In Attention and Performance VIII, R. Nickerson (ed.) Erlbaum, Hillsdale, NJ, 1980

35 Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category

Allen Newell
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania
United States

ABSTRACT

The notion of a problem space is well known in the area of problem solving research, both in cognitive psychology and artificial intelligence. The *Problem Space Hypothesis* is enunciated that the scope of problem spaces is to be extended to all symbolic cognitive activity. The chapter is devoted to explaining the nature of this hypothesis and describing some of its potential implications, with no attempt at a critical marshalling of the evidence pro and con. Two examples are used, one a typical problem solving activity (the Tower of Hanoi) and the other syllogistic reasoning. The latter is an example where the search behavior typical of problem spaces is not clearly in evidence, so it provides a useful area to explore the extension of the concept. A focal issue used in the chapter is the origin of the numerous flow diagrams that serve as theories of how subjects behave in tasks in the psychological laboratory. On the Problem Space Hypothesis these flow diagrams derive from the interaction of the task environment and the problem space.

INTRODUCTION

I am concerned with human goal-oriented cognition: what humans do when they bring to bear what they know to attain some end. I take my title from the session in which I have been invited to give an opening presentation, because it exhibits a particular feature of cognitive psychology's current state that I can use as a starting point.

Substantial areas of psychological study exist in *reasoning* (Falmagne, 1975, Revlin & Mayer, 1978, Wason & Johnson-Laird, 1972); *problem solving*

(Greeno, 1977, Newell & Simon, 1972); and *decision processes* (Slovic, Fischhoff, & Lichtenstein, 1977). Yet, in looking at any of these fields it is hard to detect the existence of the others. These three areas must in most ways be the same: problematic situations presented verbally to be dealt with by cognition and decision. Indeed, we often use the same term in all three areas. We "decide" which of two probabilistic bets to take (much studied in the area of decision processes); "decide" what chess move to make (much studied in the area of problem solving); and "decide" which conclusion follows from a syllogism (much studied in the area of reasoning). Similar semantic games could be played with the other terms. However, studies of each of these areas rarely give more than lip service to the results and theories from the others. What should be a unified scientific endeavor seems fragmented.

Multiple diagnoses for this fragmentation are possible. Deny the symptom. Argue, along with Voltaire's Dr. Pangloss, that it reflects a perfectly appropriate state for a developing science. Note that the areas have distinct intellectual roots, and mutter that history explains (and excuses) all. Grasp the nettle and assert the three psychological domains to be genuinely distinct. However, I do not wish to argue the case in detail here. I do wish to assume that enough others share my unease to let me use this issue of fragmentation as a point of entry to what fundamental category of cognition might help resolve it.

Consider the tasks used by psychologists to study cognition. Traveling through them yields a distinct impression of just one damn task after another: Maier's two-string problem, Edward's book-bag and poker-chip task, R. Sternberg's analogies task, Restle's Tangled Tale sentence, Wertheimer's parallelograms, Revlin and company's classical syllogisms, and Newell and Simon's cryptarithmetic. The world of all tasks seems like a zoo formed from a medieval bestiary—far from random, lots of odd structure, perhaps bias (too many toy tasks?), but without essential orderliness.

Repeat that trip, stopping at tasks where some theorizing has occurred. For much of the best, we find a *flow diagram* presented, such as the one in Fig. 35.1 (Revlin & Leirer, 1978), with control flowing between boxes that carry labels such as *encode* and *compare*. These flow diagrams constitute the framework of the theory of each task.

