# PROBLEM-SOLVING ABILITY

*Micheline T. H. Chi*
*Robert Glaser*
*University of Pittsburgh*

Solving problems is a complex cognitive skill that characterizes one of the most intelligent human activities. From childhood onward, we actively solve problems presented to us by the world. We acquire information about the world, and organize this information into structures of knowledge about objects, events, people, and ourselves that are stored in our memories. These structures of knowledge comprise bodies of understanding, mental models, convictions, and beliefs that influence how we relate our experiences together, and how we solve the problems that confront us in everyday life, in school, in our jobs, and at play.

This research program, conducted at the Learning Research and Development Center, University of Pittsburgh, is supported in part by Contract no. N00014–79–C–0215, NR 157–430/12–19–80, of the Office of Naval Research, Personnel and Training Research Programs, Psychological Sciences Division, and in part by the National Institute of Education. The authors are grateful to Ted Rees for comments and editing.

How do humans develop their abilities to solve problems in these situations? People differ, children from adults, and experts from novices, and these differences are based on cognitive processes and mental organizations that humans have in common, and that characterize their problem-solving abilities. In this chapter we emphasize the general characteristics of human problem-solving ability. Systematic theory on the mechanisms of human problem solving is a relatively recent advance in psychological science, and knowledge of fundamental processes provides a basis for understanding the development and acquisition of our abilities to think and solve problems.

Scientists interested in problem solving have studied various classes of problems, each having its own task characteristics, which determine to a large extent the behavior of the problem solver and the strategies appropriate to finding solutions. In addition, as we have seen in the preceding chapters, the experience and knowledge brought to the situation by the problem solver determine whether and how a solution will be reached. Puzzle problems, like tic-tac-toe or combining the links of several small chains into a larger chain with the minimal number of moves, require little knowledge of a subject-matter domain. Solving problems in elementary physics, however, requires more subject-matter knowledge, such as knowledge of force diagrams and of certain laws of physics. Puzzle problems have been studied largely because they are not complicated by needing much background knowledge, and because they reveal the strategies that people employ in searching for a solution. Detailed observations of performance in puzzle situations have been accompanied by computer simulations of these performances that precisely describe certain general strategies or *heuristics* that people use when they are confronted with novel situations. As a result of extensive work with these computer models of problem solving, the main mechanisms for solving these well-structured puzzle problems are fairly well understood. The strategies used depend on attention to perceptual cues, the goals and subgoals held in memory, and the discovery of sequential patterns of correct moves.

Research has also been carried out on the nature of expert problem solving in knowledge-rich domains like chess playing and school problems in physics and mathematics. Investigation of the performance of experts and novices in domains requiring extensive knowledge has deepened our understanding of the kinds of knowledge required for efficient problem-solving ability. In particular, the investigation of problem solving in domains requiring extensive knowledge has shown how the knowledge organizations acquired by the problem solver, which are stored in long-term memory, influence the perceptual processes and strategies of problem solving.

Two important factors, then, that influence problem solving are the nature of the task (the *task environment*) and the kind of knowledge brought to the problem by the solver. These two factors dictate the organization of this chapter. In the first main section, we will consider puzzle problems and general processes of solution. In the second, we will discuss solving of problems that require domain knowledge. We will also consider various task environments that involve insight, creativity, and ill-structured problems. Our goal, in the problem-solving task of writing this chapter, is to give you an understanding of some of the mechanisms that underlie problem-solving abilities. We can now begin by defining the various types of problems that have been studied.

## What Is a Problem?

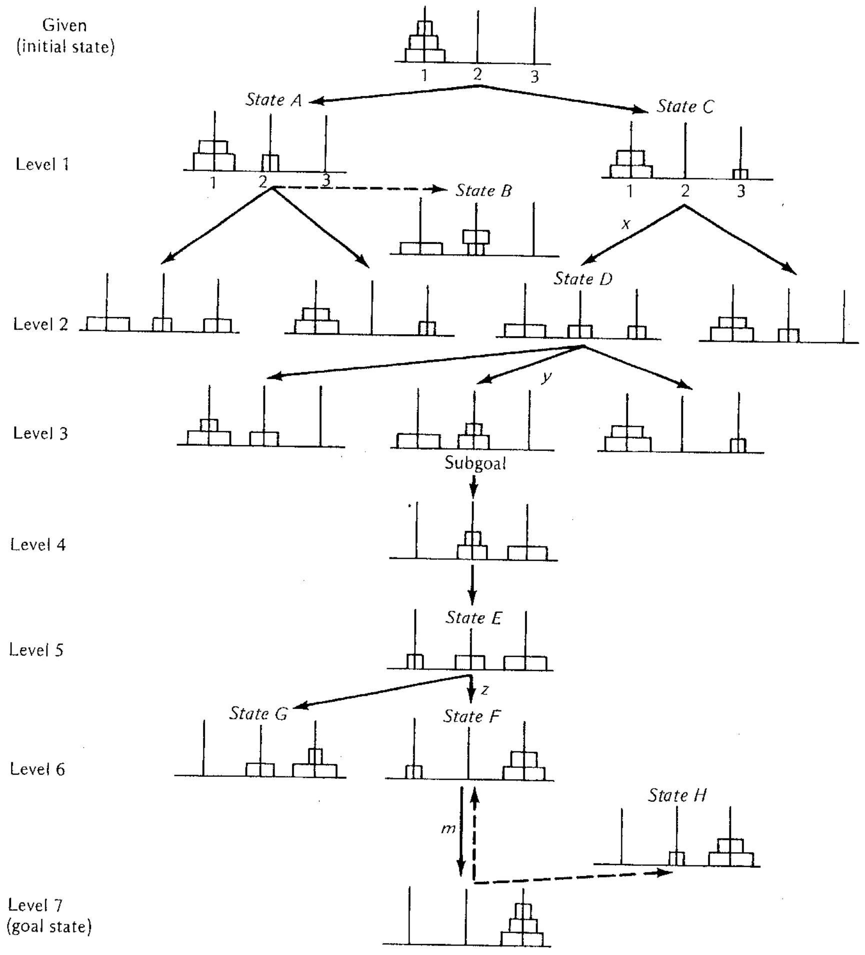
A problem is a situation in which you are trying to reach some goal, and must find a means for getting there. Figuring out puzzles, solving algebra problems, deciding how to budget a limited amount of money, trying to control inflation and reduce unemployment, are all examples of problems that we as individuals and as a society frequently encounter. Clearly, these problems cover an enormous range of difficulty and complexity, but they do have some things in common. They all have some *initial state,* whether it is a set of equations or the state of the economy, and they all have some *goal.* To solve the problem, you must perform some *operations* on the initial state to achieve that goal. Often there are some rules that specify allowable operations, and these are generally called *constraints.*

Puzzles are one form of problem with which we are all familiar. The Tower of Hanoi puzzle (shown in Figure 10.1) is a typical example, and has received a great deal of attention from psychologists. It consists of three pegs and a set of disks of different sizes. In the initial state, the disks are stacked up on the first peg (the top display in Figure 10.1) in order of decreasing size (like a pyramid), and the goal is to move all the disks onto the third peg, maintaining the pyramid (the bottom display in Figure 10.1). The constraints in this problem are that the solver can never place a larger disk on top of a smaller one, and that only one disk may be moved at a time.

