze of continuations, only a few of all the possible lines of play can be examined. When the player considers a particular move, he can construct in his imagination a picture of the board after the move has been made. He can then examine this new state description to see what features of it are favorable, what features unfavorable, and what likely continuations it suggests. In this way he is guided to examine a few paths through the maze (if he is a good player, his heuristic will usually lead him to examine the important ones), and he can explore these to some depth—sufficiently deeply to be able to evaluate directly the final positions he reaches. The best evidence we possess indicates that the strongest chess players do not examine more than (at the very most) a few dozen continuations, and these to depths ranging from a couple of moves to ten or even more (see Fig. 4). The ability of the chess master, so amazing to the novice, to explore in depth derives from his ability to explore very selectively without missing important alternatives. The "clicks" he notices, inaudible to the novice, are loud and obvious to him.

Functional Analysis

Underlying the heuristic of "reducing differences" is the general concept of functional analysis. Functional or means-end

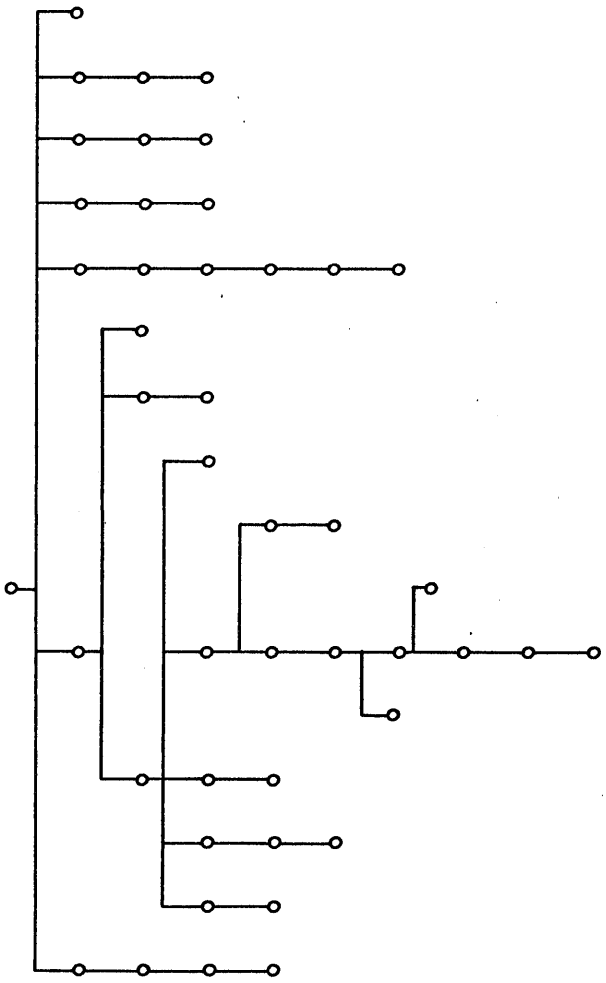


Fig. 4 — Portion of maze of continuations examined by chessmaster in middle game position. (from de Groot [1], p. 207). The subject was a master.

analysis provides a generalized heuristic that can be applied to a wide range of problems. We will describe a program for functional analysis that we have incorporated in a revised version of the Logic Theorist, and which, while not completely general, can almost certainly transfer without modification to problem solving in trigonometry, algebra, and probably other subjects like geometry and chess.

The entities that the program recognizes are expressions, differences between expressions, operators, goals and subgoals, and methods. The program can be used as a problem-solving heuristic for problems of the form: "given expression a and a set of admissible operators, to derive expression b." We have already observed that logic problems can be put in this form—and so can most other problems formulated in terms of the maze model.

Associated with each goal in the heuristic is a set of methods—procedures that may help attain the goal in question. A method may, in turn, involve establishing subgoals and applying the methods associated with these. At some point, if the heuristic is successful, a subgoal is attained by one of its methods; this success reactivates the goal at the next higher level in the hierarchy, and so on.

Let us be more concrete. These are three types of goals in the functional problem solver:

Type 0 Goal: Find a way to transform expression a into expression b.

Type 1 Goal: Reduce the difference d between expressions a and b.

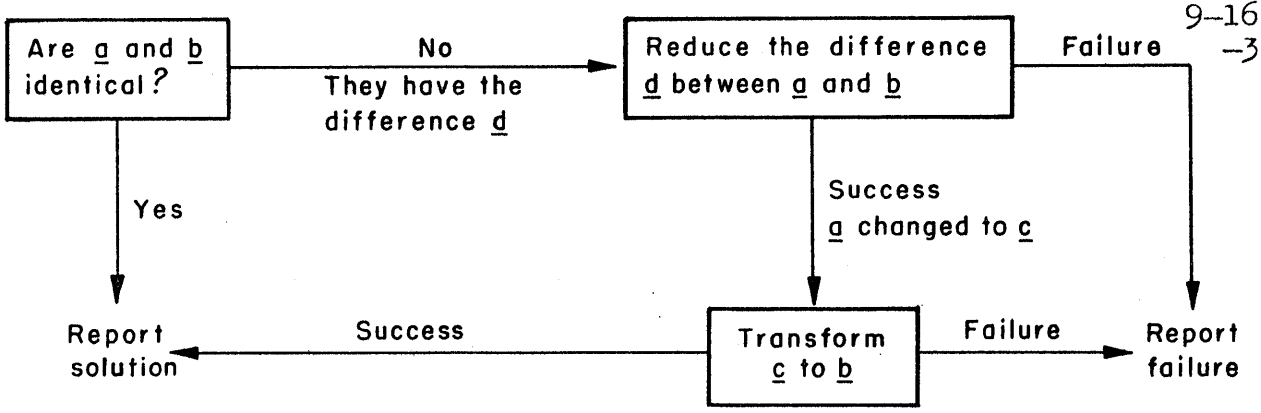
Type 2 Goal: Apply operator q to expression a.

For functional analysis, one method is associated with each of these goals. Briefly, the method associated with the type 0 goal consists in (Fig. 5A): (a) matching the two expressions to find a difference, d, between them; (b) setting up the type 1 subgoal of reducing d—if successful, a new transformed expression, c , is obtained; (c) setting up the type 0 subgoal of transforming c into b. If the last step is successfully carried out, the original problem has been solved.

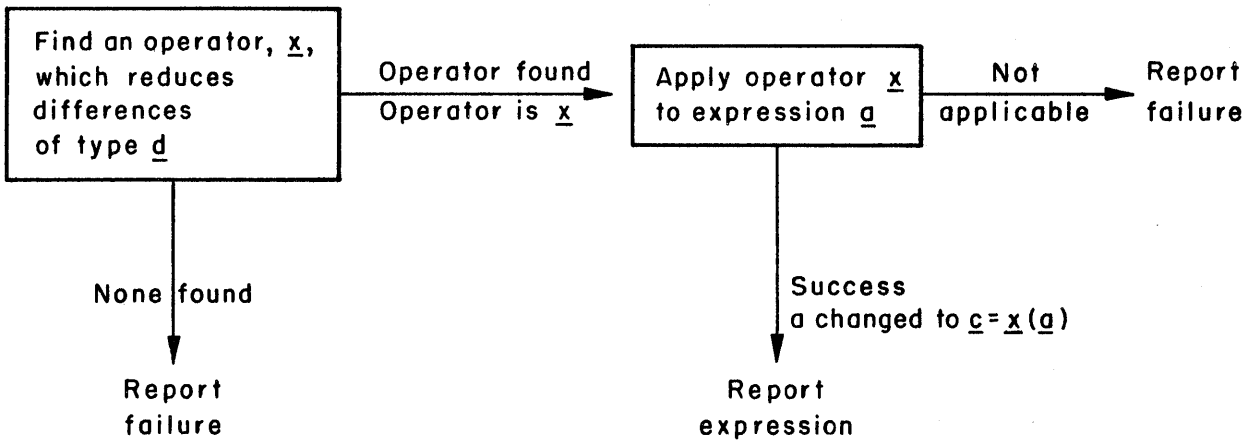
The method associated with the type 1 goal consists in (Fig. 5B): (a) searching for an operator that is relevant to the reduction of the difference, d; and (b) setting up the type 2 goal of applying the operator.

The method associated with the type 2 goal consists in (Fig. 5C): (a) determining if the conditions are met for applying q to a; (b) if so, applying the operator, if not, setting up the type 1 subgoal of reducing the difference between a and the conditions for applying q.