Figure 35.1 is taken from the area of reasoning. Analogous diagrams could be found in abundance in all three areas. I assume such diagrams are familiar enough not to require further illustration. It is not uncommon to criticize flow diagrams, questioning their usefulness as a theoretical language; I have been known to do so myself (Newell, 1973). However, my question here is different. I am willing to accept that these flow diagrams acquire additional rules and data associated with the boxes, even equations and simulation programs. My question is: *Where* did those flow diagrams originally come from? They are different for every task, though they all surely bear a family resemblance. Theorists seem simply to write a different theory for each task. Details get

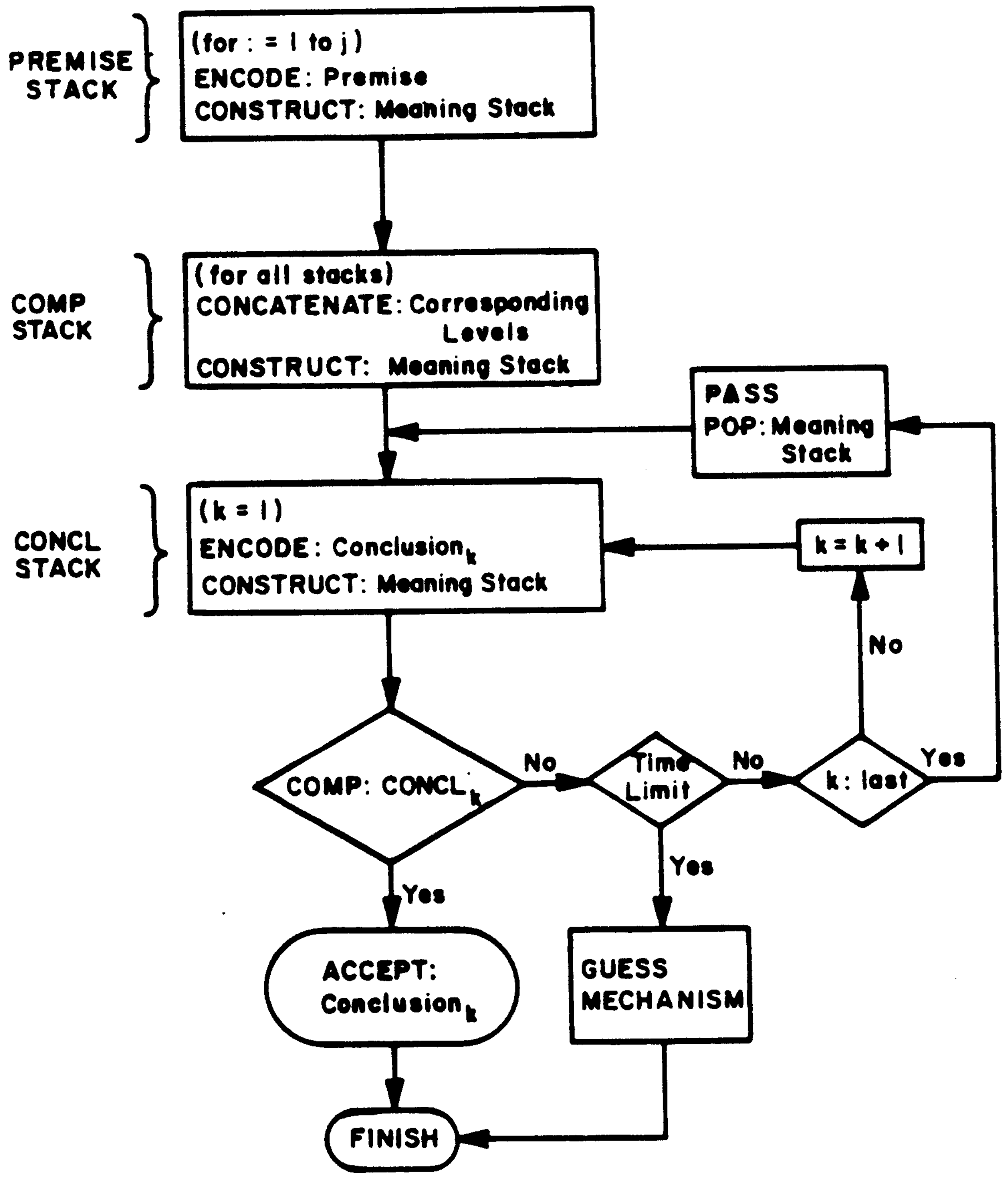


FIG. 35.1. Typical flow diagram: Conversion model of formal reasoning (Revlin & Leirer, 1978).

filled in experimentally, but the frameworks (i.e., the flow diagrams) are just written down.