Although puzzles are interesting, the kinds of problems that we most frequently encounter fall into two types: classroom problems and real-world problems. An algebra problem where we have to find the unknown quantity typifies our standard notion of a problem. Students spend a good deal of time learning to solve various kinds of classroom problems in such areas as mathematics, physics, and chemistry. A distinct difference between puzzles and classroom problems is that a fair amount of knowledge of a specific subject area is necessary for the solution of the classroom problems. Although even a young child can move a disk from one peg to another, solving an algebra problem requires knowing when and how to apply a whole set of rules for manipulating equations. An example of a fairly complex algebra problem can be found in Box 10.1.

Often the most important and difficult problems we have to solve are those that we encounter in everyday life outside the classroom. Such problems range from finding our way about in a city, to elevating ourselves out of a period of depression. In both cases, we have a clear initial state (being lost at location *A* or feeling depressed), and a clear goal (wanting to be at location *B* or feeling elated). The solution for the first problem might be either calling a taxi, asking directions, or reading a map.

**Figure 10.1**
The Tower of Hanoi.



The puzzles, classroom, and real-world problems described so far are all *well defined.* That is, you can readily recognize when the problem is solved. We know that the Tower of Hanoi problem is solved when the disks have all been transferred to the third peg. An algebra problem is solved when a quantity for the unknown is found (although it could be the wrong answer). Likewise, we know when we feel less depressed, and we know that we have accomplished our goal of finding our way when we are at location *B.*

There is another class of problems, called *ill defined.* These are generally problems in which one or several aspects of the situation is not well specified. Examples are composing a poem, or designing a house. The general nature of

**Box 10.1**
The Smalltown problem

Because of their quiet ways, the inhabitants of Smalltown were especially upset by the terrible New Year's Eve auto accident which claimed the life of one Smalltown resident. The facts were these. Both Smith and Jones were New Year's Eve babies, and each had planned a surprise visit to the other one on their mutual birthday. Jones had started out for Smith's house traveling due east on Route 210 just two minutes after Smith had left for Jones' house. Smith was traveling directly south on Route 140. Jones was traveling 30 miles per hour faster than Smith even though their houses were only five miles apart as the crow flies. Their cars crashed at the right-angle intersection of the two highways. Officer Franklin, who observed the crash, determined that Jones was traveling half again as fast as Smith at the time of the crash. The crash occurred nearer to the house of the dead man than to the house of the survivor. What was the name of the dead man?

these problems is that their descriptions are not clear, and the information needed to solve them is not entirely contained in the problem statement; consequently, it is even less obvious (than in well-defined problems) what actions to take in order to solve them.

## Puzzle Problems and Processes of Solution

Well-structured puzzle problems have been traditionally studied by psychologists and have received particular attention in the last two decades. Their popularity arose from two related considerations.

First, with the advent of computer science and its accompanying notions and techniques of computer simulation of human behavior (Newell et al., 1958), psychologists were particularly interested in rigorous investigation of the basic processes of solving problems. The technique used was to have a subject "think out loud" while solving a given problem. This problem-solving protocol is tape-recorded, transcribed, and then intensely analyzed to find out just how the subject tried to solve the problem. The problem-solving strategies used by the subject are then simulated in a computer program, to see if the program can produce similar solution patterns.

Second, puzzle problems require very little background knowledge and yet can be very difficult to solve. In such tasks, differences in individuals' abilities to solve problems can be attributed to differences in some basic underlying problem-solving processes rather than to greater or lesser subject-matter knowledge. Puzzle problems therefore lend themselves particularly well to uncovering the underlying solution processes, by tracing the sequence of operations applied to transform the initial state to the goal state.

The focus of modern cognitive psychologists on processes is in contrast with the earlier problem-solving research by Gestalt psychologists of the 1930s. The early work dealt mostly with *insight* problems, and emphasized how appropriate changes in the representation of a problem could lead to solution. A typical insight problem is the two-string problem (Maier, 1931). Two strings are hung from the ceiling. The object is to tie them together, but they are too far apart for a person to reach both of them at the same time. A book of matches, a few pieces of cotton, and a screwdriver are on the table nearby. The necessary insight for solving this problem comes from a person recognizing that the screwdriver can be used in other than its usual function. Here it may be tied to one string, to create a pendulum that can be swung to the other string. This early research focused on the conditions that impeded or facilitated problem solving, and on how intuitively improbable responses could become more probable. For example, would putting the screwdriver among a different set of items on the table make it easier for the solver to recognize that a screwdriver can be used in an unfamiliar way, as a weight?

The information-processing approach, represented by the work of Newell and Simon (1972), significantly changed problem-solving research, by turning attention away from the conditions under which solutions can be reached, and toward the component cognitive processes involved that transform the initial state of the problem to the final goal state. In order to make this problem a manageable one to study, many researchers in the late 1960s and early 1970s concentrated on a class of puzzle-like problems called *move* or *transformation* problems. These problems all have clear initial and goal states, and a small set of well-defined operations (moves) that can successively transform the initial state to the goal state.

The Tower of Hanoi is one such problem, and it has been extensively studied. A spatial metaphor is useful in analyzing this kind of problem. The states of the problem are represented as points in a space. The possible operations on each state are represented by lines leading to the states that these operations produce. Because there is only one initial state, and typically more than one operation can be applied to each state to generate several plausible transformed states, this *solution space* resembles an upside-down tree. Part of the solution space for the Tower of Hanoi is shown in Figure 10.1.

One way to look at the process of solving a problem is to think of it as a search through the solution space. The space contains many possible paths, but only one (or a few) leads to the goal state. The distance traveled down a particular branch of the tree, the levels in Figure 10.1, is often referred to as the depth ($D$) of search, and the number of alternatives at each point is the breadth ($B$). The total number of possible paths is equal to $B^D$ (if $B$ is the same number at every state). That is, the number of alternative paths to be considered explodes exponentially.

## *Representation*

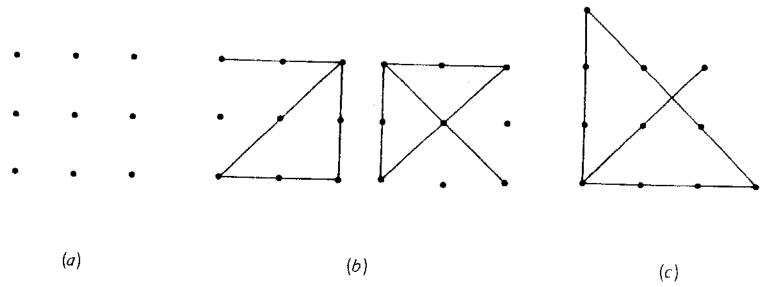
The *representation* of a problem consists essentially of the solver's interpretation or understanding of the problem. Researchers have found that the representation is very important in determining how easy a problem is to solve. In the two-string problem, for example, insight is really representation. The initial representation of the screwdriver must be broadened to include the fact that it is a heavy object, which can therefore be used as a pendulum weight.