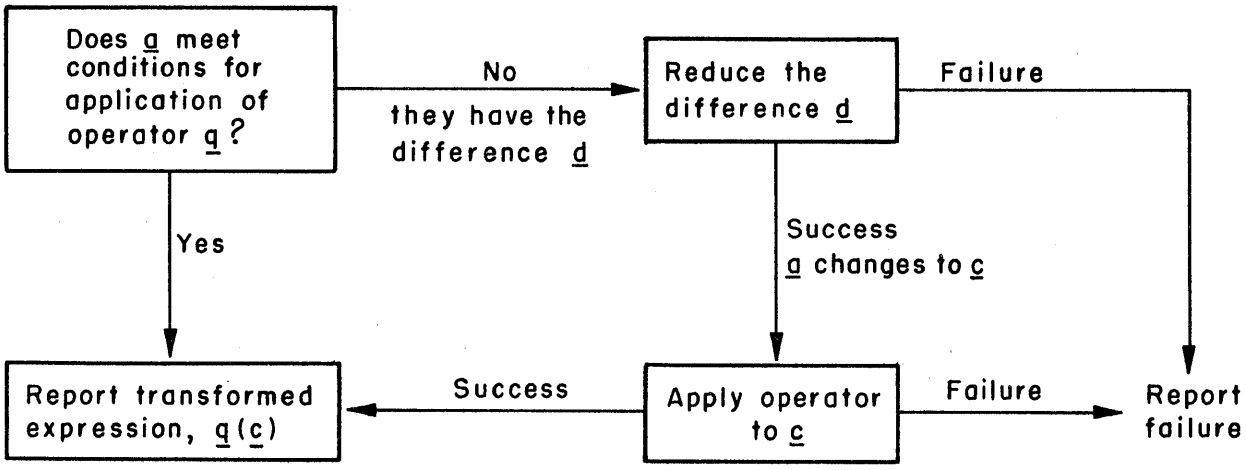
Let us see how this functional problem solver would approach the particular logic problem we examined in the last section. The problem solver is given type 0 goal of transforming expression (1) into expression (4). In trying to reach this goal (Fig. 5A), it first matches (1) with (4) to see what differences there are, and notes that the symbols P and Q occur in reverse order in (1) from the order in (4). This generates the type 1



5A — Type 0 goal: Transform a to b



5B — Type 1 goal: Reduce the difference d between a and b



5C — Type 2 goal: Apply operator q to expression a

Fig. 5 — Goals in Functional Heuristic

subgoal (Fig. 5B) of eliminating this difference. The problem solver remembers (from instruction or previous description) that operator 1 is relevant to reducing differences of this kind, and sets up the type 2 subgoal (Fig. 5C) of applying rule 1 to (1). The conditions for applicability are not met, for operator 1 requires a v between P and Q, while actually there is a $\supset$. Hence a new type 1 goal is created to change the $\supset$ into a v. Operator 6, which has this function is found, and the type 2 goal of applying it is set up and achieved. The transformed expression (2), now satisfies the conditions for applying operator 1. This is now applied, achieving the type 2 subgoal, and yielding expression (3). A similar, but simpler sequence of events leads to expression (4) — at which time the problem solver notes that it has eliminated all differences between the initial and terminal expressions, and hence has solved the original problem. Reference back to the protocol will show how closely this program models the behavior of the subjects.[1/]

1/ Our aim here is to illustrate, and not to deal with the scientific problems of how well these programs explain the protocols. In the sketch of the program given in the text, we generate the expressions in the order in which the subject wrote them down. The protocol indicates clearly that he proceeded initially in the opposite order. To simplify the exposition, we have not tried to describe the program that would simulate the subject's behavior most closely, and we have taken the liberty of editing the protocols. At another time we will undertake a more systematic analysis of the protocols as evidence.

It will be observed that neither the goals nor the methods of the functional problem-solving program make reference to logic or any other subject matter. Simply by acquiring new definitions of the terms "expressions," "differences," and "operators," the problem solver can use the functional heuristic to solve problems relating to quite different subject matter. We hope, in the near future, to test whether this heuristic will, for example, solve trigonometric identities.

The Heuristics of Planning

Another class of heuristics of great generality that increase the selectivity of solution generators are those that come under the rubric of "planning." Consider again a maze k steps in length with m branches at each choice point. Suppose that, instead of cues at each choice point, there were a cue at every second step to mark the correct path (see Fig. 6). Then the task of traversing the maze successfully could be divided into a number of subtasks: specifically, the tasks of reaching successively each of the choice points that were marked by the cues.

Such a set of subtasks would constitute a plan. In place of the original task of traversing a maze k steps in length, the problem solver would now have the task of traversing (k/2) mazes each 2 steps in length. The expected number of paths that would have to be searched to solve the first problem is, as before, $1/2 \cdot m^k$. The expected number of trials to solve the

second problem is $1/2 \cdot k/2 \cdot m^2$.

If, as in the figure, the original maze were 6 steps in length with two alternatives at each choice point, the average amount of search required would be reduced from 32 trials to 6—to which would have to be added the effort required to find the plan.

We use such a planning technique whenever we take a cross-country trip. First we sketch a general route from major city to major city; then, taking these cities as subgoals, we solve the subproblem of reaching each from the previous one.

We have devised a program of this kind to describe the way some of our subjects handle O. K. Moore's logic problems, and perhaps the easiest way to show what is involved in planning is to describe that program. On a purely pragmatic basis, the twelve operators that are admitted in this system of logic can be put in two classes, which we shall call "essential" and "inessential" operators, respectively. (See the Appendix.) Essential operators are those which, when applied to an expression, make "large" changes in its appearance—change "PvP" to "P", for example. Inessential operators are those which make "small" changes—e.g., change "PvQ" to "QvP." As we have said, the distinction is purely pragmatic. Of the twelve operators in this calculus, we have classified eight as essential and four as inessential. Roughly speaking, the inessential operators are those that change the order of the symbols in expressions, or change the connectives ("v" to "·", for example) but make no other changes.

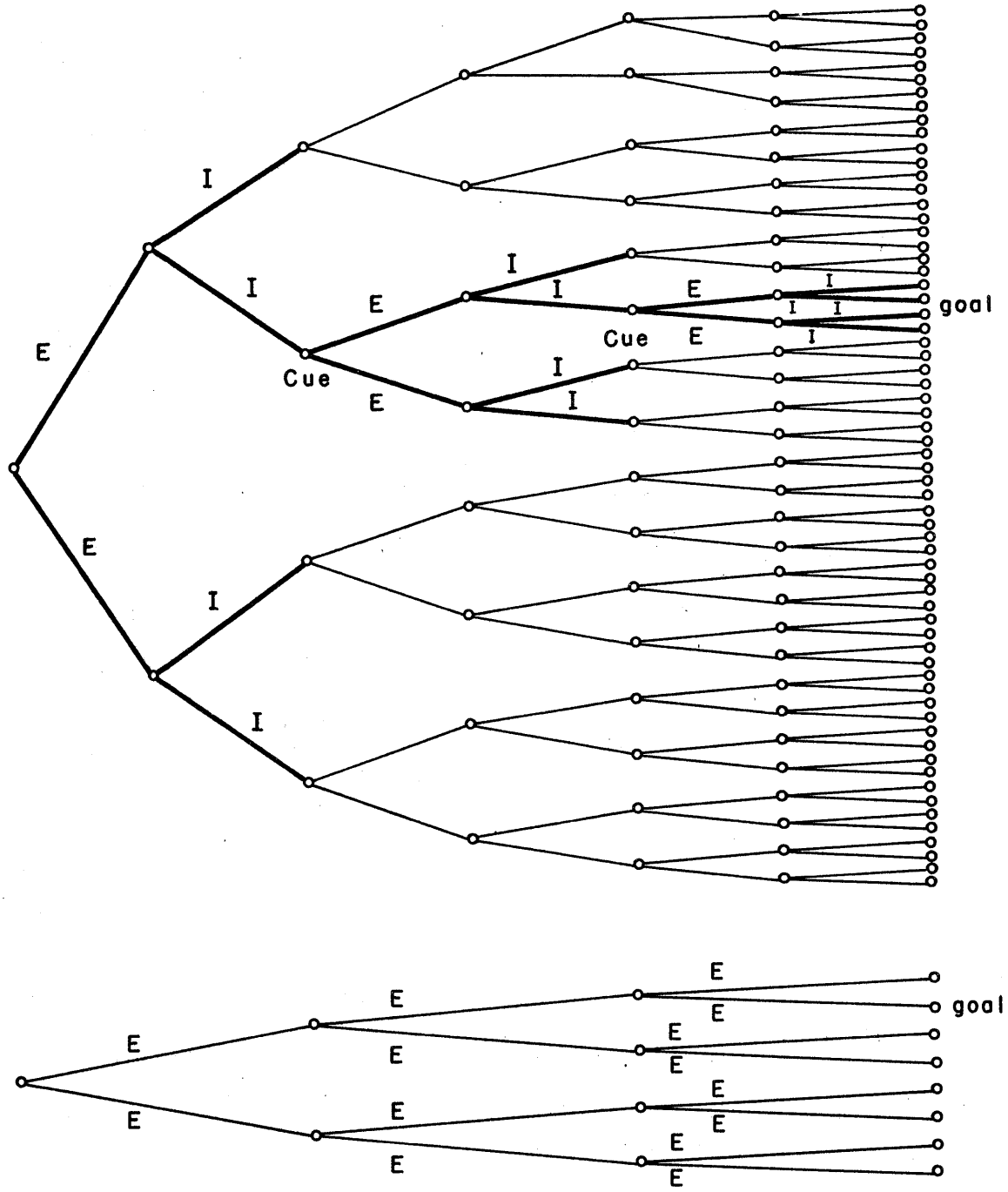


Fig. 6—Problem space (above), and planning space (below) for a simple task. The plan is used to find the cues in the larger maze; then only the darkened paths in the maze need be explored. The steps marked E are essential, those marked I, inessential—see text.