The diversity of these flow diagrams is connected to the fragmentation of cognitive studies. Diversity per se does not cause the trouble, for the tasks themselves are indeed diverse and the theories must reflect that. The difficulty lies in the emergence of each of the microtheories full blown from the theorist's pen. There is no way to relate them and thus they help ensure the division of the study of human cognition into qualitatively isolated areas.

The difficulties in finding communality do not rest just in giving common technical content to the terms in the boxes, such as *encode* and *compare*, though these surely pose problems. They emerge with each flow diagram, and equally without essential discipline. They seem weak vessels to try to shore up, though some are trying (Sternberg, 1979). But beyond the terms themselves the structure of the flow diagrams also embodies the theory and must be dealt with as well.

The diversity of these flow diagrams arises in large part because these diagram theories incorporate the detailed structure of each task within the very fiber of the theory. They are a version of the magician's trick—by the time the theory emerges, the scientific magic has already taken place. Though the theorist does not have a theory of how the subject would do a task, he himself can do what the subject does (i.e., analyze the task); hence, he can create each theory separately out of his own subjective analysis.

I have no quarrel with the theorist's direct analysis as a source of theoretical ideas. The difficulty stems from the volume of separate analyses that produce a corresponding volume of distinct flow diagrams. The prescription I take from this is the need to understand where these flow diagrams come from—not to understand where theorists get them but to understand where subjects get them. Given that humans are cognitively integrated, how does this organization occur. Can we understand how the task gives rise to the flow diagram?

I propose a solution to this in terms of *problem spaces*, a concept already familiar in the study of search in human problem solving (Erickson & Jones, 1978) and applied widely there (Simon & Lea, 1974). It was introduced formally in Newell & Simon (1972), but derives from extensive work in artificial intelligence, where all programs characterized as *heuristic search* provide prototypic examples of problem spaces. The central proposition of this chapter is to extend the scope of this concept:

Problem Space Hypothesis: The fundamental organizational unit of all human goal-oriented symbolic activity is the *problem space*.

In general terms this proposition is clear enough. There are things called *problem spaces*, which humans have or develop when they engage in goal-oriented activity. To understand such activity is to discover what problem spaces a human is using. From these flow the descriptions and predictions of interest, especially those concerned with how behavior is organized to accomplish tasks. The proposition is inclusive, claiming coverage of all symbolic goal-oriented activity, but hedges on whether all cognitive activity is symbolic. [The notion of *symbolic* (Newell & Simon, 1976) is ultimately central to the hypothesis but doesn't enter explicitly into this chapter.] It is an empirical hypothesis about the nature of human behavior. It is clearly of more general import than just where flow diagrams come from; that issue is just an entry point. In the words of the title, the hypothesis claims the problem space to be a fundamental category of cognition.

This chapter attempts to make this proposition intelligible. It does not present the case for it critically. There is no space for that. We avoid formalism as much as possible to concentrate on the central notions. We lay out quickly and oversimply the notion of a problem space, using the Tower of Hanoi task as an example, where the notion has already been applied. Then we pick another example, syllogistic reasoning, to develop what the hypothesis means in areas other than problem solving. Again, space permits only one such example, though the hypothesis is intended much more broadly. However, this provides enough of an illustration to discuss some general issues.

PROBLEM SPACES AND PROBLEMS

The Tower of Hanoi puzzle provides a convenient initial example to make the notion of a problem space concrete. It is normally considered to be a problem-solving task (Nilsson, 1971, Simon, 1975) and has been analyzed in essentially problem space terms.

