HUMAN PROBLEM SOLVING

The State of the Theory in 1970

Herbert A. Simon and Allen Newell
Carnegie-Mellon University*

When the magician pulls the rabbit from the hat, the spectator can respond either with mystification or with curiosity. He can enjoy the surprise and the wonder of the unexplained (and perhaps inexplicable), or he can search for an explanation.

Suppose curiosity is his main response--that he adopts a scientist's attitude toward the mystery. What questions should a scientific theory of magic answer? First, it should predict the performance of a magician handling specified tasks--producing a rabbit from a hat, say. It should explain how the production takes place, what processes are used, and what mechanisms perform those processes. It should predict the incidental phenomena that accompany the magic--the magician's patter and his pretty assistant--and the relation of these to the mystification process . . . It should show how changes

---

* (Note by Mr. Simon). Since my citation last year recognized that the work for which it was awarded was done by a team, rather than an individual, it is appropriate that Allen Newell, with whom I have been in full partnership from the very beginning of the effort, should be enlisted into coauthorship of this report on it. Both of us would like to acknowledge our debts to the many others who have been members of the team during the past decade and a half, but especially to J. C. Shaw and Lee W. Gregg. This paper is based on the final chapter of the authors' forthcoming book, HUMAN PROBLEM SOLVING (Prentice-Hall, 1971).

The research reported here was supported in part by United States Public Health Service Research Grant MH-07722, from the National Institute of Mental Health.

in the attendant conditions--both changes "inside" the members of the audience and changes in the feat of magic--alter the magician's behavior. It should explain how specific and general magician's skills are learned, and what the magician "has" when he has learned them.

Theory of Problem Solving--1958

Now I have been quoting--with a few word substitutions--from a paper published in the Psychological Review in 1958 (Newell, Shaw, and Simon, 1958). In that paper, titled "Elements of a Theory of Human Problem Solving," our research group reported on the results of its first two years of activity in programming a digital computer to perform problem solving tasks that are difficult for humans. Problem solving was regarded by many, at that time, as a mystical, almost magical, human activity--as though the preservation of human dignity depended on Man's remaining inscrutable to himself, on the magic-making processes remaining unexplained.

In the course of writing the "Elements" paper, we searched the literature of problem solving for a statement of what it would mean to explain human problem solving, of how we would recognize an explanation if we found one. Failing to discover a statement that satisfied us, we manufactured one of our own--essentially the paragraph I read earlier. Let me read it again, with the proper words restored, so that it will refer to the magic of human thinking and problem solving, instead of stage magic.

> What questions should a theory of problem solving answer? First, it should predict the performance of a problem solver handling specified tasks. It should explain

> how human problem solving takes place: what processes are used, and what mechanisms perform these processes. It should predict the incidental phenomena that accompany problem solving, and the relation of these to the problem-solving process. . . . It should show how changes in the attendant conditions--both changes "inside" the problem solver and changes in the task confronting him--alter problem-solving behavior. It should explain how specific and general problem solving skills are learned, and what it is that the problem solver "has" when he has learned them.

A Strategy

This view of explanation places its central emphasis upon process--upon how particular human behaviors come about, upon the mechanisms that enable them. We can sketch out the strategy of a research program for achieving such an explanation, a strategy that the actual events have been following pretty closely, at least through the first eight steps:

1. Discover and define a set of processes that would enable a system capable of storing and manipulating patterns to perform complex non-numerical tasks, like those a human performs when he is thinking.

2. Construct an information processing language, and a system for interpreting that language in terms of elementary operations, that will enable programs to be written in terms of the information processes that have been defined, and will permit those programs to be run on a computer.

3. Discover and define a program, written in the language of information processes, that is capable of solving some class of problems that humans find difficult. Use whatever evidence is available to incorporate in the program processes that resemble those used by humans. (Do not admit processes, like very rapid arithmetic, that humans are known to be incapable of.)

4. If the first three steps are successful, obtain data, as detailed as possible, on human behavior in solving the same problems as those tackled by the program.

Search for the similarities and differences between the behavior of program and human subject. Modify the program to achieve a better approximation to the human behavior.

5. Investigate a continually broadening range of human problem solving and thinking tasks, repeating the first four steps for each of them. Use the same set of elementary information processes in all of the simulation programs, and try to borrow from the subroutines and program organization of previous programs in designing each new one.

6. After human behavior in several tasks has been approximated to a reasonable degree, construct more general simulation programs that can attack a whole range of tasks--winnow out the "general intelligence" components of the performances, and use them to build this more general program.

7. Examine the components of the simulation programs for their relation to the more elementary human performances that are commonly studied in the psychological laboratory: rote learning, elementary concept attainment, immediate recall, and so on. Draw inferences from simulations to elementary performances, and vice versa, so as to use standard experimental data to test and improve the problem solving theories.

8. Search for new tasks (e.g., perceptual and language tasks) that might provide additional arenas for testing the theories and drawing out their implications.

9. Begin to search for the neurophysiological counterparts of the elementary information processes that are postulated in the theories. Use neurophysiological evidence to improve the problem solving theories, and inferences from the problem solving theories as clues for the neurophysiological investigations.

10. Draw implications from the theories for the improvement of human performance--for example, the improvement of learning and decision making. Develop and test programs of application.

11. Review progress to date, and lay out a strategy for the next period ahead.

Of course, life's programs are not as linear as this strategy, in the simplified form in which we have presented it. A good strategy would have to contain many checkpoints for evaluation of progress, many feedback loops, many branches, many iterations. Step 1 of the strategy, for example, was a major concern of our research group (and other investigators as well) in 1955-56, but new ideas, refinements and improvements have continued to appear up to the present time. Step 7 represented a minor part of our activity as early as 1956, became much more important in 1958-61, and has remained active since.