Next, we can take an expression and abstract from it those features that relate only to essential changes. For example, we can abstract from "PvQ" the expression (PQ), where the order of the symbols in the latter expression is regarded as irrelevant (i.e., (PQ) is treated as identical with (QP)). Clearly, if inessential operations are applied to the abstracted expressions, the expressions will remain unchanged, while essential operations can be expected to change them (e.g., the operator that will change "PvP" to "P" will change (PP) to (P)).

We can now set up a correspondence between our original expressions and operators, on the one hand, and the abstracted expressions and essential operators, on the other. Corresponding to the original problem of transforming a into b, we can construct a new problem of transforming a' into b', where a' and b' are the expressions obtained by abstracting a and b respectively. Suppose that we solve the new problem, obtaining a sequence of expressions, a'c'd' ...b'. We can now transform back to the original problem space and set up the new problems of transforming a into c, c into d, and so on. Thus, the solution of the problem in the planning space provides a plan for the solution of the original problem.

Let us examine (Fig. 7) an actual example of the application of the planning heuristic to the O. K. Moore problems. This example follows the protocol of one of our subjects and shows quite clearly that he used the planning heuristic in

precisely this way to solve the problem in question. The left-hand side of Figure 7 shows the sequence of expressions the subject wrote down in solving Moore's problem A4. The subject was given the expressions in lines (1) through (4) and told to derive the expression in line (11). He carried through the derivation in seven steps, the prior expressions and rules used in each step being given to the right of the derived expression.

The subject's protocol shows, however, that prior to obtaining the rigorous derivation, he had worked out a complete plan for the proof. The plan is shown, in terms of abstracted expressions, in the right-hand half of the figure. The planning activity took place immediately after the problem was presented to the subject, and before he instructed the experimenter to write down any transformations of the expressions given him as premises. Here is the protocol segment that discloses the planning activity:

> S. Well, one possibility right off the bat is when you have just a PvT like that [the problem expression] the last thing you might use is that rule 9. I can get everything down to a P and just add a vT. So that's one thing to keep in mind. . . . . I don't know if that's possible; but I think it is because I see that expressions (2) and (4) are somewhat similar. If I can cancel out the R's, that would leave me with just an S and Q; and if I have just an S and Q, I can eventually get—expression (3)—get the S's to cancel out and end up with just a Q. And if I end up with just a Q, maybe the Q's will cancel out; so you see all the way down the line. I don't know, it looks too good to be true, but I think I see it already.

At this point the subject has already constructed a four-step plan, which will lead him, as he executes it, to two

	PROOF		PLAN	
Step	Expression	Justification	Expression	Justification
(1)	PvQ	Given	PQ	Given
(2)	$-R \supset -Q$	"	RQ	"
(3)	S	"	S	"
(4)	$R \supset -S$	"	RS	"
(5)	$S \supset -R$	Rule 2 on (4)		
(6)	$S \supset -Q$	Rule 12 on (2), (5)	SQ	Rule 12 on (2), (4)
(7)	$-Q$	Rule 11 on (3), (6)	Q	Rule 11 on (3), (6)
(8)	$-P \supset Q$	Rule 6 on (1)		
(9)	$-Q \supset P$	Rule 2 on (8)		
(10)	P	Rule 11 on (7), (9)	P	Rule 11 on (1), (7)
(11)	PvT	Rule 9 on (10)	PT	Rule 9 on (10)

Figure 7

Solution of Problem A4 by Subject 8

subproblems of filling in the gaps (the other two subproblems are trivial, since they are solved simply by the translation of the abstracted expressions back into the original space). One of these subproblems is three steps in length, the other is two. Thus, for the original seven-step maze, the subject has substituted a four-step maze, a three-step maze, and a two-step maze.

To complete our illustration, let us see how the subject goes about solving the first subproblem—eliminating the R between expressions (2) and (4):

S. (Immediately following previous excerpt) Expressions (2) and (4)—we'll have to do something with them. If I invert expression (4)—apply rule 2 to it—I will have (S⊃-R). Good. O.K. Apply rule 2 to expression (4).

E. That gives [writing]: (5) S⊃-R.

S. Now apply rule 12 to expressions (2) and (4)—(2) and (5), I mean.

E. That gives [writing]: (6) S⊃-Q.

S. Right. I got rid of the R's. Now . . . . .

It will be observed that only rules 9, 11, and 12 are used in the derivation of the plan. All of these are essential rules. Rules 2 and 6, both of which are inessential, are used to solve the subproblems.

We can estimate how much reduction the planning heuristic accomplishes in the size of the maze to be searched. The number of alternative operations at each step is of the order of 10 (because of the distinction between essential and inessential operations, it may be smaller when the planning heuristic is used than when the problem is solved without it, but we will

ignore this additional source of efficiency of the planning heuristic). If m is 10, then the average number of paths to be searched without planning is $1/2 \cdot 10^7 = 5{,}000{,}000$. With planning, the number of paths is $1/2 \cdot 10^4 + 1/2 \cdot 10^3 + 1/2 \cdot 10^2 = 5{,}550$. The search required in the first case is larger by a ratio of 900:1.

Of course, these ratios assume that no other selective heuristic—apart from the planning heuristic—is employed. If the planning heuristic were superimposed, for example, on the functional analysis heuristic, the latter would reduce m to a much smaller number, hence there would be much less search either with, or without planning. Suppose, for example, that the functional analysis heuristic reduced m to 4. Then the search without planning would involve $1/2 \cdot 4^7 = 8{,}192$ paths; the search with planning would require $1/2 \cdot 4^4 + 1/2 \cdot 4^3 + 1/2 \cdot 4^2 = 168$. The savings ratio is now only 49:1.

The subject, understandably, was pleased with his heuristic. His comment on solving the problem was "See, I'm acquiring an insight." Since his protocol gives evidence of other bits of heuristic in addition to the ones we have been discussing, his value of m was probably 2 or less, and the total number of paths he searched was probably less than a dozen. The combination of heuristics he used, simple though they were, secured him a saving over blind trial and error of a factor of perhaps 500,000. These rough statistics give us a good picture of the reason for the "aha!" that goes with "insight" into the problem structure (which we would translate, "acquisition of an

additional heuristic").

Summary: The Nature of Heuristics

In this section we have seen that the success of a problem solver who is confronted with a complex task rests primarily on his ability to select—correctly—a very small part of the total problem-solving maze for exploration. The processes that carry out this selection we call heuristics. We have seen that most heuristics depend on a strategy that modifies subsequent search as a function of information obtained in previous search; and we have discussed at some length several of the most significant and powerful classes of heuristics that we have encountered in our attempts to simulate human problem solving.

Among the heuristics we examined were: processes for working backwards from the problem solution, selection heuristics, functional or means-end analysis, and planning. We provided operational meanings for these terms by sketching out what the actual processes would be in the Logic Theorist and in a chess-playing machine. We referred to our evidence from protocols of human subjects that such processes actually do occur in human problem-solving behavior. We also constructed quantitative estimates of the reduction in search that result from the selectivity of these heuristics, and used the estimates to account for the ability of humans, and of machines simulating them by using the same processes, to solve the particular problems in question.

Some Conditions of Creativity

In the remaining pages of this paper we shall use the theory of problem solving developed in preceding sections to cast light on three topics that are often discussed in relation to creativity:

(1) the use of imagery in problem solving;

(2) the relation of unconventionality to creativity;

(3) the role of hindsight in the discovery of new heuristics.

These three topics were chosen because we think our theory has something to say about them. We have not tried to include all the traditional topics in the theory of creative activity—we do not, for example, discuss the phenomenon of incubation—nor will we try to treat definitively the topics we have included. We are still far from having all the mechanisms that will be required for a complete theory of creativity. Hence, these last pages are necessarily extrapolations and are more speculative than the earlier sections.