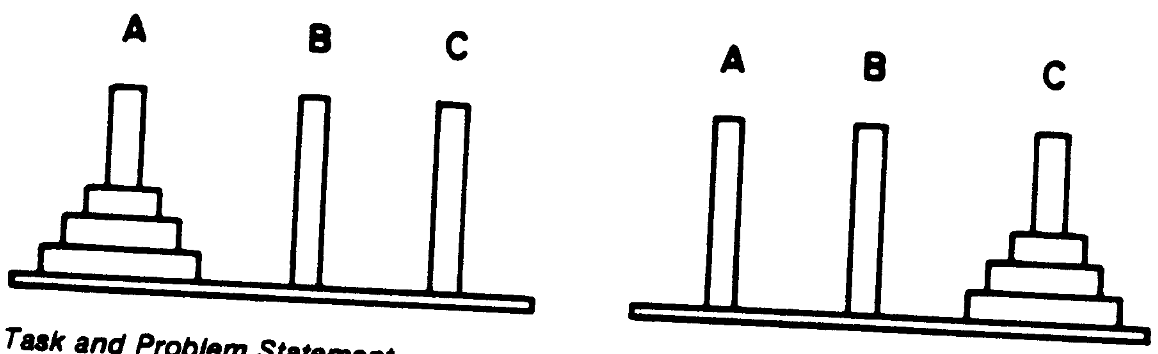
We start with informal definitions:

Problem Space: A problem space consists of a set of symbolic structures (the *states* of the space) and a set of *operators* over the space. Each operator takes a state as input and produces a state as output, although there may be other inputs and outputs as well. The operators may be partial (i.e., not defined for all states). Sequences of operators define *paths* that thread their way through sequences of states.

Problem: A problem in a problem space consists of a set of *initial* states, a set of *goal* states, and a set of *path constraints.* The problem is to find a path through the space that starts at any initial state, passes only along paths that satisfy the path constraints, and ends at any goal state.

A problem space is a set of symbolic structures within which to move around, an arena wherein many specific problems can be posed and attempted. A problem space and problem are mental constructs (i.e., mental operators and states), though they may lead to external actions. A subject *has* a problem space if he can mentally represent the states of the space and carry out the operations. He *has* a problem in a problem space if: (1) he has the space; (2) he can obtain representations of the initial states, recognize paths that satisfy the path constraints, and recognize the goal states; and (3) his behavior is controlled so as to attempt the problem in the space.

The task and one possible problem space for the Tower of Hanoi are given in Fig. 35.2. The states are all the configurations of a fixed set of disks on three



Task and Problem Statement

A board has three pegs, A, B and C (as shown at left). On Peg A are N disks of different sizes (3 in the diagram), in order with the largest at the bottom. The task is to get all the disks on Peg C in the same order (as shown at right). A disk may be moved from any peg to another, providing (1) that it is the top disk on its peg and (2) that it cannot be put on top of a disk smaller than itself.

Problem Space

States: Arbitrary configurations of the N disks on the three pegs.
Operators:

Move a disk by removing it from a peg and putting it on another peg.
Recognize a configuration as an instance of a pattern.

Problem

Initial state: The configuration shown in the diagram at left.
Goal state: The configuration shown in the diagram at right.
Path constraint: No disk may be placed on a smaller disk.

FIG. 35.2. Tower of Hanoi: Problem space.

pegs. There are two operators. One takes an arbitrary disk and peg as input and moves the disk to the peg (a new state). The other produces a symbolic expression asserting that a given configuration fits a given pattern. Possible patterns include concrete configurations, such as the two in the figure, and also patterns such as "peg P is empty" and "the disks are not in order on peg P". The problem statement specifically asserts this space to be the arena in which action takes place. It gives both initial and goal states explicitly. It imposes a path constraint; this is necessary because the space itself consists of *all* configurations of disks. It is not difficult to see that the Tower of Hanoi puzzle can be expressed in problem space form. Normal adults have the ability to create the space of configurations of Fig. 35.2 and perform the moves and recognitions. Thus, they can *have* the space of Fig. 35.2 in the manner called for—select and apply operators, test for results, etc.

Other problem spaces are possible for the Tower of Hanoi problem. Indeed, any problem space that contains this one works fine. For instance, the basic operators might be pick up a disk, move the hand, and deposit the disk. Then the move operator described as unitary in Fig. 35.2 becomes composite. Additional path constraints become necessary: A disk can be set down only on a peg; and only one disk can be moved at a time. Without these, one hand could hold a disk in mid-flight while another is moved.