Nor do strategies spring full-grown from the brow of Zeus. Fifteen years' hindsight makes it easy to write down the strategy in neat form. If anyone had attempted to describe it prospectively in 1955, his version would have been much cruder, and probably would be lacking some of the last six steps.

The Logic Theorist

The "Elements" paper of 1958 reported a successful initial pass through the first three steps in the strategy. A set of basic information processes for manipulating non-numerical symbols and symbol structures had been devised (Newell and Simon, 1956). A class of information processing or list processing languages had been designed and implemented, incorporating the basic information processes, permitting programs to be written in terms of them, and enabling these programs to be run on computers (Newell and Shaw, 1957). A program, The Logic Theorist (LT), had been written in one of these languages, and had been shown, by running it on a computer, to be capable of solving problems that are difficult for humans (Newell, Shaw and Simon, 1957).

LT was, first and foremost, a demonstration of sufficiency. The program's ability to discover proofs for theorems in logic showed that, with no more capabilities than it possessed--capabilities for reading, writing, storing, erasing, and comparing patterns--a system could perform tasks that, in humans, require thinking. To anyone with a taste for parsimony it suggested (but, of course, did not prove) that only these capabilities, and no others, should be postulated to account for the magic of human thinking. Thus, the "Elements" paper proposed that "an explanation of the observed behavior of the organism is provided by a program of primitive information processes that generate this behavior," and exhibited LT as an example of such an explanation.

The sufficiency proof, the demonstration of problem solving capability at the human level, is only a first step toward constructing an information processing theory of human thinking. It only tells us that in certain stimulus situations the correct (that is to say, the human) gross behavior can be produced. But this kind of blind S-R relation between program and behavior does not explain the process that brings it about. We do not say that we understand the magic because we can predict that a rabbit will emerge from the hat when the magician reaches into it. We want to know how it was done--how the rabbit got there. Programs like LT are explanations of human problem solving behavior only to the extent that the processes they use to discover solutions are the same as the human processes.

LT's claim to explain process as well as result rested on slender evidence, which was summed up in the "Elements" paper as follows:

> First, . . . (LT) is in fact capable of finding proofs for theorems--hence incorporates a system of processes that is sufficient for the problem-solving mechanism. Second, its ability to solve a particular problem depends on the sequence in which problems are presented to it in much the same way that a human subject's behavior depends on this sequence. Third, its behavior exhibits both preparatory and directional set. Fourth, it exhibits insight both in the sense of vicarious trial and error leading to "sudden" problem solution, and in the sense of employing heuristics to keep the total amount of trial and error within reasonable bounds. Fifth, it employs simple concepts to classify the expressions with which it deals. Sixth, its program exhibits a complex organized hierarchy of problems and subproblems.

There were important differences between LT's processes and those used by human subjects to solve similar problems. Nevertheless, in one fundamental respect that has guided all the simulations that have followed LT, the program did indeed capture the central process in human problem solving: LT used heuristic methods to carry out highly selective searches, hence to cut down enormous problem spaces to sizes that a slow, serial processor could handle. Selectivity of search, not speed, was taken as the key organizing principle, and essentially no use was made of the computer's ultra-rapid arithmetic capabilities in the simulation program. Heuristic methods that make this selectivity possible have turned out to be the central magic in all human problem solving that has been studied to date.

Thus, in the domain of symbolic logic in which LT worked, obtaining by brute force the proofs it discovered by selective search would have meant examining enormous numbers of possibilities--ten raised to an exponent of hundreds or thousands. LT typically searched trees of fifty or so branches in constructing the more difficult proofs that it found.

Mentalism and Magic

LT demonstrated that selective search employing heuristics permitted a slow serial information processing system to solve problems that are difficult for humans. The demonstration defined the terms of the next stages of inquiry: to discover the heuristic processes actually used by humans to solve such problems, and to verify the discovery empirically.

We will not discuss here the methodological issues raised by the discovery and cerification tasks, apart from one preliminary comment. An explanation of the processes involved in human thinking requires reference to things going on inside the head. American behaviorism has been properly skeptical of "mentalism"--of attempts to explain thinking by vague references to vague entities and processes hidden beyond reach of observation within the skull. Magic is explained only if the terms of explanation are less mysterious than the feats of magic themselves. It is no explanation of the rabbit's appearing from the hat to say that it "materialized."

Information processing explanations, refer frequently to processes that go on inside the head--in the mind, if you like--and to specific properties of human memory: its speed and capacity, its organization. These references are not intended to be in the least vague. What distinguishes the information processing theories of thinking and problem solving described here from earlier discussion of mind is that terms like "memory" and "symbol structure" are now pinned down and defined in sufficient detail to embody their referents in precisely stated programs and data structures.

An internal representation, or "mental image," of a chess board, for example, is not a metaphorical picture of the external object, but a symbol structure with definite properties on which well-defined processes can operate to retrieve specified kinds of information (Baylor and Simon, 1966; Simon and Barenfeld, 1969).

The programmability of the theories is the guarantor of their operationality, an iron-clad insurance against admitting magical entities into the head. A computer program containing magical instructions does not run, but it is asserted of these information processing theories of thinking that they can be programmed and will run. They may be empirically correct theories about the nature of human thought processes or empirically invalid theories; they are not magical theories.

Unfortunately, the guarantee provided by programability creates a communication problem. Information processing languages are a barrier to the communication of the theories as formidable as the barrier of mathematics in the physical sciences. The theories become fully accessible only to those who, by mastering the languages, climb over the barrier. Any attempt to communicate in natural language must perforce be inexact.