Planning and Imagery

Among the issues that have surrounded the topic of imagery in the literature on thinking the following have been prominent:

1. What internal "language" is used by the organism in thinking—to what extent is this "language" related to the sense modalities, and is the thinking represented by elements that correspond to abstract "symbols" or to pictures, or to

something else?

2. To what extent do the internal representations, whatever their nature, involve generalization and abstraction from that which they represent?

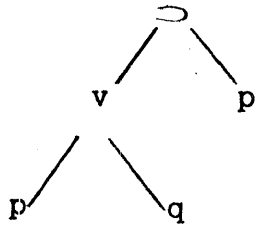
Using the example of planning we have been considering, we believe some clarification can be achieved of both issues.

Some Comments on Representation

How are the objects of thought represented internally? We are asking here neither a physiological nor a "hardware" question. We wish an answer at the level of information processing, rather than neurology or electronics. In a state description of an information-processing system, we can talk of patterns of elementary symbols. These symbols may be electric charges, as in some computer memories, or they may be the cell assemblies of Hebb's theories, or they may be something quite different. We are not interested in what they are made of. Given that there are some such patterns—that the system is an information processing system—our question is in what way the patterns within mirror, or fail to mirror the patterns without that they represent.

Let us take a simple example from logic. We may write on a piece of paper the expression "$(p \vee q) \supset p$." What would it mean to say that the "same" expression was held in memory by the Logic Theorist? With the present program it would mean that somewhere in memory there would be a branching pattern of

elementary symbols (or the internal counterparts of elementary symbols) that would look like:



Of course, there would not literally be mounds of ink like "p", but there would be internal elementary patterns in one-one correspondence to these. Note however, that the correspondence between the internal and external representations as a whole is far more complicated than the correspondences between elementary symbols. The external representation of the expression in the illustration is a linear array of symbols; the internal representation has branches that make it topologically distinct from a linear array. The external representation uses symbols like "(" and ")"; these are absent from the internal representations—the grouping relations they denote being implicit in the branching structure itself (i.e., the cluster pvq, which is enclosed in parentheses in external representation, is a subtree of the entire expression in the internal representation).

The implicitness of certain aspects of the internal representation goes even deeper than we have just indicated. For the tree structure we represented on the paper above by connecting symbols with lines is represented within the computer memory by the fact that there are certain information processes available that will "find the left subtree" and "find the right subtree" of such a tree structure. The actual physical

locations of these elements in memory can be (and usually are) completely scattered, so long as these information processes have means for finding them.

Let us take another example. If we wish to represent on paper the concept of a pair of elements, P and Q, abstracted from the order of the pair, we can write something like: (PQ), and append it to the statement that (PQ) is equivalent for all purposes with (QP). In an internal representation, order-independence of the terms of the pair might be secured in a quite different way. Suppose that the symbols P and Q were stored (in some order) on a list in memory, but that the only information processes available for dealing with lists were processes that produced the same output regardless of the order of the items on the list. Suppose, for example, that the "print list" process always alphabetized the list before printing. Then this process would always print out "(PQ)" regardless of whether the items were stored on the list as PQ or QP. In this case, the order-independence of the information processes applicable to the lists would be an implicit internal representation of the equivalence of (PQ) with (QP).[1/]

The main lesson that we learn from these examples is that the internal representation may mirror all the relevant properties

---

1/ A simple example of this in humans is well known to teachers of matrix algebra. Since all the elementary arithmetic and algebraic systems that students have encountered previously contain the commutative law, students must be taught that the matrix product AB is not equivalent to the product BA.

of the external representation without being a "picture" of it in any simple or straightforward sense. It is not at all clear whether a human subject would be aware that his internal representation of a logic expression "carried" the information about the expression in quite a different way from the string of symbols on paper, or that, if he were aware, he could verbalize what the differences were.

A similar point has been made in discussions of "encoding." Our examples show, however, that encoding may involve something far more complex than translating a string of symbols in one alphabet into another string of symbols in another alphabet. The encoded representation may not be a string at all, and there may be important differences in what is explicit and what implicit in the two representations.

Representation and the Sense Modalities.

Since the internal representation of information need not be a simple mapping of what is "out there," or even of what is received by the sense organs, it is not easy to know what is meant by saying that a particular internal representation is or is not "visual" or "auditory." Is the internal branching structure that represents the logic expression inside the Logic Theorist a visual image of the string of symbols on paper or is it not?

There is an obvious fallacy in saying that it is not just because the spatial (or even the topological) relations are not

the same in the two. The internal representations we carry around in our heads of even the most visual of pictures cannot possibly have the same metrical relations within (and possibly not even the same topologic relations) as without.

We believe that the explanation of why some memories are visual, some auditory, and some verbal lies in a quite different direction from a simple "mapping" theory. Since our explanation rests on considerations that have not even been touched upon in the present paper, we cannot discuss it at length. However, a very brief statement of it may help us understand the role of imagery in creative thought.

We will assert that an internal representation is visual if it is capable of serving as an input to the same information processes as those that operate on the internal representations of immediate visual sensory experiences. These information processes that can be applied to visual sensations literally serve as a "mind's eye," for they can operate on memories that have been encoded in the same way as sensory inputs, and when they are so applied produce the phenomena of visual imagination. Since there must be processes that can deal with sensory inputs, there is nothing mysterious in the notion that these same processes can deal with inputs from memory, and hence nothing metaphysical or non-operational about the concept of "mind's eye" or "mind's ear."

But the mind's eye is used not only to process inputs that "nature" coded in visual form. Often we deliberately construct

visual representations of abstract relations (e.g., we draw boxes to represent states of a system, and arrows connecting the boxes to represent the processes that transform one state into another). What can be the advantage of the imagery? The advantage lies in the fact that when we encode information so as to be accessible to visual processes, we have automatically built into the encoded information all the relations that are implicit in the information processes that constitute the mind's eye. For example, when we represent something as an arrow, we determine the order in which the items connected by the arrow will be called into attention.

We are led in this way to the concept of systems of imagery. A system of imagery comprises a set of conventions for encoding information, and coordinated with these a set of information processes that apply to the encoded information. As we have seen, the information processes for interpreting the encoded information may be just as rich in implicit conventions as the processes for encoding. It is the fact that the encoding makes available the former as well as the latter that makes it useful sometimes to represent information in a modality for which we have a rich and elaborate system of imagery.

Abstraction and Generalization.

Bishop Berkeley founded his epistemology on the personal difficulty he experienced in imagining a triangle which is "neither oblique nor rectangle, equilateral, equicrural nor

scalenon, but all and none of these at once." Hume, on the other hand, found this feat of imagination perfectly feasible!

The Logic Theorist would have to take Hume's side against Berkeley. For in the planning program the problem solver has the capacity to imagine a logic expression comprised of two variables joined by a connective, in which the connective is neither v nor · nor ⊃, but all and none of these at once. For this is precisely what the representation (PQ) stands for and the way in which it is used by the planning processes.

Once we admit that the relation between the object sensed and its internal representation is complex, there is no difficulty in admitting as corollary that the internal representation may abstract from all but a few of the properties of the object "out there." What we call "visual imagery", for example, may admit of colorless images even if all light that falls on the retina is colored.

The fact that the planning heuristic of the Logic Theorist possesses generalized or abstracted images of logic expressions does not prove, of course, that humans construct similar abstractions. What it does prove is that the notion of an image of a triangle "neither oblique nor rectangle, equilateral, equicrural, nor scalenon, but all and none of these at once" is not contradictory, but can be given a straightforward operational definition in an information-processing system. Finally, since the information processes that can operate on the abstracted expressions in the Logic Theorist are of the same kind as those

that operate on the full-bodied expressions, we would be forced, by any reasonable criterion, to regard the two images as belonging to the same modality.

The Uses of Imagery.

We have already hinted at the uses of imagery, but we would like now to consider them a little more explicitly. To think about something, that something must have an internal representation of some kind, and the thinking organism must have some processes that are capable of manipulating the representation. We have called such a combination of representation and processes a system of imagery.

Often, the term image is used somewhat more narrowly to refer to those representations that correspond to one or another of the sense modalities. Thus, we have visual images, auditory images, and tactile images, but we would not, in this narrower usage, speak of "abstract images"—i.e., representations and processes not used for representations of any of the sensory inputs.

When a particular representation is used for something, a large number of properties are imputed implicitly to the object represented because these properties are imbedded in the information processes that operate on representations of the kind in question. Thus, if we represent something as a line, we are likely—because that is the way our visual imagery operates—to impute to it the property of continuity.