The representation of move problems, like the Tower of Hanoi, is fairly straightforward, because the initial and final states (and the permissible moves) are clearly spelled out. However, we can conceive of situations where the representation fails to embody one or more aspects of the problem. For example, if the constraints on the operations are not properly encoded, then the solution space could be enlarged unnecessarily. For example, when children try to solve the Tower of Hanoi problem, they may have trouble remembering the constraint that a larger disk may not be placed on a smaller disk, or they may forget what the goal state should look like (Klahr and Robinson, 1981). If the children forget a constraint, they essentially add branches to the solution space, making the correct path harder to find. The resulting search space is referred to as the solver's *problem space*. The dotted path from State *A* to State *B* in Figure 10.1 shows an illegal addition to the solution space.

In certain problems, a solver might also add unnecessary constraints, thereby taking away branches, so that the proper solution path is not even present in the problem space. The problem thus becomes impossible to solve. Adding or deleting branches because of improper representation is common. One example of adding a constraint to the representation is to be seen in the problem of drawing four straight lines through nine dots arranged as three rows of three without retracing or lifting the pencil (see Figure 10.2a). Almost invariably, solvers unconsciously add the constraint that the line cannot go outside of the square indicated by the dots, as shown in Figure 10.2b.

The representation of a problem can also be faulty if solvers encode the goal state improperly. This makes the problem impossible to solve, because solvers then do not know what to search for. An illustration of a problem in which the goal state is improperly encoded is the well-known sixteen-matches problem (see Figure 10.3a). The goal is to form four squares, given the constraint that only three matches may be moved. One of the common causes of error is that the goal is often represented as in Figure 10.3b, instead of in arrangements

**Figure 10.2**
The nine dots problem: (a) initial figure; (b) typical attempts at solution; (c) correct solution.



like those in Figure 10.3c, where each square uses up four sticks. That is, the initial representation of the goal does not emphasize that sixteen sticks can make up four squares perfectly, so that no stick should be shared by two squares. (Another way to look at this is that the representation did not contain the constraint that no stick should be shared by two squares.)

The difficulty in solving move problems is not generally related to solvers' misrepresenting the initial state, but rather to their omitting constraints or not having a clear representation of the goal state, as just described. However, other puzzle problems, such as the two-string problem, are difficult precisely because of initial-state representation. For problems of greater complexity, the initial representation is even more important, as we will discuss later in this chapter.

## *Searching for a Solution*

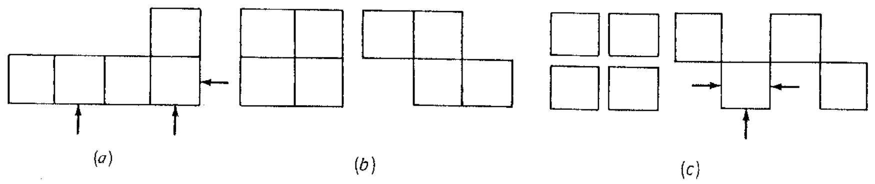
The process of finding a solution to a problem can be visualized as a search through the paths in the problem space until one that leads to the goal is found. Since move problems are generally not difficult to represent, research on solving move problems has tended to concentrate on uncovering the strategies that effective problem solvers use to find a solution in the problem space. There are a variety of strategies for carrying out this search.

One strategy is to try paths randomly, hoping to stumble on the goal. A *random search* is adequate if your search space is small. However, since the search space expands exponentially for most problems, the chance of a random search being successful is quite small.

Another obvious strategy is that of systematically searching the entire tree. In a *depth-first search,* for instance, you search a particular path all the way to the bottom. If this state is not the goal, you back up one level and then start searching down again via an untried path. When all paths from a particular state have been tried, you back up one more level and start down again, and so on. This method (and any such exhaustive method) requires much recordkeeping to keep track of which paths you have tried, and which state you should back up to when all current links have been tried. Except for very simple problems, the memory required for this recordkeeping is too great for human beings. Exhaustive methods are often not feasible even for computers to use.

The key to the effective strategies actually used by humans is to reduce the search space by considering only one branch or a very few branches at each point. For instance, de Groot (1966) found that chess players use a strategy that can be called depth-first, because they initially follow one path straight down for

**Figure 10.3**
The move three matches problem (see text for discussion).



a few moves. They make no attempt to be exhaustive or to backtrack systematically, however. Instead, they tend to jump back to the beginning position or to some important intermediate point and gradually explore just a few alternative branches. Obviously, if only a few alternatives at each point are going to be explored, they had better be good ones; so good strategies are those that guide the selection of promising moves or the elimination of unpromising ones.

*Means/ends analysis.* A powerful strategy for finding good alternatives, which takes the goal state into account, is *means/ends analysis.* This strategy was used as an important search strategy in one of the earliest attempts to create a computer program to solve problems, the General Problem Solver (Ernst and Newell, 1969). The general idea is to discover what differences there are between the current state and the goal state, and then to find operations that will reduce them. If there is more than one such operation, the one that reduces the largest difference is applied first. In other words, find the best means to achieve the desired end.

In the Tower of Hanoi, the differences are simply that disks are not on the proper peg in the proper order. At the initial state, the move that does the most to reduce the difference is placing the small disk on the third peg (see state *C* in Figure 10.1). Note, however, that this move actually introduces a new difference, because the disk is in the wrong (bottom) position on the correct (third) peg. Being in the wrong position must therefore be defined to be a smaller difference than not being on the correct peg. That is, state *C* is the correct move, because there is less of a difference between state *C* and the goal state than between state *A* and the goal.

Means/ends analysis, however, is not guaranteed to find the solution. At state *D* in Figure 10.1, for example, the correct move is to take path *y*, even though it does not reduce the overall difference. General methods like this one, which reduce the number of moves that must be considered but are not guaranteed to succeed in all situations, are called *heuristics.*

Means/ends analysis can be used to work not only from the initial state to the goal, but also backward from the goal. In this situation, the strategy is to find an operation (such as path *m* in Figure 10.1) that can produce the goal state when it is applied to a state which is closer to the initial state (e.g., state *F*). In the Tower of Hanoi example, to find state *F*, we can transform state *E* via path *z*, working forward. To work backward from the goal, we want to find the transformation (here, path *m*) that, when applied in a forward manner, will take us from state *F* to the goal. Working backward, however, two states (states *F* and *H*) can be transformed from the goal, only one of which (state *F*) overlaps with one of the two states (states *F* and *G*) generated in a forward way, from state *E*. So by combining forward and backward searches, we can select the right move almost by default.