Smaller spaces may also be possible. The states could be limited to ordered stacks of blocks and the move operator could apply only where the disk on the receiving peg was larger than the disk being moved. This space incorporates the path constraint as part of the operator. Whether this problem space is possible for a given subject depends on whether the subject's behavior can become organized so that the test for legality of a move is reliably incorporated within the move itself. If a move must actually be considered and the result viewed to see if it is legal, then the subject can have the space of Fig. 35.2 but cannot have this smaller space.

Each problem space provides a possible way to represent a task so as possibly to obtain a solution. It describes an ability of the subject to confine behavior to the problem space. It describes the units of behavior (i.e., the operators) the subject will use in working on a task.

Resource and capacity limits exist on processing. In problem spaces these take the form of two principles.

Serial Action: At most one problem space operator can be performed at a time.

Thus, at any point in time the subject is located at a single *current state* to which a *current operator* is being applied to yield a *current result* (i.e., a new state). Seriality specifically refers to this action. More than one problem or problem space may be active [e.g., to apply an operator in one problem space may require passing into another problem space (concerned with the mechanics of the operator), thus working in two problem spaces at once, though of course doing only one operator].

Finite Stock: The subject has a limited set of states (the stock) available to become the current state.

This is a memory and access limitation. However, the stock is not identical with short-term memory; some states may exist in long-term memory or be available perceptually as external memory. The actual size of the stock is variable in the same way as the size of human short memory, depending on the complexity of the state, the external memories used, etc.

In the Tower of Hanoi problem these two principles mean that a subject can consider only one move at a time (in thought, not just in the external world) and, after having considered a move, there will be only a few new states possible: the new state resulting from the move; the state from which the move was just considered; the state that is set up in the external world (if that is different); the original initial state; possibly another remembered one.

Given a problem in a problem space, the only way a subject can solve the problem is by *searching* in the space: working out from the current state by applying operators, adding new states to the stock to be used at new points of search, evaluating whether the results help, etc. To accomplish this search requires performing repeatedly a fixed set of functions:

Search Control: The following functions determine the behavior in a problem space while working on a given problem.

Decide to quit the problem.
Decide if a goal state has been produced.
Select a state from the stock to be the current state.
Select an operator to be the current operator.
Decide to save the new state just produced by an operator.

These functions operate within a cycle consisting of repeating the following three steps:

1. Select a state; select an operator.
2. Apply operator to state, producing new state.
3. Decide if a goal state; decide to quit; decide to save the new state.

The subject has a mechanism (the *architecture*) for carrying out this control cycle. It is a fixed mechanism that works for all problem spaces: bringing the selected operator and state in contact, so the new state can be produced; bringing the decision processes in contact with this new state, so goal attainment can be determined; and so on. Its exact properties are important, including memory management for the limited stock and basic error detection, though we do not discuss the architecture further.

More than one function can be performed at some steps: the two selections in step 1 and the three decisions in step 3. Within a step no process takes priority over any other. In what order the selection of an operator and of a state takes place depends on the content of the selection processes. Likewise, depending on the particular situation, the decisions could be made in any order.

Control over the search depends on the knowledge of these functions that the subject has *immediately available*. This restriction to immediate availability arises as follows. All knowledge about a task is obtained by taking steps in a problem space; that is what it means to be working in a problem space. Hence, choosing what step to take cannot itself be an extended deliberation that considers and reasons about the problem—that would involve taking steps in the problem space, contra assumption. What the control processes of selection and decision can involve is applying stored knowledge available in the subject's long-term memory. But there is a limit to how much can be done without thinking a new thought about the task itself (i.e., without moving in the problem space).

The restriction implies that subjects cannot normally take too long to make a step, though no strict temporal limit exists. Subjects take steps in a problem space every few seconds (Newell & Simon, 1972). Where the time per state is fairly long (10–15 sec), either the state is relatively complex, taking time to assimilate, or the operators take time to apply. Unlike the search control, the application of an operator need not be immediate but can involve going into a different problem space. A typical example is applying an unfamiliar complex formal rule, as in algebra.