Herein lies both the power and the danger of imagery as a tool of thought. The richer the properties of the system of imagery we employ, the more useful is the imagery in manipulating the representation, but the more danger there is that we will draw conclusions based on properties of the system of imagery that the object represented doesn't possess. When we are aware of the danger—and are conscious that we have encoded information into a system of imagery with strong properties—we are likely to call the image a "metaphor."

Often we are not aware of the danger. As has often been observed, Aristotle's logic and epistemology sometimes mistook accidents of Greek grammar for necessary truths. From this standpoint, the significance of modern mathematics, with its emphasis on rigor and the abstract axiomatic method, is that it provides us with tests that we can apply to the products of thinking to make sure that only those assumptions are being used that we are aware of.

The imagery used in the planning heuristic drastically reduces the space searched by the solution generator by abstracting from detail. This is probably not the only function of imagery for humans, although it is the one best documented by our present programs. We think there is evidence from data on human subjects that even in those cases where there is not a rich set of processes associated with the representation, imagery may provide a plan to the problem solver at least in the sense of a list of the elements he is dealing with and a list of which

of these is related. We will have to leave detailed discussion of this possibility to another occasion.

Summary: Imagery

We have applied our problem-solving theory to the classical problem of the role of imagery in thought. Although our analysis of imagery is admittedly speculative, it provides a possible explanation of the relation of internal representations to the sense modalities, and provides an example from one of the computer programs of generalization or abstraction, and of an abstract "visual" image. Finally, the theory shows how images of various kinds can be used as the basis for planning heuristics.

Unconventionality and Creativity

Thus far, our view of the problem-solving process has been a short-range one. We have taken as starting point a system of heuristics possessed by the problem solver, and have asked how it would govern his behavior. Since his initial system of heuristics may not enable the problem solver to find a solution in a particular case, we must also understand how a system of heuristics is modified and developed over time when it is not adequate initially.

Change of Set and Learning

Although all adaptive change in heuristics might be termed "learning," it is convenient to distinguish relatively short-run and temporary changes from longer-run more or less permanent

changes. If we use "learning" to refer only to the latter, then we may designate the former as "changes in set."

There is a basis for the distinction between set change and learning in the structure of the problem-solving organism. The human problem solver (and the machine simulation) is essentially a serial rather than a parallel instrument, which because of the narrow span of its attention, does only one or a few things at a time. If it has a rich and elaborate system of heuristics relevant to a particular problem, only a small part of these can be active in guiding search at any given moment. When in solving a problem one subsystem of heuristics is replaced by another, and the search, as a consequence, moves off in a new direction, we refer to this shift as a change in set. Change in set is a modification of the heuristics that are actively guiding search, by replacing them with other heuristics in the problem-solver's repertoire; learning is change in the repertoire of heuristics itself.

Stereotypy

A major function of heuristics is to reduce the size of the problem space so that it can be searched in reasonable time. Effective heuristics exclude those portions of the space where solutions don't exist or are rare, and retain those portions where solutions are relatively common. Heuristics that have been acquired by experience with some set of problems may be exceedingly effective for problems of that class, but may

prove inappropriate when used to attack new problems. Behaviorally, stereotypy is simply the subject's persistence in using a system of heuristics that the experimenter knows is inappropriate under the circumstances.

It is a very common characteristic of puzzles that the first steps toward solution require the solver to do something that offends common sense, experience, or physical intuition. Solutions to chess-mating problems typically begin with "surprising" moves. In the same way, a number of classical experiments with children and animals show that a simple problem of locomotion to a goal can be made more difficult if a barrier forces the subject to take his first steps away from the goal in order ultimately to reach it. When the task has this characteristic, the problem solver is obviously more likely to succeed if his repertoire of heuristics includes the injunction: "If at first you don't succeed, try something counterintuitive."

Is Unconventionality Enough?

It sometimes seems to be argued that people would become effective problem solvers if only we could teach them to be unconventional. If our analysis here is correct, unconventionality may be a necessary condition for creativity, but it is certainly not a sufficient condition. If unconventionality simply means rejecting some of the heuristics that restrict search to a limited subspace, then the effect of unconventionality will generally be a return to relatively inefficient

trial-and-error search in a very much larger space. We have given enough estimates of the sizes of the spaces involved, with and without particular heuristics, to cast suspicion on a theory of creativity that places its emphasis on increase in trial and error.

Let us state the matter more formally. Associated with a problem is a space of possible solutions. Since the problem solver operates basically in a serial fashion, these solutions must be taken up and examined in some order. If the problem solver has no information about the distribution of solutions in the space of possibilities, and no way of extracting clues from his search, then he must resort to a solution generator that is, to all intents and purposes, "random"—that leads him to solutions no more rapidly than would a chance selection. At some later stage the problem solver learns how to change the solution generator so that—at least for some range of problems—the average search required to find a solution is greatly reduced. But if the modified generator causes some elements in the solution space to be examined earlier than they would otherwise have been, it follows that the examination of others will be postponed.

The argument for unconventionality is that at some point a class of problems may be faced where the generator looks at just the wrong elements first, or even carefully filters out the right ones so that they will never be noticed (as in the chess example of queen sacrifices). A return to the original

trial-and-error generator would eliminate this perverse blindness of the generator, but at the expense of reinstating a search through an enormous space. What is needed in these cases is not an elimination of the selective power of a solution generator, but the replacement of the inappropriate generator by an appropriate one.

We have neither the data nor the space to illustrate this point from classical instances of scientific creativity, but we can give a simple example from chess. A chess novice is always stunned when his opponent demolishes him with a "creative" unconventional move like a sacrifice of a major piece. The novice has carefully trained himself to reject out of hand moves that lose pieces (and kicks himself for his oversights). If he tries to imitate his more experienced opponent, he usually loses the sacrificed piece. Clearly the opponent's secret is not simply that he is willing to be unconventional—to consider paths the novice rejects. The secret is that the experienced player has various additional pieces of heuristic that guide him to promising "unconventional" moves by giving him clues of their deeper and more devious consequences. It is the possession of this additional selectivity that allows him, in appropriate positions, to give up the selectivity embodied in the novice's rule of always preserving major pieces. The evidence we possess on the point indicates rather strongly that the amount of exploration undertaken by the chess master is no greater than that undertaken by relatively weak players (1).

He does not generate more solution possibilities; he does generate them in a different sequence.

Nature Abhors a Vacuum

We see that set change in particular, and unconventionality in general, are likely to facilitate the solution of a problem only if the problem solver has an appropriate new heuristic to replace the inappropriate heuristic that has been "blinding" him. Accordingly, to understand the success of effective and creative problem solvers we must examine not only the motivational and attitudinal factors that enable them to change an initial set or to violate accepted conventions; we must pay equal attention to the richness of their systems of heuristics that makes any particular piece of heuristic dispensable, and to their learning processes that generate new heuristics to fill the vacuums created by the rejection of the ones previously used.

Learning by Hindsight

Our experience with the simulation of learning has been much more limited than our experience with the simulation of problem solving. The chess-playing program is, to date, entirely a performance program; and only a few experiments have been carried out with learning heuristics for the Logic Theorist. Nevertheless, from these explorations and from our theoretical model we can draw some implications about learning

processes that help us understand how the creative problem solver can gradually improve his heuristics.

In the next two sections we will describe two kinds of learning that have actually been tested with the Logic Theorist. Both kinds of learning involve "hindsight," and in the third section, we shall undertake a more general analysis of the role of hindsight in the acquisition of new heuristics.

Memory of Specific Results

The simplest kind of learning in a maze is to remember the path to a solution so that the same solution can be reached at once in a later trial. There is no difficulty in programming a machine for this kind of learning, provided that its memory is large enough; and little enough difficulty for a human. Thus it is probable that most high school geometry students, unless they have an enlightened teacher, focus their energies on memorizing theorems and their proofs.