There is the further danger that, in talking about these theories in ordinary language, the listener may be seduced into attaching to terms their traditional meanings. If the theory speaks of "search," he may posit a little homunculus inside the head to do the searching; if it speaks of "heuristics" or "rules of thumb," he may introduce the same Homunculus to remember and apply them. Then, of course, he will be interpreting the theory magically, and will object that it is no theory.

The only solution to this problem is the hard solution. Psychology is now taking the road taken earlier by other sciences, is introducing essential formalisms to describe and explain its phenomena. Natural language formulations of the phenomena of human thinking did not yield explanations of what was going on, formulations in information processing languages appear to be yielding such explanations. And the pain and cost of acquiring the new tools must be far less than the pain and cost of trying to master difficult problems with inadequate tools.

Our account today will be framed in ordinary language. But we must warn you that it is a translation from information processing languages which, like most translations, has probably lost a good deal of the subtlety of the original. In particular, we warn you against attaching magical meanings to terms that refer to entirely concrete and operational phenomena taking place in fully defined and operative information processing systems. The account will also be Pittsburgh-centric. It will refer mainly to work of the Carnegie-RAND group, although information processing psychology enlists an ever-growing band of research psychologists, many of whom are important contributors of evidence to the theory presented here.

Theory of Problem Solving--1970

The dozen years since the publication of the "Elements" paper has seen a steady growth of activity in information processing psychology--both in the area of problem solving and in such areas as learning, concept formation, short-term memory phenomena, perception and language behavior.

Firm contact has been made with more traditional approaches, and information processing psychology has joined (or been joined by) the main stream of scientific inquiry in experimental psychology today.[1] Instead of tracing history here, we should like to give a brief account of the product of the history, of the theory of human problem solving that has emerged from the research.

The theory makes reference to an information processing system, the problem solver, confronted by a task. The task is defined objectively (or from the viewpoint of an experimenter, if you prefer) in terms of a task environment. It is defined by the problem solver, for purposes of attacking it, in terms of a problem space. The shape of the theory can be captured by four propositions:[2]

(1) A few, and only a few, gross characteristics of the human information processing system are invariant over task and problem solver.

(2) These characteristics are sufficient to determine that a task environment is represented (in the information processing system) as a problem space, and that problem solving takes place in a problem space.

(3) The structure of the task environment determines the possible structures of the problem space.

(4) The structure of the problem space determines the possible programs that can be used for problem solving.

These are the bones of the theory. In the next pages, we will undertake to clothe them in some flesh.

1/ We have undertaken a brief history of these developments in an Appendix to our HUMAN PROBLEM SOLVING, loc. cit.

2/ Ibid., Chapter 14.

Characteristics of the Information Processing System

When human beings are observed working on well-structured problems that are difficult but not unsolvable for them, their behaviors reveal certain broad characteristics of the underlying neurophysiological system that supports the problem solving processes; but at the same time, the behaviors conceal almost all of the detail of that system.

The basic characteristics of the human information processing system that shape its problem solving efforts are easily stated: The system operates essentially serially, one-process-at-a-time, not in parallel fashion. Its elementary processes take tens or hundreds of milliseconds. The inputs and outputs of these processes are held in a small short-term memory with a capacity of only a few symbols. The system has access to an essentially infinite long-term memory, but the time required to store a symbol in that memory is of the order of seconds or tens of seconds.

These properties--serial processing, small short-term memory, infinite long-term memory with fast retrieval but slow storage--impose strong constraints on the ways in which the system can seek solutions to problems in larger problem spaces. A system not sharing these properties--a parallel system, say, or one capable of storing symbols in long-term memory in milliseconds instead of seconds--might seek problem solutions in quite different ways from the system we are considering.

The evidence that the human system has the properties we have listed comes partly from problem solving behavior itself. No problem solving behavior has been observed in the laboratory that seems

interpretable in terms of simultaneous rapid search of disjoint parts of the solver's problem space. On the contrary, the solver always appears to search sequentially, adding small successive accretions to his store of information about the problem and its solution.[1/]

Additional evidence for the basic properties of the system as well as data for estimating the system parameters comes from simpler laboratory tasks. The evidence for the five or ten seconds required to store a symbol in long-term memory comes mainly from rote memory experiments; for the seven-symbol capacity of short-term memory, from immediate recall experiments; for the 200 milliseconds needed to transfer symbols into and out of short-term memory, from experiments requiring searches down lists or simple arithmetic computations.[2/]

These things we do learn about the information processing system that supports human thinking--but it is significant that we learn little more, that the system might be almost anything so long as it meets these few structural and parametral specifications. The detail

---

1/ Claims that human distractability and perceptual capability imply extensive parallel processing have been refuted by describing or designing serial information processing systems that are distractable and possess such perceptual capabilities. (We are not speaking of the initial "sensory" stages of visual or auditory encoding, which certainly involve parallel processing, but of the subsequent stages, usually called perceptual.) For further discussion of this issue see Simon (1967) and Simon and Barenfeld (1969). Without elaborating here, we also assert that incremental growth of knowledge in the problem space is not incompatible with experiences of sudden "insight." For further discussion of this point see Newell, Shaw and Simon (1962) and Simon (1966).