The restriction that search control involves only bringing to bear immediately available knowledge is important, because it limits the complexity of the search control process. It lets us think of the search control primarily in terms of the knowledge it embodies rather than the processes for making that knowledge effective.

A subject can attempt to solve a problem in a problem space with *any* body of search control knowledge: from none at all, yielding undirected search, to knowledge that completely specifies all choices correctly, yielding the solution directly. Thus, to have a problem space is already to have relevant ways of behaving in a task situation, though the larger the space and the less effective the control knowledge, the smaller the chance of success. For instance, if a subject has only the knowledge represented in Fig. 35.2 (i.e., no special search control knowledge at all), the problem can still be attempted. A combinatorial search problem then occurs, which can be analyzed in terms of the branching factor of the search (three) and the depth to solution ($2^N - 1$ for N disks) (Nilsson, 1971).

Subjects of course are never this ignorant; they have some effective knowledge for controlling the search. Figure 35.3 gives some examples. In accordance with the previous remarks, it is sufficient to express the search control just by the knowledge involved, without explicit characterization of the processing. K1 to K3 simply provide minimal knowledge for their respective control functions. K4 says to move forward with each step; it induces a depth-first strategy. K5 is the most obvious means-ends principle. The Tower of Hanoi is problematic just because K5 does not suffice (i.e., disks cannot simply be moved to the desired peg). It is a puzzle just because K6, the obvious means-ends knowledge to avoid difficulties, is ineffective with more than two disks.

K7–K9 reflect a general avoidance of duplicate and redundant activity. K7 cuts down the branching factor from 3 to 2. K8 cuts it to 1 on every other move, giving an effective branching factor of 1.4. Finally, K9 is a slightly more penetrating formulation to avoid looping, reducing the branching to 1 (except at the initial position). However, if K9 is to be implemented, the subject must remember the peg from which the smallest disk came, which is a minor augmentation of the problem space state. Thus, adding K7–K9 completely

Decide to quit the problem
K1. Quit if succeed.
K2. Quit when told by experimenter.

Decide if goal state has been produced
K3. A state is a goal state if it is the exact pattern desired.

Select a state from the stock to be the current state
K4. Make new state the current state.

Select an operator to be the current operator
K5. Move a disk to the peg specified by a goal state.
K6. Move an obstructing disk to another peg.
K7. Do not move back to just prior position.
K8. Do not move a disk twice in a row.
K9. Do not move smallest disk back to its just prior peg.

Decide to save a new state just produced by an operator
K10. Add newly produced state to the stock.

FIG. 35.3. Tower of Hanoi: Search control knowledge.

determines the course of problem solving. It leaves open only the (fateful) choice on the very first move. Subjects do exist who proceed in exactly this fashion (Anzai & Simon, 1979); upon discovering they have made the wrong initial choice, they recall the initial move, chose the alternate one, and then proceed according to the same heuristics to solve the problem.

K10 goes hand in hand with K4. It would lead to remembering the entire path, except that short-term memory is limited. Additional control knowledge would be needed to determine which states to retain if some have to be abandoned. We haven't shown such knowledge, because its form depends on the nature of the architecture (i.e., what aspects of short-term memory can actually be controlled by the subject). Moreover, with the other search control knowledge of Fig. 35.3, the subject never goes back to a deliberately remembered state but simply always pushes forward.

Various means exist for organizing search control knowledge. One is to create new problems (i.e., subgoals), attempt them, and incorporate the results in the further search. For simplicity we cast the basic functions of search control to be those required, assuming the problem to be fixed. But the subject also selects problems and problem spaces. Though not requiring additional capabilities, except for memory management, much additional power and complexity accrues thereby, namely, the goal hierarchy. For instance, organizations of subgoals and search control knowledge can be fashioned into *methods*, which coordinate the selection and actions in various useful ways. A number of these methods, which have shown up repeatedly in *artificial intelligence* investigations of problem solving and elsewhere, have acquired familiar names:

Generate and Test: Generate in any way possible (e.g., systematically or haphazardly) a sequence of candidate states, testing each for whether it is the desired state.