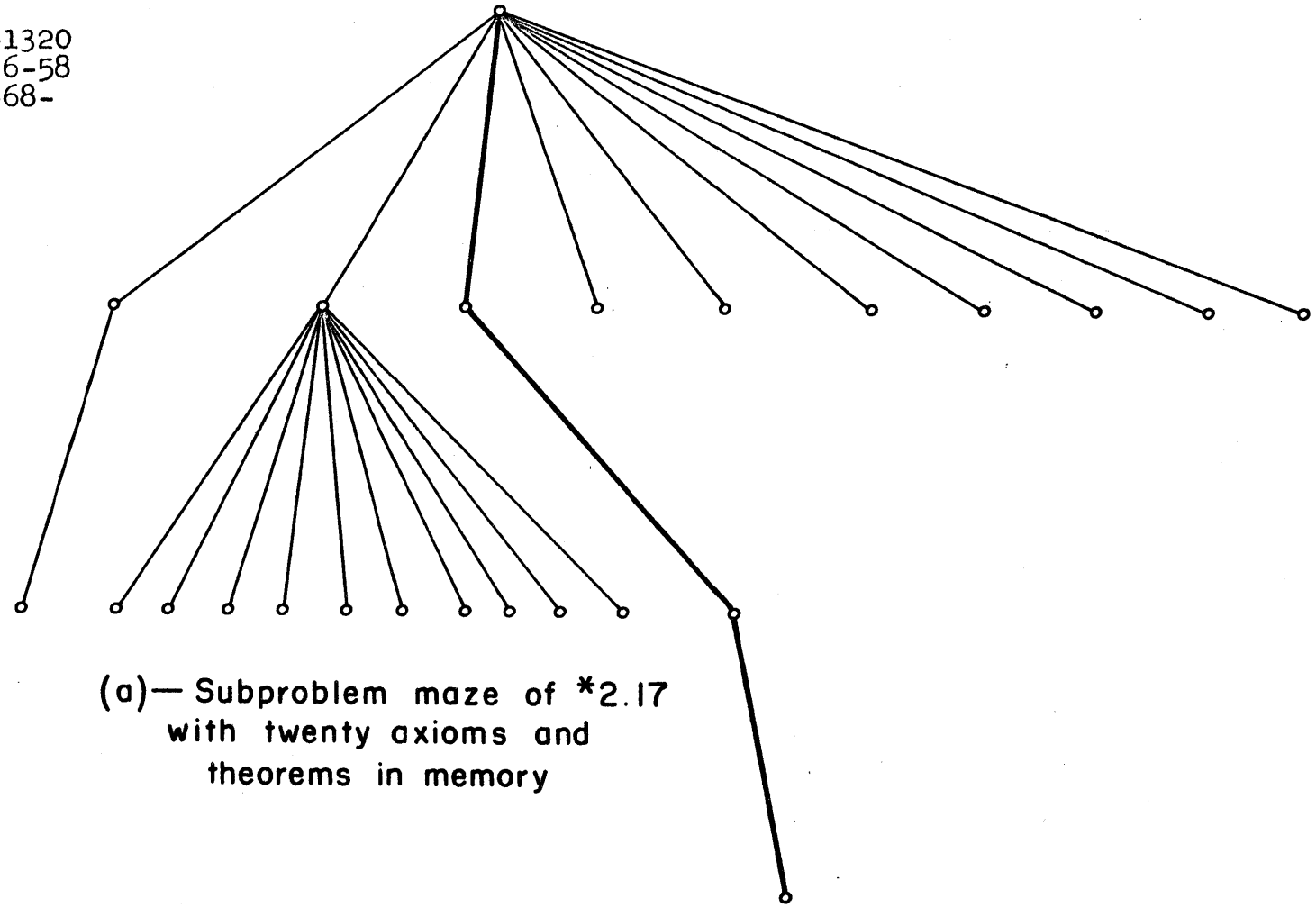
The Logic Theorist stores in memory the theorems it has proved (it could also remember the proofs themselves, but at present is not programmed to do so), and hence can use these as starting points in exploring new parts of the maze.

One should not underestimate the enhancement of problem-solving power that can be obtained even with this "routine" kind of learning, particularly if the teacher is careful to present tasks to the problem solver in an appropriate order. We have already seen how much the search for a long proof can be reduced if a plan is provided first; but a plan consists simply in

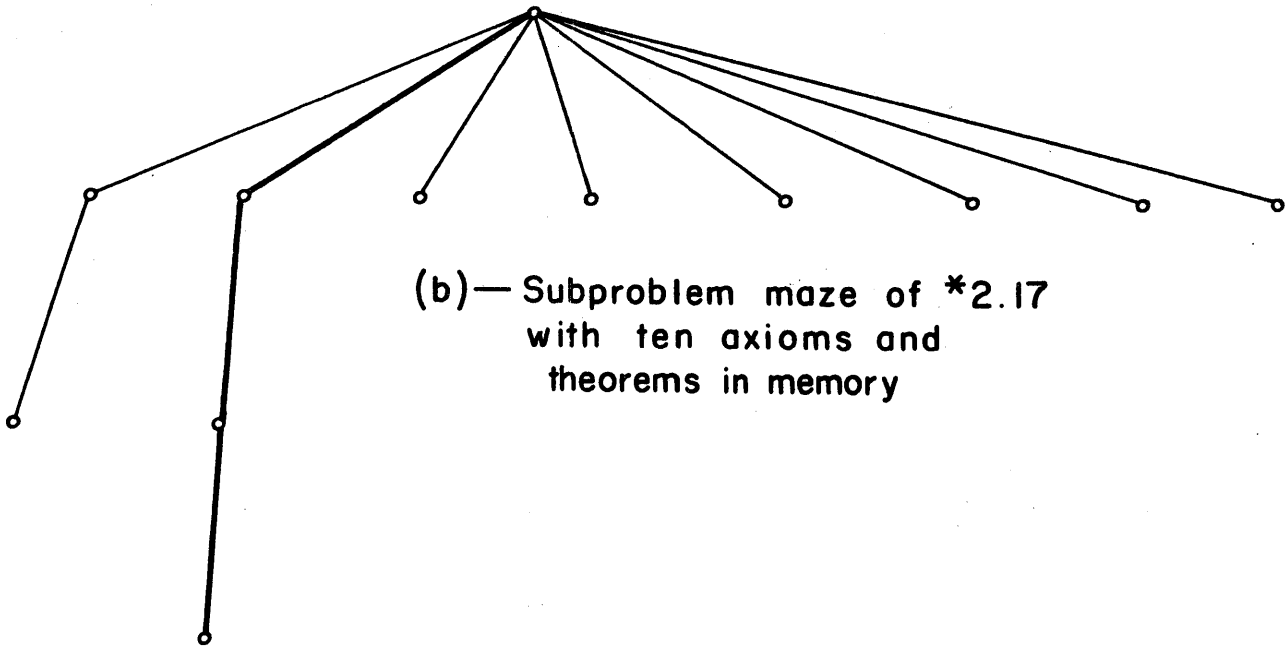
dividing the original problem into a series of smaller problems—marking the path the problem solver is to follow. Exactly the same effect can be secured if the subproblems generated by the planning heuristic are instead provided by the teacher.

On the other hand, storage of specific information about paths in the maze is not always helpful in subsequent problem solving. We have conducted some experiments with the Logic Theorist in which a theorem is presented for proof (a) after all previous theorems have been stored in memory, and alternatively (b) after a carefully selected small set of "powerful" theorems has been stored in memory. In a considerable number of cases, the program proves the theorem more quickly, and with far less search, in the second condition than in the first. For example, in one case (Theorem *2.48 of Principia) (17) the Theorist achieved a three-step proof when it had in memory only the axioms and one prior theorem (*2.16) in one-third the time it took to find a two-step proof when it held in memory all prior theorems. We are reminded by this example of the blinding effects of excesses of pedantry on human problem solvers also. A small arsenal of good general-purpose weapons may be much more effective than a storehouse of specific, narrowly useful ones.

A graphical impression of the qualitative difference that is produced in the Logic Theorist's problem-solving behavior when different numbers of prior theorems are held in memory can be obtained from Figure 8. The upper half of the figure



(a)— Subproblem maze of *2.17 with twenty axioms and theorems in memory



(b)— Subproblem maze of *2.17 with ten axioms and theorems in memory

Fig. 8

shows the maze of subproblems the Theorist explored while proving Theorem *2.17 with all axioms and prior theorems (twenty in all) held in memory; the lower half shows the maze explored while proving Theorem *2.17 with only the axioms and five theorems (ten in all) held in memory. In the former case, 23 branches had to be explored to find a three-step proof; in the latter case only 11 branches to find the same proof.

Differentiation: Specialized Methods

As the problem solver accumulates a larger and larger store of results and techniques, his problem of selection becomes more difficult unless he acquires at the same time additional clues on the basis of which to differentiate parts of the problem space in order to use special techniques under special circumstances. We have developed one example in the Logic Theorist of a process for learning specialized techniques. The Logic Theorist uses four basic methods of attack on problems. In each method it employs theorems already proved as its "raw materials." It turns out, empirically, that some theorems are used principally in connection with certain methods, other theorems with other methods. The Logic Theorist, when it has used a particular theorem in connection with a particular method to solve a problem, associates the theorem with that method. The next time it has occasion to use the same method, it tries theorems that have had a history of success with that method before it tries the other theorems.