An additional benefit of working backward (which is unfortunately not true in the Tower of Hanoi problem), is that often the number of "backward branches" to be considered is smaller than the number when working forward. Another difference between the forward and backward strategy is that we can try out moves as soon as they are decided on in the forward strategy, whereas backward strategy requires a chain of moves to be selected first. We then apply them in the same order as they would have been applied if we had been using the forward strategy. So working backward requires planning, and is therefore more sophisticated than forward strategy.

Although the Tower of Hanoi is quite a simple puzzle, means/ends analysis is useful in more-complicated situations. Newell and Simon (1972) observed subjects using it in solving *cryptarithmetic* problems (in which an arithmetic problem is presented with letters standing for numbers, and the solver must discover which letter stands for which number). Reed and Simon (1976) show how it is used on the missionaries and cannibals problem (in which one boat of limited capacity must be used to transport missionaries and cannibals across a river without anyone being eaten), and Atwood and Polson (1976) demonstrated its use in the water-jug problems. Simon and Simon (1978) noted a subject who worked backward while solving distance-rate-time problems; and Larkin et al. (1980) observed similar solution strategies being used to solve physics problems.

Means/ends analysis does have limitations, of course. In particular, it will fail if there is no operation that will reduce the remaining differences. Such an occasion will arise when a problem requires a *detour* to get to the goal. The term *detour* generally refers to a path that *increases* the differences between the current and goal states, at least temporarily. This is the situation that was mentioned in state *D* in Figure 10.1. Here the differences from the goal state are that the large and medium disks are on the wrong pegs, and the small one is in the wrong position even though it is on the correct peg. None of the available moves actually reduces the overall difference. The correct move, which is moving the smallest disk (taking path *y*), actually produces a larger difference than the previous state.

*Subgoaling.* A very useful strategy, which can be used in conjunction with means/end analysis, is *subgoaling.* Subgoaling is simply picking out an intermediate state on the solution path to reach as the temporary goal. In effect, subgoaling divides a problem into two or more subproblems, thus transforming the entire search space into two or more spaces of smaller depth.

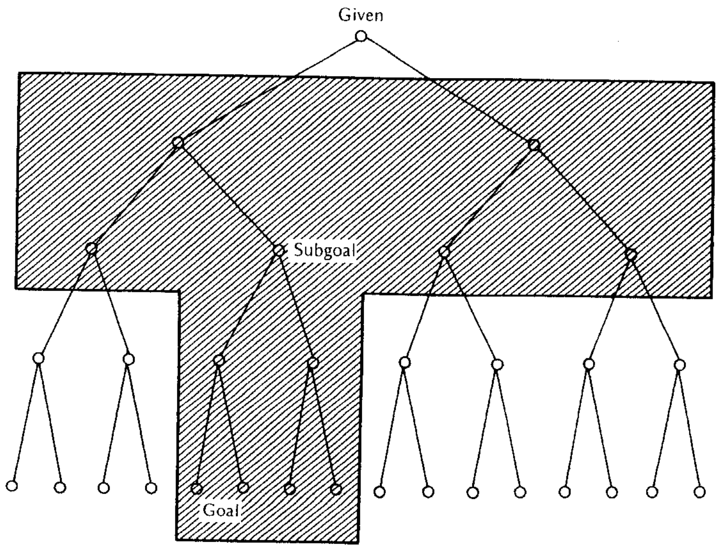
Choosing a subgoal well can allow you to use means/ends analysis in situations where, by itself, it might not reduce a difference (as when a detour is needed). In the Tower of Hanoi, one useful subgoal is for you to get the largest disk onto its proper peg, because you must put it there before you can place the others on top of it. Look at state *D* in Figure 10.1 again, with that subgoal in mind. Now the differences for you to consider are that (a) the large one is on the wrong peg and (b) the small one is blocking it. By using means/ends analysis now, you will see that the correct move (taking path *y*) does reduce the difference, whereas the less-efficient ones do not. Subgoaling can therefore remove the apparent need to move away from a goal in order to get to it. In effect, when you use subgoaling with means/ends analysis, you remove the limitation of means/ends analysis.

Subgoaling also reduces the search space significantly. Remember that the size of the space increases exponentially with its depth. If the subgoal we have selected is on the solution path, then, once we have reached it, we have reduced significantly the number of alternative paths to be considered to reach the goal, as Figure 10.4 shows. The obvious advantage to subgoaling is that we have divided a larger problem into smaller ones.

A pragmatic difficulty, though, is how to choose useful subgoals. There are several heuristics for finding them. One way is for you to work backward first, and then use the new state that you have reached as the subgoal to work forward to. Another method is to *decompose* the main goal. In the Tower of Hanoi problem, the goal is for you to have the set of disks on the third peg in a certain order, and an obvious subgoal is for you to first put the larger disk on the third peg. In order for you to achieve this subgoal, the third peg must be empty and the large disk must be uncovered. In other words, the other disks must be on the middle peg. Because large disks can never be on top of small ones, a two-disk subgoal is generated, with the middle peg as the goal peg. This form of the subgoal is very useful when there are more than three disks in the problem, because it decomposes the puzzle into a hierarchic series of smaller problems.

Another heuristic for setting subgoals is for the solver to consider only one constraint at a time, which serves systematically to narrow down the problem space. Actually, this may be the way most people solve some everyday problems, since they may not initially think of all the constraints involved. Suppose, for example, that you are buying your first house. Your initial constraint is that the price be under $50,000. After looking around at a few houses, you realize that in order to cut down on the heating bills, you need a well-insulated contemporary house rather than one of those old mansions. Thus, your next

**Figure 10.4**
Reducing the search space with a subgoal.



search will be limited only to smaller contemporary houses, but still under the initial price constraint of $50,000.

Although subgoaling seems theoretically to be an ideal strategy, research on the effect of providing a subgoal shows that it does not always help. Providing a subgoal for problem solvers can instead increase their confusion, because they do not seem to know how to continue after the subgoal has been reached. Hayes (1966), for example, found that people took longer to solve problems when a subgoal was provided in the statement of the problem, even though the subgoal helped them reach the situation quite easily, than when they were allowed to solve the problem by their own methods.

*Generate-and-test.* A heuristic that is often useful in many cases is for you to generate a set of possible solutions to a given problem directly, and then to test each one to see if it is the correct solution. Consider the task of opening a small combination lock. Possible solutions are all short series of numbers (usually three) that fall between the smallest and largest numbers on the dial. These are easy to generate. Testing is also easy: just turn the dial and see if the lock opens. Fortunately (or unfortunately, depending on your situation), the total number of possible solutions is quite large; success is certain, but testing all 999 combinations (at one per second) would take 17 minutes.

Generate-and-test is a useful technique only when it is reasonably easy for you to generate the set of potential solutions and test them. This may occur, for example, when the set is fairly small and the path from the current state to the solution is unimportant to you, or when you have difficulty generating the search space.