2/ Some of this evidence is reviewed in Newell and Simon (1971), Chapter 14.

is elusive because the system is adaptive. For a system to be adaptive means that it is capable of grappling with whatever task environment confronts it. Hence, to the extent a system is adaptive, its behavior is determined by the demands of that task environment rather than by its own internal characteristics. Only when the environment stresses its capabilities along some dimension--presses its performance to the limit--do we discover what those capabilities and limits are, and are we able to measure some of their parameters.[1/]

Structure of Task Environments

If the study of human behavior in problem situations reveals only a little about the structure of the information processing system, it reveals a great deal about the structure of task environments.

Consider the cryptarithmetic problem

```
  DONALD
+GERALD
 ROBERT
```

which has been studied on both shores of the Atlantic, in England by Bartlett and in the United States in our own laboratory.[2/] The problem is to substitute numbers for the letters in the three names in such a way as to produce a correct arithmetic sum. As the problem is usually posed, the hint is given that D=5. If we look at the protocols of subjects who solve the problem, we find that they all substitute numbers for the letters in approximately the same sequence. First, they set T=0, then E=9 and R=7, then A=4 and L=8, then G=1, then N=6 and B=3, and finally, O=2.

1/ Simon, 1969, Chapters 1 and 2.

2/ Bartlett, 1958; Newell, 1967; Newell and Simon, 1971, Part II.

To explain this regularity in the sequence of assignments, we must look first at the structure of the task itself. A cryptarithmetic problem may be tackled by trying out various tentative assignments of numbers to letters, rejecting them and trying others if they lead to contradictions. In the DONALD+GERALD problem, hundreds of thousands of combinations would have to be tried to find a solution in this way. (There are 9!=362,880 ways of assigning nine digits to nine letters.) A serial processor able to make and test five assignments per minute would require a month to solve the problem; many humans do it in ten minutes or less.

But the task structure admits a heuristic that involves processing first those columns that are most constrained. If two digits in a single column are already known, the third can be found by applying the ordinary rule of arithmetic. Hence, from D=5, we obtain the rightmost column: 5+5=T, hence T=0, with a carry of 1 to the next column. Each time a new assignment is made in this way, the information can be carried into other columns where the same letter appears, and then the most-constrained column of those remaining can be selected for processing. For the DONALD+GERLAD problem (but not, of course, for all cryptarithmetic problems) it turns out that the correct assignments for T, E, R, A, L and G can all be found in this way without any trial-and-error search whatsoever, leaving only N, B and O for the possible permutations of 6, 3 and 2.

Not only does this heuristic of processing the most-constrained columns first almost eliminate the need for search, but it also reduces the demands on the short-term memory of the problem solver. All the

information he has acquired up to any given point can be represented on a single external display, simply by replacing each letter by the digit assigned to it as soon as the assignment is made. Since the assignments are definite, not tentative, no provision need be made by an error-free processing system for correcting wrong assignments, nor for keeping track of assignments that were tried previously and failed. The human information processing system is subject to error, however, hence requires back-up capabilities not predictable from the demands of the task environment.

Hence, from our knowledge of properties of this task environment, we can predict that an error-free serial information processing system using the heuristic we have described could solve the DONALD+GERALD problem rather rapidly, and without using much short-term memory along the way. But if it solved the problem by this method, it would have to make the assignments in the particular order we have indicated.

The empirical fact that human solvers do make the assignments in roughly this same order provides us with one important piece of evidence (we can obtain many others by analysing their thinking-aloud protocols and eye movements) that they are operating as serial systems with limited short-term memories. But the empirical data show that there are few task-independent invariants of the human processor beyond the basic structural features we have mentioned. Since the problem solver's behavior is adaptive, we learn from his protocol the shape of the task environment of DONALD+GERALD--the logical interdependencies that hold among the several parts of that problem. We also learn from

the protocol the structure of the problem space that the subject uses to represent the task environment, and the program he uses to search the problem space. Though the problem space and program are not task-invariant, they constitute the adaptive interface between the invariant features of the processor and the shape of the environment, and can be understood by considering the functional requirements that such an interface must satisfy.

Problem Spaces

Subjects faced with problem-solving tasks represent the problem environment in internal memory as a space of possible situations to be searched in order to find that situation which corresponds to the solution. We must distinguish, therefore, between the task environment--the omniscient observer's way of describing the actual problem "out there"--and the problem space--the way a particular subject represents the task in order to work on it.

Each node in a problem space may be thought of as a possible state of knowledge to which the problem solver may attain. A state of knowledge is simply what the problem solver knows about the problem at a particular moment of time--knows in the sense that the information is available to him and can be retrieved in a fraction of a second. After the first step of the DONALD+GERALD problem, for example, the subject knows not only that D=5, but also that T=0 and that the carry into the second column from the right is 1. The problem solver's search for a solution is an Odyssey through the problem space, from one knowledge state to another, until his current knowledge state includes the problem solution--that is, until he knows the answer.

Problem spaces, even for relatively "simple" problem solving tasks, are enormous. Since there are 9!=362,880 possible assignments of nine digits to nine letters, we may consider the DONALD+GERALD space to be 9! in size, which is also the size of the space of tic-tac-toe. The sizes of problem spaces for games like chess or checkers are measured by very large powers of ten--10^{120}, perhaps, in the case of chess. The space of the problem called "life" is, of course, immensely larger.

For a serial information processing system, however, the exact size of a problem space is not important, provided the space is very large. A serial processor can visit only a modest number of knowledge states (approximately ten per minute, the thinking-aloud data indicate) in its search for a problem situation. If the problem space has even a few thousand states, it might as well be infinite--only highly selective search will solve problems in it.