To study the effects of introducing this learning of associations between particular methods and particular theorems, we performed the following experiment. As a pre-test, we instructed the Logic Theorist to attempt in sequence the first 52 theorems in Chapter 2 of Principia Mathematica (17), allowing it, when attempting a particular theorem, to use all prior theorems (whether it had succeeded in proving them or not), but not to use the special methods learning program. Then, we erased the results of this experience from memory, and as a test of the learning, instructed the Theorist to attempt the same 52 theorems, this time using the special methods learning program. The main result of the experiment can be seen by comparing the times required by the program to obtain proofs for the twenty theorems that were proved in both pre-test and test and whose proofs were not trivial. (We disregard 18 additional theorems proved on both runs, but having trivial, one-step proofs.)

The abscissa of each point in Figure 9 shows the time required to prove a theorem in the pre-test; the ordinate of that point, the time to prove the same theorem in the test run. The remarkable fact about this scatter diagram is that it consists of two straight lines, each containing about half of the points. For the points on the upper line, almost twice as long was required to discover a proof on the test run as on the pre-test; for the points on the lower line, less than half as long was required on the test as on the pre-test.

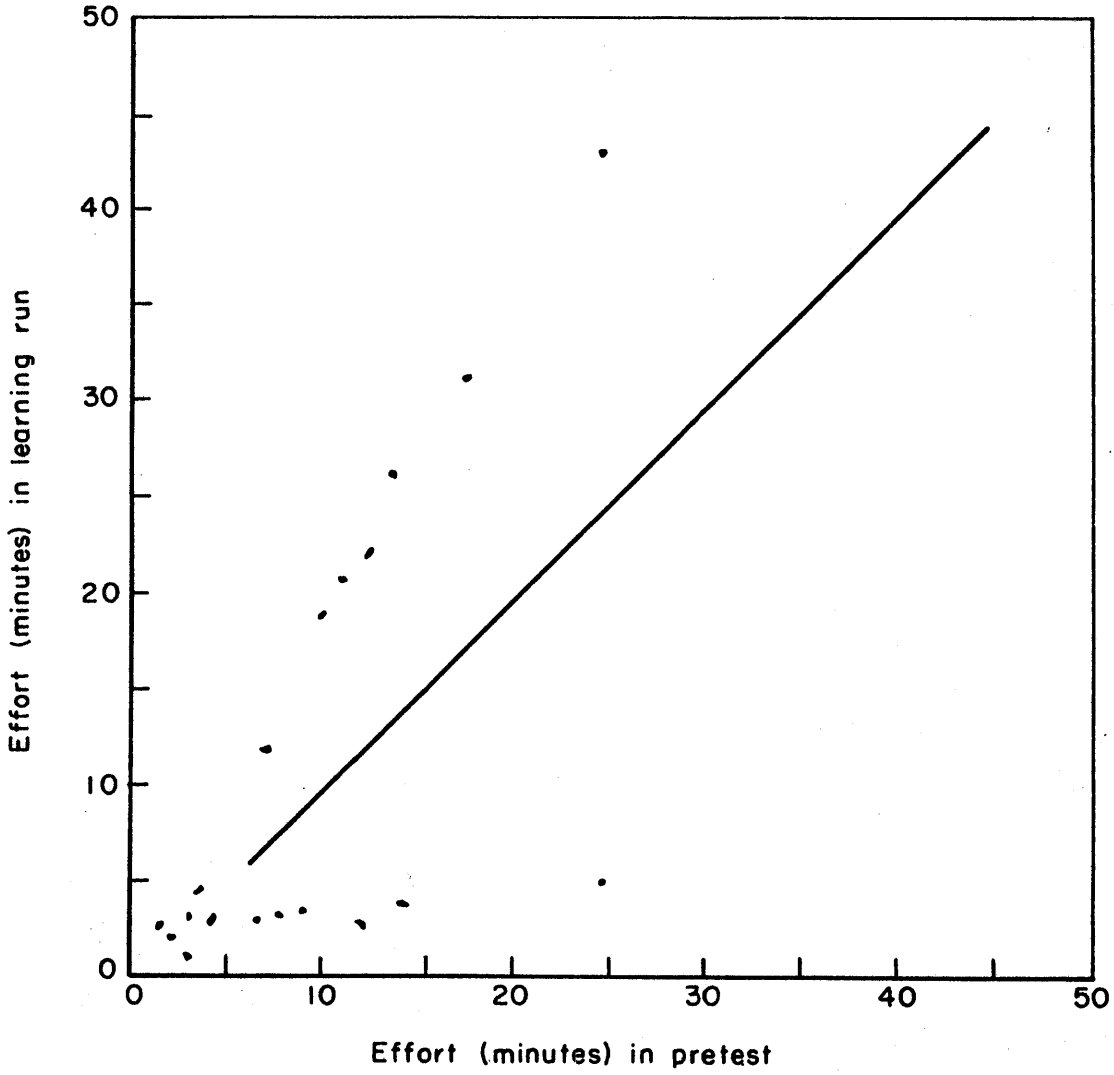


Fig. 9 — Effect of special methods learning on performance

Closer examination of the machine's protocols provides a simple explanation. In the test run, the program tried its special high-priority theorems first. Only when these failed—i.e., when the problem was of a new "type" that did not yield to any methods that had worked on previous problems—did the program fall back on its full store of available theorems. The additional time required in the test run for these problems was the time spent in the futile attempt to use the special theorem lists it had learned. On the other hand, where a problem yielded to proof by a method that had worked on a previous problem, this was soon discovered in the test run with a corresponding large improvement in performance. Comparison of the mazes for pretest and test runs of the latter group of problems reveals the characteristic difference—quite similar to that in Figure 2—between shallow widely-branching trees involving much search in the former case, and deep, sparsely-branching trees in the latter.

The Contribution of Hindsight to Heuristics

The learning programs we have mentioned have two important characteristics in common: (1) they consist in a gradual accumulation of selective principles that modify the sequence in which possible solutions will be examined in the problem space; (2) the selective principles are obtained by hindsight—by analysis of the program's successes in its previous problem-solving efforts. We believe that both of these characteristics

are to be found in most of the important learning processes in humans. Since we have already discussed the first at length, we will turn next to some comments on the second.

Creative problem solving is mysterious because it is hard to see how needles are found in haystacks without interminable search. We have tried to dispel the mystery of the performance by exhibiting devices that are capable of narrowing search to a very small part of the haystack. In one sense, however, this only pushes the mystery back. We can now regard the task of learning an effective problem-solving heuristic as itself a problem-solving task. The space of possible heuristics for problem solving is a space that must be enormous, even as problem spaces go. How do we find solutions in that space? How do we learn effective heuristics?

We must be careful not to overexplain the phenomenon by discovering learning mechanisms far more powerful than would be needed to account for the historical facts of scientific discovery. One of the key heuristics that underlies physical intuition in dynamics is the notion that forces produce changes in velocity (rather than producing velocities). Evidence from which this idea might be derived is available to anyone with eyes. Yet at least hundreds of man-years of search by highly intelligent men were required to discover the idea, and even after it was enunciated by Galileo, another century of work was required before even the most intelligent scientists had cleared it of all obscurity and confusion. We have an even

better-documented case in chess, where the game had a large literature and numbers of professional players for two centuries or more before Steinitz discovered some of the principles of positional play. From these and other examples we might conclude that the spaces that have to be searched to find important new heuristics are indeed large, and that the heuristics available for searching them are not generally very effective.

With this caution we can return to the question of how learning takes place—granted that it doesn't take place very often or very fast. Let us suppose that the Logic Theorist solves a difficult problem, and that it retains for a time in memory not only the correct path (the proof) that it finally discovered but also a record of the other paths it tried. One could then program it to re-examine various of the choice points at which it had not selected the correct branch on the first trial to look for relations between the state description at that point and characteristics of the correct branch. It could also be programmed to examine expressions just beyond the choice point along correct and incorrect paths in order to determine whether there were consistent differences between the expressions along the correct paths and those along the incorrect. Whatever differentia were discovered by such a program between correct and incorrect paths could be incorporated in the path generator. With the use of such procedures, a single successful experience of solving a problem after much trial-and-error search could become the basis for a great deal of learning.

More formally, suppose we wish to search a space of possible clues to determine which of these clues should be incorporated in a solution generator for maze paths. In terms of our general problem-solving theory we need a clue generator and a clue verifier. Hindsight contributes nothing to the construction of the clue generator, but it provides a cheap and effective verifier, since any possible clue we generate can be tested at once against a considerable number of instances.

We wish to offer a final comment on the "hindsight" aspects of learning. Suppose that, in terms of available computing power, the problem-solving organism can afford to explore only a few hundred paths in searching any particular problem space. Then an effective strategy for dealing with a large class of problems would be to abandon problems that did not yield solutions after moderate effort had been applied, to do learning by hindsight on the easier problems that proved solvable, and thus gradually to add to the number of problems that could be handled successfully.