Particularly relevant examples of the generate-and-test strategy are scientific research and medical diagnosis. In research, an investigator will generate a hypothesis to explain some observed phenomena (the behavior of humans solving problems, for instance) and then devise experiments to test it. In medicine, physicians typically make a tentative diagnosis (or hypothesis) based on a partial description of the initial state (the patient's symptoms), and then order various examinations and laboratory tests to confirm or disconfirm the diagnosis. Even if the initial hypothesis is disconfirmed, the testing process generally adds new information about the initial state and can lead to new hypotheses. These examples lend themselves particularly well to a generate-and-test heuristic, mainly because it is obviously difficult for us to find a set of operations that can transform the initial state into the goal state. In the medical diagnosis example, the initial state consists mainly of symptoms, and the goal state is to explain what causes them. Typically, this initial state is incompletely known; some of the symptoms may be irrelevant; and there may be multiple causes at work (a patient may be suffering from heart, lung, and kidney problems, for instance). In such a situation, it is much easier to hypothesize a particular cause, find out what its effects would be, and compare them with the known symptoms.

In sum, the outcome of the research on puzzle-type problems has been very useful in uncovering several standard and powerful general-purpose problem-solving techniques: means/ends analysis, subgoaling, and generate-and-test. These heuristics have been observed to be used across a wide range of problems, including puzzles, physics problems, and medical diagnosis.

## Domain Problems and Knowledge

Although research on puzzle problems, particularly move problems, flourished well into the 1970s and provided valuable insights into the kinds of general problem-solving strategies people use, researchers found, when studying more-complicated kinds of problems, that strategies did not, by themselves, sufficiently describe problem-solving performance. Knowledge of the problem domain is also important, and can influence the use of general problem-solving heuristics. Striking demonstrations of the influence of knowledge on problem-solving processes were provided by researchers in both cognitive psychology and artificial intelligence. In cognitive psychology, de Groot (1966) and then Chase and Simon (1973a,b) explored what makes master chess players different from less-expert ones. De Groot obtained protocols from former world champions and from some fairly skillful club players as they tried to find the best move in a given situation. Surprisingly, he found no strategy differences. All players tended to consider the same number of moves. They also looked ahead about the same number of moves as they tried to evaluate each move, and they used the same strategy to guide this search. In contrast to other players, experts simply recognized the best move and gave it first consideration, evaluating the other moves only as a way of double-checking themselves. This ability to perceive the problem in a way that restricts the problem space has since been shown to occur in other areas as well. For instance, in medical diagnosis research, both experts and novices are found to use the same kind of generate-and-test heuristic. The experts, however, start with a more accurate hypothesis (Elstein et al., 1978). So what differentiates an expert problem solver from a nonexpert is not the use of different or more powerful heuristics, but an ability to choose the best path to solution without considering all the others.

Research in artificial intelligence aimed at creating chess-playing programs (e.g., Wilkins, 1980) has further demonstrated that human skill is not based on strategies to guide search through the solution space. The number of possible moves at a given position can be fairly large, perhaps 30 or 35, and because the size of a search space increases exponentially with each subsequent move on a given possible path of play, the size of the chess space is extremely large (somewhere around $10^{120}$ paths). Even the most powerful computers cannot search this large space entirely, and it must be drastically reduced by using sophisticated computational strategies to evaluate and eliminate moves. Although these programs can play at tournament level, they succeed because they can do far more searching than humans, using complex statistical strategies to compute and evaluate the best move in a way that humans do not. Even with these advantages, they still cannot match the best players. Clearly, good chess players do something very different from searching through the space for a good move.

The evidence indicates that what humans actually do is to build up an extremely large store of knowledge about typical chess positions through years of experience. De Groot (1966) provided a clue when he found that chess masters, when shown a chess position for only five seconds, were able to remember it with very high accuracy, whereas the less-expert players could not. This difference could not be attributed to the experts' superior visual memory, because when random board positions were used that were not like chess patterns, masters did as poorly as novices (Chase and Simon, 1973a). Using very simple techniques, Chase and Simon showed that the masters were actually perceiving groups of pieces, or *chunks.* They asked subjects to reproduce from memory board positions that had just been shown to them, or to copy a position from one board to another. During the reproduction task, groupings could be detected if significantly longer pauses occurred after the placement of several individual pieces in quick succession. Because the contents of a chunk are closely associated in memory, once a chunk is accessed, recall of its component pieces will be rapid. However, more time is needed to access another chunk. When masters were copying a position, clear evidence of grouping could be detected by their periodic glances at the board. The subject would turn to look at the board, place a series of pieces, then look back again, place the next series of pieces, and so on.

Significant pauses and head turns, as evidence of chunking, were observed with both expert and novice subjects. However, expert groupings contained three to six pieces, whereas novice groupings contained only one or two. It is the larger sizes of the experts' chunks that enabled them to remember so many more pieces from a chessboard than did novices. Moreover, the pieces in the expert patterns were related in identifiable ways to chess knowledge. They constituted very common patterns, such as a castled-king or a pawn chain, or local clusters of pieces. Additional support for the existence of chunks comes from eye-movement studies. When experts were asked to analyze a board position, their eye fixations showed that they were concentrating on groups of pieces, such as those in important attack and defense relationships (Simon and Barenfeld, 1969).

Differences in chunking between experts and nonexperts have been observed in other domains, as well. In electronics, Egan and Schwartz (1979) found that skilled technicians reconstructed symbolic drawings of circuit diagrams according to the functional nature of the elements in the circuit, such as amplifiers, rectifiers, and filters. Novice technicians, however, produced chunks based more on the spatial proximity of the elements. When Akin (1980) asked architects to reconstruct building plans from memory, several levels of patterns were produced. First, the architects recalled local patterns consisting of wall segments and doors, then rooms and other areas, then clusters of rooms or areas. In other words, the reconstruction processes exhibited a hierarchic pattern of chunks within chunks.

The important implication of this research is that when experts look at an apparently complicated situation, they are able to represent it in terms of a small number of patterns or chunks. If the situation is very intricate, as with the architectural drawings, their knowledge is further organized into embedded sets or hierarchic structures.

## *How Structured Knowledge Facilitates Problem Solving*

Although understanding the cognitive processes involved in finding a good chess move is clearly a difficult and challenging problem for analysis, we are especially interested in what these findings tell us about solving the kinds of problems we encounter in the classroom and in life in general. Remember that the initial representation of a problem is very important in determining how easy the problem is to solve. Just as a chess player's knowledge allows a representation of a given situation to be formed, a problem solver's knowledge somehow determines the problem representation. Then the proper problem-solving procedures, if they are known, must be retrieved from memory and applied. It is the problem solver's representation that guides retrieval of appropriate solution procedures.