Many of you have tried to solve the Tower of Hanoi problem. (This is very different from the problem of Hanoi in your morning newspaper, but fortunately much less complex.) There are three spindles, on one of which is a pyramid of wooden discs. The discs are to be moved, one by one, from this spindle, and all placed, in the end, on one of the other spindles, with the constraint that a disc may never be placed on another that is smaller than it is. If there are four discs, the problem space contains only 3^4=81 possible arrangements of discs on spindles, yet the problem is non-trivial for human adults. The five-disc problem, though it admits only 243 arrangements, is very difficult for most people; and the problems with more than five discs almost unsolvable--until the right heuristic is discovered!

Problems like this one--where the problem space is not immense--tell us how little trial-and-error search the human problem solver is capable of, or is willing to endure. Problems with immense spaces inform us that the amount of search required to find solutions, making use of available structure, bears little or no relation to the size of the entire space. To a major extent, the power of heuristics resides in their capability for examining small, promising regions of the entire space and simply ignoring the rest. We need not be concerned with how large the haystack is, if we can identify a small part of it in which we are quite sure to find a needle.

Thus, to understand the behavior of a serial problem solver, we must turn to the structure of problem spaces and see just how information is imbedded in such spaces that can be extracted by heuristic processes and used to guide search to a problem solution.

Sources of Information in Problem Spaces

Problem spaces differ not only in size--a difference we have seen to be usually irrelevant to problem difficulty--but also in the kinds of structure they possess. Structure is simply the antithesis or randomness, providing redundancy that can be used to predict the properties of parts of the space not yet visited from the properties of those already searched. This predictability becomes the basis for searching selectively rather than randomly.

The security of combination safes rests on the proposition that there is no way, short of exhaustive search, to find any particular point in a fully random space. (Of course, skilled safecrackers know that

complete randomness is not always achieved in the construction of realworld safes, but that is another matter.)

Non-randomness is information, and information can be exploited to search a problem space in promising directions and to avoid the less promising. A little information goes a long way to keep within bounds the amount of search required, on average, to find solutions.

Hill-Climbing. The simplest example of information that can be used to solve problems without exhaustive search is the progress test--the test that shows that one is "getting warmer. In climbing a (not too precipitous) hill,a good heuristic rule is always to go upward. If a particular spot is higher, reaching it probably represents progress toward the top. The time it takes to reach the top will depend on the height of the hill and its steepness, but not on its circumference or area--not on the size of the total problem space.

Types of Information. There is no great mystery in the nature of the information that is available in many typical problem spaces; and we now know pretty well how humans extract that information and use it to search selectively. For example, in the DONALD+GERALD problem, we saw how information was obtained by arithmetic and algebraic operations. Now, abstracting from particular examples, can we characterize the structure of problem spaces in more general terms?

Each knowledge state is a node in the problem space. Having reached a particular node, the problem solver can choose an operator from among a set of operators available to him, and can apply it to

reach a new node. Alternatively, the problem solver can abandon the node he has just reached, select another node from among those previously visited, and proceed from that node. Thus, he must make two kinds of choices: choice of a node from which to proceed, and choice of an operator to apply at that node.

We can think of information as consisting of one or more evaluations (not necessarily numerical, of course) that can be assigned to a node or an operator. One kind of evaluation may rank nodes with respect to their promise as starting points for further search. Another kind of evaluation may rank the operators at a particular node with respect to <u>their</u> promise as means for continuing from that node. The problem solving studies have disclosed examples of both kinds of evaluations: for node and operator selection, respectively.

When we examine how evaluations are made--what information they draw upon--we again discover several varieties. An evaluation may depend only on properties of a single node. Thus, in theorem-proving tasks, subjects frequently decline to proceed from their current node because "the expression is too complicated to work with." This is a judgment that the node is not a promising one. Similarly, we find frequent statements in the protocols to the effect that "it looks like Rule 7 would apply here."

In most problem spaces, the choice of an efficient next step cannot be made by absolute evaluation of the sorts just illustrated, but instead is a function of the problem that is being solved. In theorem-proving, for example, what to do next depends on what theorem

is to be proved. Hence, an important technique for extracting information to be used in evaluators (of either kind) is to compare the current node with characteristics of the desired state of affairs and to extract differences from the comparison. These differences serve as evaluators of the node (progress tests) and as criteria for selecting an operator (operator relevant to the differences). Reaching a node that differs less from the goal state than nodes visited previously is progress; and selecting an operator that is relevant to a particular difference between current node and goal is a technique for (possibly) reducing that difference.

The particular heuristic search system that finds differences between current and desired situations, finds an operator relevant to each difference, and applies the operator to reduce the difference is usually called means-ends analysis. Its common occurrence in human problem solving behavior has been observed and discussed frequently since Duncker (1945). Our own data analyses reveal means-ends analysis to be a prominent form of heuristic organization in some tasks--proving theorems, for example. The procedure is captured in the General Problem Solver (GPS) program which has now been described several times in the psychological literature.[1/] The GPS find-and-reduce-difference

---

1/ Brief descriptions of GPS can be found in E. R. Hilgard and G. H. Bower, Theories of Learning (3rd edition), New York: Appleton-Century-Crofts, 1966, and E. R. Hilgard and R. C. Atkinson, Introduction to Psychology, (4th edition), New York: Harcourt Brace and World, 1967. For an extensive analysis of GPS, see Ernst and Newell, 1969. The relation of GPS to human behavior is discussed in Newell and Simon, 1971, Chapter 9.

heuristic played the central role in our theory of problem solving for a decade beginning with its discovery in 1957, but more extensive data from a wider range of tasks has now shown it to be a special case of the more general information-extracting processes we are describing here.