Assume that we have a class of problems with an initial value of $\underline{m}=10$. Then, if the problem-solver had a limit of 500 paths per problem, he would only be able to solve problems of length $\underline{k}=2$ or less. As learning proceeded, however, the new heuristic would reduce the effective value of $\underline{m}$. By the time that $\underline{m}$ had been reduced to 6, problems of length 3 could be solved; when $\underline{m}$ reached 2, problems of length 8 could be

solved; and reductions of m below an average level of 2 would increase very rapidly the lengths of the problems that could be handled.

Concluding Remarks

In this paper we have treated creative activity as problem-solving activity characterized by the novelty and difficulty of the task. We have proposed an explanation for creative problem solving, developing this explanation along three parallel lines: (a) by constructing an abstract model of problem-solving behavior that provides operational meaning to such concepts as "problem difficulty" and "power of a heuristic"; (b) by specifying programs for digital computers that simulate human problem-solving behavior, and using the abstract model to understand the effectiveness of the programs for solving problems; and (c) by re-examining some of the classical problems in the literature of problem solving and creativity to see what light the theoretical model, the computer programs, and data on human behavior cast on them.

The main results of our investigations up to the present time are embodied in a number of computer programs, some of which have actually been run on a machine, some of which are coded but have not been run, and some of which are specified at the level of flow diagrams. The programs that we have referred to specifically here include: (a) the original program of the Logic Theorist, adapted to proving theorems in Whitehead and Russell's Principia (17), (b) a learning program that modifies

this basic program, permitting the Logic Theorist to learn to use special methods for special classes of problems, (c) a revision of the Logic Theorist that adapts it to solving logic problems in the form used by O. K. Moore in his experiments with human subjects and that incorporates a program for functional means-end analysis, (d) a supplement to this last program that gives it the capacity to construct plans, (e) a program for a chess-playing machine. A number of other programs that are in process of construction by members of the RAND-Carnegie group and by others were not specifically mentioned here, but have provided some of the background for our theorizing.

Data obtained by comparing in detail the operation of these programs with the behavior of human subjects is limited. We have now accumulated human protocols that will permit such comparison for both the logic and chess programs, but our main tests of the theory have thus far been of a grosser sort. We would chiefly rely on the fact that we have specified programs enabling mechanisms to solve complex problems so large that they would not yield to a brute-force approach, using even the most powerful computers. The success of these programs in obtaining problem solutions is the primary evidence for the theory of the problem-solving process that underlies their design.

We should like to stress our specific findings less than the methodology we have described for understanding the human mind. The use of computer programs to simulate information

processes allows us to study the behavior of systems of great complexity—far greater complexity than can be handled reliably with either verbal or classical mathematical techniques. We have constructed a theory of human thinking in terms of its underlying information processes, and we have indicated how the theory can give precision to topics that, however important, have in the past been discussed in exceedingly vague terms. We have, for example, identified in the program of the Logic Theorist notions like "grasp of problem structure," "visual image," "abstraction," and "set."

Some of the programs we have described perform work that is considered difficult, and even mildly creative, when it is done by humans. Although these programs fall considerably short in performance of the highest levels of creativity of which humans are capable, there is every reason to suppose that they are qualitatively of the same genus as these more complex human problem-solving processes. In another place, we have predicted that within ten years a computer will discover and prove an important mathematical theorem, and compose music that is regarded as aesthetically significant. On the basis of our experience with the heuristics of logic and chess, we are willing to add the further prediction that only moderate extrapolation is required from the capacities of programs already in existence to achieve the additional problem-solving power needed for such simulation.

A P P E N D I X

It may be helpful to the reader, in following the specific examples in the text, to have a brief description of the problem-solving task involving logic expressions that was designed by O. K. Moore and Scarvia B. Anderson.

A logic expression is a sequence of symbols of two types: (1) variables—P, Q, R, etc.—and (2) connectives—not (–), and (·) or (v), and implies (⊃). An example from the text is R·(–P⊃Q), which may be interpreted as "R and (not-P implies Q)." The subjects are not provided with this interpretation, however, but are told that the expressions are code messages and that the connectives are named "tilde" (–), "dot" (·), "wedge" (v), and "horseshoe" (⊃).

The following rules are provided for transforming one or two given logic expressions into a new expression (recoding expressions). We will state them here only approximately, omitting certain necessary qualifications.

One-Line Rules

1. $AvB \Leftrightarrow BvA$
   $A \cdot B \Leftrightarrow B \cdot A$

2. $A \supset B \Leftrightarrow -B \supset -A$

3. $AvA \Leftrightarrow A$
   $A \cdot A \Leftrightarrow A$

4. $Av(BvC) \Leftrightarrow (AvB)vC$
   $A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$

5. $AvB \Leftrightarrow -(-A \cdot -B)$
   $A \cdot B \Leftrightarrow -(-Av-B)$

6. $A \supset B \Leftrightarrow -AvB$
   $AvB \Leftrightarrow -A \supset B$

7. $Av(B \cdot C) \Leftrightarrow (AvB) \cdot (AvC)$
   $A \cdot (BvC) \Leftrightarrow (A \cdot B)v(A \cdot C)$

8. $A \cdot B \Rightarrow A$
   $A \cdot B \Rightarrow B$

9. $A \Rightarrow AvX$, where X is any expression

The rules can be applied to complete expressions, or (except rule 8) to subexpressions. Double tildes cancel, i.e., $--A \Leftrightarrow A$ but this cancellation is not stated in a separate rule.

Two-Line Rules

10. If A and B are given, they can be recoded into $A \cdot B$.
11. If A and $A \supset B$ are given, they can be recoded into B.
12. If $A \supset B$ and $B \supset C$ are given, they can be recoded into $A \supset C$.

Subjects were instructed in the use of these rules, then were given problems like those described in the text. They were asked to think aloud while working on the problems, and each time they applied a rule to recode one or two given expressions, the new expression was written on the blackboard by the experimenter, together with the numbers of the expressions and rule used to obtain it.

By inspection of the rules it can be seen that in the planning space, where connectives and the order of the symbols are disregarded, rules 1, 2, 5 and 6 would leave expressions unchanged. These are the inessential rules; the others, in altered form, become the essential rules. Rule 8, for example, becomes simply: $AB \Rightarrow A$.

REFERENCES

1. De Groot, A. D., Het Danken van den Schaker, Amsterdam, Noord-Hollandsche Uitgevers Maatschappij, 1946.

2. Dinneen, G. P., "Programming Pattern Recognition," Proceedings of the 1955 Western Joint Computer Conference, Institute of Radio Engineers, 1955, pp. 94-100.

3. Duncker, K., "On Problem-Solving," Psychol. Monogr., 58, No. 270, 1945.

4. Johnson, D. M., The Psychology of Thought and Judgment, Harper, New York, 1955.

5. Moore, O. K. and Scarvia Anderson, "Search Behavior and Problem Solving," Amer. Sociol. Rev., 1954, 19, pp. 702-714.

6. Newell, A. and H. A. Simon, "The Logic Theory Machine," IRE Transactions on Information Theory, Vol. IT-2, No. 3, September, 1956.

7. Newell, A., J. C. Shaw and H. A. Simon, "Empirical Explorations of the Logic Theory Machine: a Case Study in Heuristics," Proceedings of the Western Joint Computer Conference, Institute of Radio Engineers, February, 1957.

8. Newell, A. and J. C. Shaw, "Programming the Logic Theory Machine," Proceedings of the Western Joint Computer Conference, February, 1957.

9. Newell, A., J. C. Shaw and H. A. Simon, "Elements of a Theory of Human Problem Solving," Psychol. Rev., 1958, 65, No. 3.

10. Newell, A., J. C. Shaw and H. A. Simon, "Chess Playing Programs and the Problem of Complexity," J. Res. and Development, IBM Corporation.

11. Patrick, Catherine, "Creative Thought in Poets," Arch. Psychol., 1935, 178.

12. Patrick, Catherine, "Creative Thought in Artists," J. Psychol., 1937, 4, pp. 35-73.

13. Polya, G., Mathematics and Plausible Reasoning, Princeton University Press, Princeton, 1954.

14. Polya, G., How to Solve It, Doubleday, New York, 1957.

15. Selfridge, O. G. "Pattern Recognition and Modern Computers," Proceedings of the 1955 Western Joint Computer Conference, Institute of Radio Engineers, 1955, pp. 91-93.

16. Wallas, G., The Art of Thought, Harcourt, New York, 1926.

17. Whitehead, A. N. and Russell, B., Principia Mathematica, University Press, Cambridge, 1925-1927.