In order to consider this process of representation in more detail, we will find it very helpful to use the concept of a *schema.* Recall that a schema is a theoretical construct which describes the format of an organized body of knowledge in memory. Researchers conceive of a schema as a modifiable information structure that represents generic concepts stored in memory. Schemata represent knowledge that we experience, such as the interrelations between objects, situations, events, and sequences of events that normally occur. In this sense, schemata contain prototypical information about frequently experienced situations, and they are used to interpret new situations and observations (Rumelhart, 1981). Often a great deal of relevant information is not apparent, so that you may have trouble understanding a situation without filling in the missing data by means of prior knowledge. Estes (National Academy of Sciences, 1981) explains this point by describing the following vignette:

*At the security gate, the airline passenger presented his briefcase.*
*It contained metallic objects.*
*His departure was delayed.*

In order to understand this commonplace incident, an individual must already know a good deal about air terminals. Such *prior knowledge* is represented in memory by a schema that specifies the relations between the roles played by various people in the terminal, the objects typically encountered, and the actions that typically ensue. Schema theory assumes that there are memory structures *(schemata)* in memory for recurrent situations that are experienced, and that a major function of schemata is to construct interpretations of new situations.

The objects of a schema may be thought of as variables or slots into which incoming information can fit. If enough slots of a particular schema are filled, it becomes *active.* An active schema can then guide you to seek information to fill its remaining slots. If such additional information is not available in the environment, then you will fill its slots with information typical of a particular situation, activate its procedures, and access as needed any other knowledge it contains. In effect, the schema is a prototypical structure that can incorporate observed phenomena. People often react very rapidly and effectively to incoming stimuli. Recognizing a familiar face (for example) seems to happen almost instantaneously, and the speed is independent of the number of faces one knows. The features of a face fit into slots in "face schemata," and one of these schemata becomes active and provides the person's name.

Schemata are closely related to chunks. When a chess player looks at the board, the individual pieces can fit into the slots of "chess-pattern schemata." These schemata then provide the names or symbols that represent the patterns in memory, and later enable the player to retrieve the individual pieces when they are needed. Further, a set of active schemata can activate some higher-level "position schema," which provides the optional move. With the concept of schemata in mind, we can look in more detail at how knowledge and its organization affect problem solving.

It is now easier to think about how problem representations are formed. Essentially, they are formed in terms of the existing schemata and the slots they contain. If a problem is of a very familiar type, it can trigger an appropriate problem schema; if not, some more-general schema will be triggered. In any event, the slots in the schema control what features of the problem are incorporated into the representation; features that do not fit into a slot will be ignored.

Once a schema is triggered, a solver can decide on the solution if the schema contains the necessary information. If it is a specific and appropriate schema, it might contain precisely the right procedures, enabling the solver to proceed easily and rapidly. If it is a general schema, it might only contain a general prescription for how to proceed. In this situation the solver will have to search for procedures which fit the given problem and the general prescription. The solution will then be much more difficult to achieve, and may be impossible to achieve if the proper procedures can not be found. If an inappropriate schema is somehow triggered, the solver will not make any progress at all. Thus the importance of the knowledge structure, how it is organized into schemata, becomes clear. It is the organization and structure provided by schemata that allow relevant knowledge to be found in memory. Thus either lack of knowledge or lack of access to knowledge because of inadequate structure may be the reason for failure to solve a problem.

There is some experimental support for this interpretation. For instance, Hinsley et al. (1978) asked high school and college students to classify 76 algebra problems in any way they wished. There was considerable agreement among the students on the types of categories. Problems were grouped that would be solved in the same way, such as triangle problems, ratio problems, and river-current problems. Furthermore, students could classify these problems after hearing only the first sentence. For example, a problem starting with "John walked three miles east and then four miles north" could readily be classified as a triangle problem, that is, one using the Pythagorean theorem for solution. The rapidity with which problems are classified appears to rule out the possibility that problem categories are identified after the students have formulated a

solution. Instead, it is suggested that the problems were indeed triggering some appropriate schemata in memory.

The existence of specific schemata can also be uncovered by showing how they can be mistakenly accessed. For example, poorer students, in particular, will classify problems erroneously if they contain "cover stories" that trigger a specific schema (Silver, 1981). A problem used by Hinsley et al. (1978) might be mistakenly identified as a triangle problem, even though it is actually a distance-rate-time problem, because irrelevant information about the triangular relation between the three problem elements has been introduced. Good students were not fooled, however.

A similar contrast was found by Chi et al. (1981). Students who had completed introductory physics with an A grade and physics instructors both grouped typical physics problems. The students tended to group problems that contained similar physical entities (as shown in Figure 10.5), whereas the instructors grouped problems according to the underlying principle (as shown in Figure 10.6). These studies show that good problem solvers are not fooled by the superficial features of a problem statement.

The results of these categorization studies, which show that good problem solvers are not deceived by cover stories, indicate that the skill of expert problem solvers arises from the complexity and completeness of their schemata. Their schemata must contain rules that are more complex than simple linking of superficial (or *primary*) cues in the problem statement with solution procedures. The schemata of the novice learners may be developed initially with these simple rules, but in order for novices to learn not to be misled by cover stories, they must also develop *secondary* knowledge (Chi et al., 1981), that is, knowledge that incorporates the interaction between the primary cues in the problem statement. Hence, a complete schema must contain not only the procedures (such as the equation to compute the hypotenuse of a triangle), but also the primary and secondary cues needed to identify the problem type appropriately.

Another way to illustrate these separate components of schemata is to use *problem isomorphs*. These are sets of problems whose underlying structures and solutions are all the same, but whose context can be quite different. An example of a problem isomorph is the tea ceremony (Hayes and Simon, 1974). In it, three people are conducting an oriental tea ceremony, and they must distribute responsibility for various parts of the ceremony among themselves according to certain rules. In actuality, this puzzle is the familiar Tower of Hanoi, with the pegs replaced by people, and the disks by parts of the ceremony. Anyone who is familiar with the Tower problem and who notices the correspondence can solve the Tea Ceremony quite easily. However, studies show that virtually no one notices the similarity. One way to interpret this finding is that the slots of the Tower of Hanoi schema were designed specifically for pegs and disks. When people and ceremonies were presented instead, the Tower of Hanoi solution schema was therefore not accessed, even though it contained very efficient procedures for solving the tea-ceremony puzzle.