Search Strategies. Information obtained by finding differences between already-attained nodes and the goal can be used for both kinds of choices the problem solver must make--the choice of node to proceed from, and the choice of operator to apply. Examining how this information can be used to organize search has led to an explanation of an important phenomenon observed by de Groot (1965) in his studies of choice in chess. de Groot found that the tree of move sequences explored by players did not originate as a bushy growth, but was generated, instead, as a bundle of spindly explorations, each of them very little branched. After each branch had been explored to a position that could be evaluated, the player returned to the base position to pick up a new branch for exploration. de Groot dubbed this particular kind of exploration, which was universal among the chessplayers he studied, "progressive deepening."

The progressive deepening strategy is not imposed on the player by the structure of the chess task environment. Indeed, one can show that a different organization would permit more efficient search. This alternative method is called the scan-and-search strategy, and works somewhat as follows: Search proceeds by alternation of two phases: (1) in the first phase, the node that is most promising (by some

evaluation) is selected for continuation; (2) in the second phase, a few continuations are pursued from that node a short distance forward, and the new nodes thus generated are evaluated and placed on a list for phase 1. The scan-search organization avoides stereotypy. If search has been pursued in a particular direction because it has gone well, the direction is reviewed repeatedly against other possibilities, in case its promise begins to wane.

A powerful computer program for finding checkmating combinations, called MATER, constructed with the help of the scan-search strategy, appears a good deal more efficient than the progressive deepening strategy.[1/] Nevertheless, in chess and the other task environments we have studied, humans do not use the scan-search procedure to organize their efforts. In those problems where information about the current node is preserved for them in an external memory, they tend to proceed almost always from the current knowledge state, and back up to an earlier node only when they find themselves in serious trouble.[2/] In task environments where information about the current node is not preserved externally (e.g., the chessboard under rules of touch-move), and especially if actions are not reversible, they tend to preserve information about a base node to which they return when evaluation rejects the current node. This is essentially the progressive deepening strategy.

We can see now that the progressive deepening strategy is a response to limits of short-term memory, hence provides additional evidence for the validity of our description of the human information processing

---

1/ Baylor and Simon, 1966.
2/ Newell and Simon, 1971, Chapters 12 and 13.

system. When we write a problem solving program without concern for human limitations, we can allow it as much memory of nodes on the search tree as necessary--hence we can use a scan-search strategy. To the human problem solver, with his limited short-term memory, this strategy is simply not available. To use it, he would have to consume large amounts of time storing in his long-term memory information about the nodes he had visited.

That, in sum, is what human heuristic search in a problem space amounts to. A serial information processor with limited short-term memory uses the information extractable from the structure of the space to evaluate the nodes it reaches and the operators that might be applied at those nodes. Most often, the evaluation involves finding differences between characteristics of the current node and those of the desired node (the goal). The evaluations are used to select a node and an operator for the next step of the search. Operators are usually applied to the current node, but if progress is not being made, the solver may return to a prior node that has been retained in memory--the limits of the choice of prior node being set mostly by short-term memory limits. These properties have been shown to account for most of the human problem solving behaviors that have been observed in the three task environments that have been studied intensively: chess playing, discovering proofs in logic, and cryptarithmetic; and programs have been written to implement problem solving systems with these same properties.

Alternative Problem Spaces

Critics of the problem solving theory we have sketched above complain that it explains too little. It has been tested in detail against behavior in only three task environments--and these all involving highly structured symbolic tasks.[1/] More serious, it explains behavior only after the problem space has been postulated--it does not show how the problem solver constructs his problem space in a given task environment. Why, when he is faced with a cryptarithmetic problem, does he enter a problem space in which the nodes are defined as different possible assignments of letters to numbers? How does he become aware of the relevance of arithmetic operations for solving the problem? What suggests the "most-constrained-column-first" heuristic to him?

Although we have been careful to distinguish between the task environment and the problem space, we have not emphasized how radical can be the differences among alternative problem spaces for representing the same problem. Consider the following example: An elimination tournament, with 109 entries, has been organized by the local tennis club. Players are paired, the losers eliminated, and the survivors re-paired until a single player emerges victorious. How should the pairings be arranged to minimize the total number of individual matches that will have to be played? An obvious representation is the space of all possible "trees" of matchings of 109 players--an entirely infeasible space to search. Consider an alternative space in which each node is a possible

---

1/ The empirical findings, only some of which have been published to date, are collected in Parts II, III, and IV, of Newell and Simon, 1971.

sequence of matches constituting the tournament. This is, again, an enormous space, but there is a very simple way to solve the problem without searching it. Take an arbitrary sequence in the space, and note the number of surviving players after each match. Since the tournament begins with 109 players, and since each match eliminates one player, there must be exactly 108 matches to eliminate all but one player--no matter which sequence we have chosen. Hence, the minimum number of matches is 108, and any tree we select will contain exactly this number.

There are many "trick" problems of this kind where selection of the correct problem space permits the problem to be solved without any search whatsoever. In the more usual case, matters are not so extreme, but judicious selection of the problem space makes available information that reduces search by orders of magnitude in comparison with what is required if a less sophisticated space is used.

We cannot claim to have more than fragmentary and conjectural answers to the questions of representation. The initial question we asked in our research was: "What processes do people use to solve problems?" The answer we have proposed is: "They carry out selective search in a problem space that incorporates some of the structural information of the task environment." Our answer now leads to the new question: "How do people generate a problem space when confronted with a new task?" Thus our research, like all scientific efforts, has answered some questions at the cost of generating some new ones.