Heuristic Search: Apply heuristics to reject possible operators from the current state and to reject newly produced states; remember the states with untried operators in some systematic way (different schemes yield search strategies, such as *depth first, breadth first, progressive deepening,* and *best first*).

Hill Climbing: Generate and apply operators from the current state; select one that produces a state with an improved evaluation and move to it.

Means-Ends Analysis: Compare the current state with the desired state to detect any difference; use that difference to select an operator that reduces or eliminates it; otherwise proceed as in heuristic search.

Operator Subgoaling: If an operator cannot be applied to a current state, set up a subgoal to find a state in which the operator can be applied; otherwise proceed as in heuristic search or means-ends analysis.

Planning: Abstract from the present state (by processing only selected information throughout) and proceed to solve the simplified problem; use what is remembered of the path as a guide to solving the unabstracted problem.

These are often called the *weak methods,* because they can be relevant when little is known about the task, though, when so evoked, they are not very powerful. However, they often suffice to solve a problem, if the space is not large or if extensive search is undertaken. These methods are simply organizations of search control knowledge, though it is beyond the scope of this chapter to lay this out. Functions, such as *evaluate* (hill climbing) and *detect differences* (means-ends analysis) can be performed within search control if sufficiently simple (i.e., sufficiently like a recognition). Complex versions would require search control to evoke subgoals to accomplish them in some problem space. The methods also are *open,* in that they do not specify completely all the functions and hence admit the addition of more search control knowledge (e.g., about differences in means-ends analysis).

The first two methods, generate-and-test and heuristic search, occur automatically in a problem space (indeed, heuristic search gave rise to the notion of the problem space). They are identified as distinct methods, because they are normally realized within control structures (standard programming languages) where they must be constructed and deliberately evoked, just like any other method. The other methods all require the task to have some additional characteristics (the weak conditions) that are known to the subject: Hill climbing requires evaluation of states; mean-ends analysis requires differences; operator subgoaling requires the conditions of operator applicability to be symbolizable; etc.

Let us add a single item of control knowledge for the Tower of Hanoi:

K11. Get an obstructing disk on the other-peg.

K11 expresses operator subgoaling, because the construct of obstruction derives from inapplicable operators. This differs from K6 only in setting a subgoal (*get*) rather than selecting an actual operator (*move*). With K11 some problems can be solved without stringent knowledge about avoiding duplicate paths (K8 or K9). The architecture is assumed to manage the resulting stack of subgoals that builds up as the knowledge is used repeatedly and recursively. The limits to subgoal management set limits to the effectiveness of solving Tower of Hanoi problems with just K11.

Subjects not only show problem-solving behavior in the Tower of Hanoi, they ultimately show skilled behavior, proceeding to solve each puzzle in a direct way that takes time but does not exhibit search. Simon (1975) has provided an analysis of four distinct, complete strategies for skilled behavior in the Tower of Hanoi. Three of the strategies arise by adding more search control knowledge to what we have assembled so far. The *goal-recursion strategy* simply expands K11 to apply to pyramids of disks rather than just disks, its key notion being the invention of the concept of pyramid by the subject. The *simple perceptual strategy* expands K11, not by the pyramid but by the less powerful notion of the largest obstructing disk (on the source peg). The *sophisticated perceptual strategy* adds to this latter an item that extends the notion of obstructions to the target peg as well as the source peg. (The fourth strategy, *move pattern*, raises issues about mimicing external behavior sequences that lie outside our illustration.)

The problem space hypothesis asserts that skilled, routine behavior is organized within the problem space by the accumulation of search control knowledge. This is just what we see in these three strategies. Though how the accumulation occurs is not given, the final result consists simply of the addition of search control knowledge. There is no fundamental difference between problem solving and routine behavior in control organization, except in completeness and adequacy of search control knowledge.