In sum, studies on the solution of problems where a great deal of domain knowledge is involved indicate clearly that a very relevant part of success in

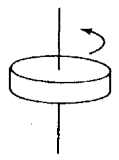
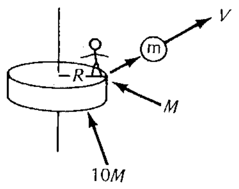
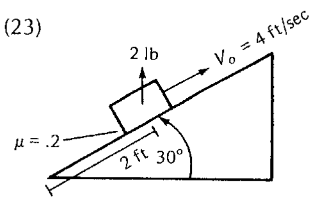
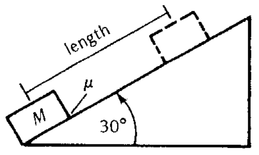
| *Diagrams depicted from problems categorized by novices within the same groups* | *Novices' explanations for their similarity groupings* |
|---|---|
| Problem 10 (11)  | *Novice 2:* "*Angular* velocity, *momentum,* circular things" *Novice 3:* "*Rotational* kinematics, *angular* speeds, *angular* velocities" *Novice 6:* "Problems that have something *rotating; angular* speed" |
| Problem 11 (39)  | |
| Problem 7 (23)  | *Novice 1:* "These deal with blocks on an *inclined plane*" *Novice 5:* "*Inclined-plane* problems, coefficient of *friction*" *Novice 6:* "Blocks on *inclined planes* with angles" |
| Problem 7 (35)  | |

**Figure 10.5**
Examples from novices' problem categories. Problem numbers represent problem and chapter number from Halliday and Resnick.(1974) (Taken from Chi,Feltovich, and Glaser, 1981.)

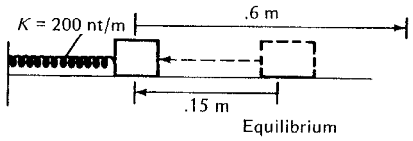
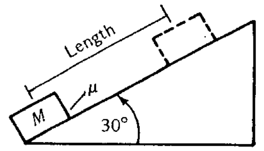
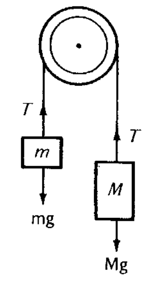
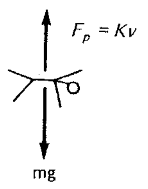
| *Diagrams depicted from problems categorized by experts within the same groups* | *Experts' explanations for their similarity groupings* |
|---|---|
| Problem 6 (21)  | *Expert 2:* "*Conservation of Energy*" *Expert 3:* "*Work-Energy Theorem.* They are all straight forward problems." *Expert 4:* "These can be done from energy considerations. Either you should know the *Principle of Conservation of Energy,* or work is lost somewhere." |
| Problem 7 (35)  | |
| Problem 5 (39)  | *Expert 2:* "These can be solved by *Newton's Second Law*" *Expert 3:* "*F = ma; Newton's Second Law*" *Expert 4:* "Largely use *F = ma; Newton's Second Law*" |
| Problem 12 (23)  | |

**Figure 10.6**
Example from experts' problem categories. Problem numbers represent problem and chapter number from Halliday and Resnick(1974). (Taken from Chi,Feltovich, and Glaser, 1981.)

problem solving is the ability to access a large body of well-structured domain knowledge. Therefore, one important direction of current research is to explore how a large body of knowledge is organized and represented, so that it can be easily accessed for successful solving of problems.

## Ill-Defined Problems

Perhaps the best way to define an ill-defined problem is by default. That is, problems which do not fall into the class of well-defined problems, such as those that have been studied and described in the preceding discussion, may be considered ill-defined problems. One framework that can be used to conceptualize ill-defined problems draws on the information-processing approach we mentioned earlier in discussing puzzle problems. In that framework, a problem has a clear initial state, a set of permissible operators, and a goal state. A problem qualifies as *ill*-defined if any one or all of the three components are *not* well specified.

For example, the initial state may be vague, as in economics problems. Our economy is so complex that we really do not understand it very well. For any given economics problem, we have only a partial picture of the initial state, based on all the various statistics the government and other agencies collect. Not only do we have incomplete data to describe the initial state, but even professional economists cannot agree on the interpretation of the statistics that we do have.

A problem can also be considered ill defined if the operators are not well specified. In our economics problem, the various actions that might be taken to modify the initial state are not clear, and many possible actions have not yet even been formulated.

Finally, the problem is ill defined if the goal state is not clear. What conditions of controlled inflation and unemployment are to be attained? How much is too much? As you know, experts disagree vehemently on the answer to these questions. In fact, one prominent characteristic of the ill-defined problem is that there is generally a lack of consensus even among experts about what the appropriate solution is.

There has been very little research done on ill-structured problems. One interesting piece of work, recently conducted by Voss and his associates (Voss et al., 1983a; Voss et al., 1983b), used problems from the social sciences, much like the economics problem we have just described. Imagine you are the Minister of Agriculture for the Soviet Union. Crop productivity has been too low for the past several years. What would you do to increase crop production? This ill-structured problem has all the components unspecified. The initial state is far more complicated than just bad crops. Among other things, the Soviet political system, current agricultural methods, the amount of arable land, and the weather are all part of it. The task is to find some actions or operations that might improve the situation, but there is no mention at all of what they might be. And, finally, the goal is vaguely stated. How much of an increase is an appropriate goal? 5 percent? 20 percent? 200 percent?

In order to try to disentangle the various effects of knowledge, Voss and colleagues used subjects who were political scientists specializing in the Soviet Union, students who were taking a political-science course on Soviet domestic policy, and chemistry professors. Not surprisingly, they found strong knowledge effects. The most comprehensive and detailed solutions were produced by the Soviet experts, and the worst were produced by the students and the chemists. About 24 percent of the solution protocols of the Soviet experts focused on elaborating the initial state of the problem, as opposed to 1 percent for the students and the chemists.

Prominent in the experts' protocols at the initial phase of problem solving were the identification of possible constraints, such as the Soviet ideology and the amount of arable land. Defining constraints will provide a means of testing possible solutions, such as fostering private competition (a capitalist solution) or increasing planting, which can be rejected immediately under these constraints.

The obvious way to solve a problem of this sort is to eliminate its causes; so it is important to enumerate causes in the problem representation. If you realize that there are a series of separate causes, you will naturally decompose the problem into subproblems, that is, use subgoaling. All the subjects, from expert to novice, used this strategy. The differences, however, were that the experts tended to create a few very general subproblems that might encompass several related causes, whereas the novices related solutions very directly to individual causes. For instance, one expert identified the Soviet bureaucracy, the attitudes of the peasants toward agricultural practices, and the lack of infrastructure (railroads, fertilizer plants, fuel-distribution networks, etc.) as the main subproblems. In the most extreme examples of this tendency, the problem was recast into a single "subproblem," such as inadequate technological development.

There are at least two conclusions we can draw from this research. First, even in dealing with ill-defined problems, solvers use heuristics not unlike those, such as subgoaling, that they use in well-defined problems. Second, the very nature of ill-defined problems means that solvers define the problems better for themselves. This suggests that knowledge of the problem domain really makes it easier for a solver to define the problem, such as in identifying the constraints. For example, Reitman (1965) found from his protocols of a composer writing a fugue that, at any moment in time, the composer was really dealing with a very well-defined subproblem, even though the subproblem required some initial work to define it.