By way of parenthesis, however, we should like to refute one argument that seems to us exaggerated. It is sometimes alleged that search in a well-defined problem space is not problem solving at all--that the real problem solving is over as soon as the problem space has been selected. This proposition is easily tested and shown false. Pick a task environment and a particular task from it. To do the task, a person will first have to construct a problem space, then search for a solution in that space. Now give him a second task from the same environment. Since he can work in the problem space he already has available, all he needs to do this time is to search for a solution. Hence, the second task--if we are to accept the argument--is no problem at all. Observation of subjects' behavior over a sequence of chess problems, cryptarithmetic puzzles, or theorem-finding problems shows the argument to be empirically false. For the subjects do not find that all the problems become trivial as soon as they have solved the first one. On the contrary, the set of human behaviors we call "problem solving" encompasses both the activities required to construct a problem space in the face of a new task environment, and the activities required to solve a particular problem in some problem space, new or old.

Where Is the Theory Going?

Only the narrow seam of the present divides past from future. The theory of problem solving in 1970--and especially the part of it that is empirically validated--is primarily a theory that describes the problem spaces and problem-solving programs, and shows how these

adapt the information processing system to its task environment. At the same time that it has answered some basic questions about problem solving processes, the research has raised new ones: how do problem solvers generate problem spaces; what is the neurological substrate for the serial, limited-memory information processor; how can our knowledge of problem solving processes be used to improve human problem solving and learning? In the remaining pages of this paper, we should like to leave past and present and look briefly--using Milton's words--into "the never-ending flight of future days."

Constructing Problem Spaces

We can use our considerable knowledge about the problem spaces subjects use to solve problems in particular task environments as our taking-off place for exploring how the problem spaces come into being, how the subjects construct them.

Information for Construction. There are at least six sources of information that can be used to help construct a problem space in the face of a task environment:

1. The task instructions themselves, which describe the elements of the environment more or less completely, and which may also provide some external memory--say, in the form of a chessboard.

2. Previous experience with the same task or a nearly identical one. (A problem space available from past experience may simply be evoked by mention of the task.)

3. Previous experience with analogous tasks, or with components of the whole task.

4. Programs stored in long-term memory that generalize over a range of tasks.

5. Programs stored in long-term memory for combining task instructions with other information in memory to construct new problem spaces and problem solving programs.

6. Information accumulated while solving a problem, which may suggest changing the problem space. (In particular it may suggest moving to a more abstract and simplified planning space.)

The experience in the laboratory with subjects confronting a new task, and forced, thereby, to generate within a few minutes a problem space for tackling the task, suggests that the first source--task instructions and illustrative examples accompanying them, play a central role in generation of the problem space. The array presented with the cryptarithmetic problem, for example, suggests immediately the form of the knowledge state (or at least the main part of it); namely, that it consists of the same array modified by the substitution in it of one or more digits for letters.

The second source--previous experience with the same task--is not evident, of course, in the behavior of naive subjects, but the third source--analogous and component tasks--plays an important role in cryptarithmetic. Again, the form of the external array in this task is sufficient to evoke in most subjects the possible relevance of arithmetic processes and arithmetic properties (odd, even, and so on).

The fourth source--general-purpose programs in long-term memory--is a bit more elusive. But, as we have already noted, subjects quite frequently use means-ends programs in their problem solving endeavors,

and certainly bring these programs to the task from previous experience. We have already mentioned the General Problem Solver, which demonstrates how this generality can be achieved by factoring the specific descriptions of individual tasks from the task-independent means-ends analysis processes.

The fifth and sixth sources on the list above are mentioned because common sense tells us that they must sometimes play a role in the generation of problem spaces. We have no direct evidence for their use.

What evidence we have for the various kinds of information that are drawn upon in constructing problem spaces is derived largely from comparing the problem spaces that subjects are observably working in with the information they are known to have access to. No one has, as yet, really observed the process of generation of the space--a research task that deserves high priority on the agenda.

Some Simulation Programs. Some progress has been made, however, in specifying for computers several programs that might be regarded as candidate theories as to how it is done by humans. Two of these programs were constructed, by Tom Williams and Donald Williams, respectively, in the course of their doctoral research.[1/] A General Game Playing Program, designed by Tom Williams, when given the instructions for a card or board game (somewhat as these are written in Hoyle, but with the language simplified and smoothed), is able, by interpreting these instructions, to play the game--at least legally if not well. GGPP relies primarily on the first, fourth, and fifth sources of information from the list above.

1/ T. Williams, 1965; D. Williams, 1969.

It has stored in memory general information about such objects as "cards," "hands," "boards," "moves," and is capable of combining this general information with information derived from the specific instructions of the game.

The Aptitude Test Taker, designed by Donald Williams, derives its information from worked-out examples of items on various kinds of aptitude tests (letter series, letter analogies, number series and analogies, and so on) in order to construct programs capable of taking the corresponding tests.

These programs put us into somewhat the same position with respect to the generation of problem spaces that LT did with respect to problem solving in a defined problem space: that is to say, they demonstrate that certain sets of information processing mechanisms are sufficient to do the job over some range of interesting tasks. They do not prove that humans do the same job in the same way, using essentially the same processes, or that these processes would suffice for all tasks. It should be noted that the programs written by the two Williamses are erected on the same kind of basic information processing system that was used for earlier cognitive simulations. They do not call for any new magic to be put in the hat.

Planning and Abstracting Processes. The processes for generating problem spaces are not unrelated to some other processes about which we do have empirical data--planning processes. In several of the tasks that have been studied, and especially in the logic task, subjects are often observed to be working in terms more abstract than those that

characterize the problem space they began with. They neglect certain details of the expressions they are manipulating (e.g., the operators or connectives), and focus upon features they regard as essential.