Simon (1975) embodies the various strategies in a *production system*, a species of pattern-directed rule-oriented programming system (Newell & Simon, 1972). This can be seen as an operational realization of the search control knowledge, and in fact such systems are active candidates for the underlying architecture of human cognition (Newell, 1973; 1980).

THE CATEGORICAL SYLLOGISM: A REASONING TASK

There is a long history of research into human performance on classical Aristotelian syllogisms. Its fascination arises from the basic clash in our civilization between rationality and irrationality, and its projection to the clash between logic and psychology. Should humans behave according to the dictates of formal logic? Do they have their own "psychologic"? Are they simply "irrational"? Formal syllogisms provide ample evidence that people in fact cannot reason very well. They make lots of mistakes in answering (say) "All A are B; some C are not B; are some A not C necessarily?" The tone for research was set many years ago by Woodworth & Sells (1935), whose hypothesis of an *atmosphere* effect describes one form of illogic (or at least superficiality) whereby subjects respond according to the affirmative or negative tenor of the syllogism.

Current research is constructing information-processing theories of how people perform syllogisms, decomposing the task into various stages (e.g., encoding, comparing, responding) and finding experimental demonstrations of which stages seem responsible for various aspects of performance (Falmagne, 1975; Revlin & Mayer, 1978). It forms a useful example for us, because though research is active and current, there is little contact between its processing models and those in problem solving and decision making, as a glance at the two recent books just cited shows. Thus, we can see what analysis in terms of problem spaces might add.

Standard syllogistic reasoning tasks are formed from three terms (e.g., A, B, C), combined into two assertions (the major and minor premise) involving pairs of terms (A and B; B and C), from which some assertions (the conclusion) about the third pair (A and C) may or may not follow. Four logical assertions are possible from the two quantifiers (*all, some*), negation (*no/not*), and the copula of implication (*are*):

Major Premise:	All A are B	
Minor Premise:	Some B are C	
Conclusion:	All A are C	Which of the list follow?
	No A are C	
	Some A are C	
	Some A are not C	
	Nothing follows	

Both abstract syllogisms (above) and concrete syllogisms (No Senators belong to the Harem Club; some sheiks belong to the Harem Club; therefore no sheiks are Senators) are used.

Current theory can be typified by the flow diagram shown earlier (Fig. 35.1) from Revlin and Leirer (1978). Its stages permit focusing on whether the difficulty lies in encoding (how sentences are interpreted) or in inferencing; and on whether the difficulty lies with the givens or the conclusion. Typical of recent work is the hypothesis (Revlin's conversion model) that subjects apply conversion operations that take "All A are B" into "All B are A," thus producing errors in the encoding stage, and the hypothesis (Erickson, 1978) that subjects solve the problem by constructing internal Venn diagrams but fail to construct all possibilities.

The problem space hypothesis implies that subjects solve syllogistic reasoning tasks by working in a problem space. They have a representation for the possible states of knowledge about the syllogism plus operators for generating and manipulating that knowledge. If their search control knowledge is sufficient, they will go directly to an answer; when faced with difficulties, they will search in this space (i.e., try different manipulations in attempting to solve the problem). They do not have simply an algorithm or method, as in Fig. 35.1. If such a flow diagram describes their behavior, then it is the resultant of the problem space and the task environment.

Let's look at one possible problem space for syllogistic reasoning.

Abstract Object Problem Space

Figure 35.4 shows a problem space based on positing objects that have the attributes, A, B, C mentioned in a syllogism. These objects are abstract: They can be either *necessary* or only *possible;* and they can have only some of the attributes mentioned in the syllogism. Thus, they represent various situations of partial knowledge, which the subject is to flesh out by moving in the space.

This space is a natural one. It corresponds to attempting to imagine the things that are being talked about in the syllogism, as in the following monologue about the syllogism: All A are B; no C are B; therefore necessarily some A are not C.

"OK, there are some things that have A and also have B.
Though some things that don't have A could also exist, and they could have B or not.

But things that have C cannot also have B.
Though things that don't have C could have B or not.

Now, does that force some things that have A to not have C?

Well, those things that have A, have B...
and having B they can't have C.