Because ill-defined problems require this special component, that is, a process for adding information to the problem situation, sometimes people refer to the solution of these problems as a creative act (Newell et al., 1964). Another name for a creative act is *insight*. This term often seems to imply that the solution was achieved in one single step, rather than in a series of discrete transformations, as happens with well-defined problems. The two-string problem we mentioned earlier seems to match this description. However, contemporary research is just beginning to explore the process of insight, and it appears that insight itself can be decomposed into several component processes, such as encoding the information in a selective way, combining information in a novel way, or comparing certain aspects of two objects (Sternberg and Davidson, 1983). For example, Fleming's discovery of penicillin required him to notice that the bacteria in the vicinity of a moldy dish containing a culture that had been destroyed. We could therefore say that Fleming was simply more insightful or creative than the other researchers. However, encoding information selectively requires that a person *know* which piece of information is relevant. That is, Fleming must have had some stored knowledge that only certain potent substances could destroy bacteria. Two tentative conclusions can thus be gleaned from the current preliminary research. First, creative and insightful acts may not necessarily be discrete and unitary; they may actually be composed of a sequence of subprocesses. Second, the execution of these subprocesses in an apparently creative way may require some existing stored knowledge.

## Summary

Over the years, psychologists have learned a lot about the nature of the problem-solving process. The importance of the initial representation of a problem was discovered quite a few years ago by the Gestalt school in its examination of insight problems. The actual nature of initial representations and their influence on problem solving was made clear only in the last few decades, however, when the notion of a solution space was developed. Not too long ago researchers, especially those in artificial intelligence, thought that effective problem solving might result mainly from applying general strategies for guiding a searching process through problem spaces. Early attempts to create computer programs to solve problems, such as the General Problem Solver, took this approach.

As we have seen, though, this picture of problem solving has recently been shown to be far too simple. Specific knowledge of a domain is of overriding importance in the effective solution of problems. In addition, this knowledge must be well-structured, so that relevant knowledge can be accessed at the proper time. Research is beginning to uncover just what "well structured" means, but considerable work is left to be done on how we can retrieve information in a rapid and effective manner from the wealth of knowledge we all possess.

Even more work needs to be done on the kinds of problems that are probably the most important to us: ill-structured, real-world problems. The process of solving such problems is difficult, complex, and thus difficult to study, especially because it is often not clear whether an ill-structured problem has been solved. In addition, the most important real-world problems are often solved (or made worse) by means of complex (and little understood) social interactions.

## References

Akin, O. 1980. *Models of Architectural Knowledge.* London: Pion.

Atwood, M. E., and P. G. Polson. 1976. A process model for water jug problems. Cognitive Psychology, 8, 191–216.

Chase, W. G., and H. A. Simon. 1973a. Perception in chess. *Cognitive Psychology,* 1973, 4, 55–81.

———. 1973b. The mind's eye in chess. *In* W. G. Chase, ed., *Visual Information Processing* (New York: Academic Press).

Chi, M. T. H., P. J. Feltovich, and R. Glaser. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science,* 5, 121–152.

de Groot, A. 1966. Perception and memory versus thought: Some old ideas and recent findings. *In* B. Kleinmuntz, ed., *Problem Solving: Research, Method, and Theory* (New York: Wiley).

Egan, D. E., and B. J. Schwartz. 1979. Chunking in recall of symbolic drawings. *Memory and Cognition,* 7, 149–158.

Elstein, A. S., L. S. Shulman, and S. A. Sprafka. 1978. *Medical Problem Solving.* Cambridge, Mass.: Harvard Univ. Press.

Ernst, G. W., and A. Newell. 1969. *GPS: A Case Study in Generality and Problem Solving.* New York: Academic Press.

Halliday, D., and R. Resnick. 1974. *Fundamentals of Physics.* New York: Wiley.

Hayes, J. R. 1966. Memory, goals, and problem solving. *In* B. Kleinmuntz, ed., *Problem Solving: Research, Method, and Theory* (New York: Wiley).

Hayes, J. R. and H. A. Simon. 1974. Understanding written problem instructions. *In* L. W. Gregg, ed., *Knowledge and Cognition* (Hillsdale, N.J.: Erlbaum).

Hinsley, D. A., J. R. Hayes, and H. A. Simon. 1978. From words to equations: Meaning and representation in algebra word problems. *In* P. A. Carpenter and M. A. Just, eds., *Cognitive Processes in Comprehension* (Hillsdale, N.J.: Erlbaum).

Klahr, D. and M. Robinson. 1981. Formal assessment of problem-solving and planning processing in preschool children. *Cognitive Psychology,* 13, 113–148.

Larkin, J. H., J. McDermott, D. P. Simon, and H. A. Simon. 1980. Models of competence in solving physics problems. *Cognitive Science,* 4, 317–345.

Maier, N. R. F. 1931. Reasoning in humans, II: The solution of a problem at its appearance in consciousness. *Journal of Comparative Psychology,* 12, 181–194.

National Academy of Sciences. 1981. *Outlook for Science and Technology: The Next Five Years.* Washington, DC: N.A.S.

Newell, A., J. C. Shaw, and H. A. Simon. 1964. The processes of creative thinking. *In* H. E. Gruber, G. Terrell, and M. Wertheimer, eds., *Contemporary Approaches to Creative Thinking, Vol. 3* (New York: Atherton Press).

———. 1958. Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development,* 2, 320–335.

Newell, A., and H. A. Simon. 1972. *Human Problem Solving.* Englewood Cliffs, N.J.: Prentice-Hall.

Reed, S. K., and H. A. Simon. 1976. Modeling strategy shifts in a problem solving task. *Cognitive Psychology,* 8, 86–97.

Reitman, W. 1965. *Cognition and Thought.* New York: Wiley.

Rumelhart, D. E. 1981. *Understanding Understanding.* La Jolla: Univ. of California, Center for Human Information Processing.

Silver, E. A. 1981. Recall of mathematical problem information: Solving related problems. *Journal for Research in Mathematics Education,* 12, 54–64.

Simon, D. P. and H. A. Simon. 1978. Individual differences in solving physics problems. *In* R. Siegler, ed., *Children's Thinking: What develops?* (Hillsdale, N.J.: Erlbaum).

Simon, H. A., and M. Barenfeld. 1969. Information processing analysis of perceptual processes in problem solving. *Psychological Review,* 76, 473–483.

Sternberg, R. J., and J. E. Davidson. 1983. Insight in the gifted. *Educational Psychologist,* 18(1), 51–57.

Voss, J. F., T. R. Greene, T. A. Post, and B. C. Penner. 1983a. Problem solving skill in social sciences. *In* G. Power, ed., *The Psychology of Learning and Motivation: Advances in Research and Theory,* Vol. 17. New York: Academic Press.

Voss, J. F., S. W. Tyler, and L. A. Yengo. 1983b. Individual differences in the solving of social science problems. *In* R. F. Dillon and R. R. Schmeck, eds., *Individual Differences in Cognition* (New York: Academic Press).

Wilkins, D. 1980. Using patterns and plans in chess. *Artificial Intelligence,* 14, 165–203.