One way of describing what they are doing is to say that they are abstracting from the concrete detail of the initial problem space in order to construct a plan for a problem solution in a simpler abstract planning space. Programs have been written, in the context of GPS, that are also capable of such abstracting and planning, hence are capable of constructing a problem space different from the one in which the problem solving begins.

The evidence from the thinking-aloud protocols in the logic task suggests, however, that the human planning activities did not maintain as sharp a boundary between task space and abstract planning space as the simulation program did. The human subjects appeared able to move back and forth between concrete and abstract objects without treating the latter as belonging to a separate problem space. In spite of this difference, the data on planning behavior gives us additional clues as to how problem spaces can be generated and modified.

Production Systems

A hypothesis about the structure of a complex system--like a human problem solving program--becomes more plausible if we can conceive how a step-by-step development could have brought about the finished structure. Minerva sprang full-grown from the brow of Zeus, but we expect terrestrial systems to evolve in a more gradual and lawful fashion--our distrust of the magician again.

Anyone who has written and debugged a large computer program has probably acquired, in the process, a healthy skepticism that such an entangled, interconnected structure could have evolved by small, self-adapting steps. In an evolving system, a limited, partial capability should grow almost continuously into a more powerful capability. But most computer programs have an all-or-none character; disable one subroutine and a program will probably do nothing useful at all.

A development of the past few years in computer language construction has created an interesting possible solution to this difficulty. We refer to the languages known as production systems. In a production system, each routine has a bipartite form, consisting of a condition and an action. The condition defines some test or set of tests to be performed on the knowledge state. (E.g., "Test if it is Black's move.") If the test is satisfied, the action is executed; if the test is not satisfied, no action is taken and control is transferred to some other production. In a pure production system, the individual productions are simply listed in some order, and considered for execution in turn.

The attraction of a production system for our present concerns--of how a complex program could develop step by step--is that the individual productions are independent of each other's structures, and hence productions can be added to the system one by one. In a new task environment, a subject learns to notice conditions and make discriminations of which he was previously unaware (a chessplayer learns to recognize an open file, a passed pawn, and so on). Each of these new discriminations can become the condition part of a production, whose action is relevant to that condition.

We cannot pursue this idea here beyond noting its affinity to some classical stimulus-response notions. We do not wish to push the analogy too far, for productions have some complexities and subtleties of structure that go beyond stimulus-response ideas, but we do observe that linking a condition and action together with its response. One important difference is that, in the production, it is the condition--i.e., the tests--and not the stimulus itself that is linked to the response. In this way, the production system illuminates the problem of defining the effective stimulus, an old bugaboo of S-R theory.

Perception and Language

We have seen that research on problem solving has begun to shift from asking how search is conducted in a problem space--a subject on which we have gained a considerable understanding--to asking how internal representations of problems are built up in human minds. But the subject of internal representation links problem solving research with two other important areas of psychology: perception and psycholinguistics. The further extension of this linkage (see Step 8 in the strategy outlined in our introductory section) appears to be one of the principal tasks for the next decade.

In another place one of us has described briefly the main connections between problem solving theory and the theories of perception and psycholinguistics.[1/] We will simply indicate these connections even more briefly here.

1/ Simon, 1969, pp. 42-52.

Information comes to the human problem solver principally in the form of statements in natural language and visual displays. For information to be exchanged between these external sources and the mind, it must be encoded and decoded. The information as represented externally must be transformed to match the representations in which it is held inside. It is very difficult to imagine what these transformations might be as long as we have access only to the external representations, and not to the internal. It is a little like building a program to translate from English to language X, where no one will tell us anything about language X.

The research on problem solving has given us some strong hypotheses about the nature of the internal representations that humans use when they are solving problems. These hypotheses define for us, therefore, the endpoint of the translation process--they tell us something about language X. The hypotheses should provide strong clues to the researcher in perception and to the psycholinguist in guiding their search for the translation process. Indeed, we believe that these cues have already been used to good advantage in both areas, and we anticipate a great burgeoning of research along these lines over the coming decade.

Links to Neurophysiology

The ninth step in the strategy set forth in our introduction was to seek the neurophysiological counterparts of the information processes and data structures that the theory postulates. In this respect, we are in the position of nineteenth century chemistry which postulated

atoms on the basis of observations of chemical reactions among molecules, and without any direct evidence for their existence; or in the position of classical genetics, which postulated the gene before it could be identified with any observed microscopic structures in the cell.

Explanation in psychology will not rest indefinitely at the information processing level. But the explanations that we can provide at that level will narrow the search of the neurophysiologist, for they will tell him a great deal about the properties of the structures and processes he is seeking. They will put him on the lookout for memory fixation processes with times of the order of five seconds, for the "bottlenecks" of attention that account for the serial nature of the processing, for memory structures of small capacity capable of storing a few symbols in a matter of a couple of hundred milliseconds.

All of this is a prospect for the future. We cannot claim to see in today's literature any firm bridges between the components of the central nervous system as it is described by neurophysiologists and the components of the information processing system we have been discussing here. But bridges there must be, and we need not pause in expanding and improving our knowledge at the information processing level while we wait for them to be built.

The Practice of Education

The professions always live in an uneasy relation with the basic sciences that should nourish and be nourished by them. It is really only within the present century that medicine can be said to

References

Simon, H. A., & Kotovsky, K. Human acquisition of concepts for sequential patterns. Psychol. Rev., 1963, 70, 534-546.

Williams, D. S. Computer program organization induced by problem examples. Unpublished doctoral dissertation, Carnegie-Mellon Univ., 1969.

Williams, T. G. Some studies in game playing with a digital computer. Unpublished doctoral dissertation, Carnegie Institute of Technology, 1965.