Psychological Review
1979, Vol. ———

The Theory of Learning by Doing

Yuichiro Anzai and Herbert A. Simon
Carnegie-Mellon University

This article proposes a theory of the processes that enable a student to learn while engaged in solving a problem. It gives a microscopic account of learning in a specific situation, based on a detailed analysis of a single human problem-solving protocol. It proposes general mechanisms, however, that make no specific reference to an individual subject or task, and it shows how these interact with specific task information gained during the problem-solving process. The adequacy of the mechanisms for producing the learning is guaranteed by a computer simulation of the process in the form of an adaptive production system.

Learning takes place in a wide variety of situations and probably by a number of different processes. This article proposes a theory of the specific processes that enable a student to learn while engaged in solving a problem. No claim is made that the theory embraces all possible kinds of learning by doing. Instead, the theory focuses on giving a sufficiently microscopic account of learning in a specific situation to guarantee that the mechanisms proposed by the theory are adequate to that situation. The guarantee is buttressed by expressing the theory as a computer program that when confronted with a problem situation, will work on the problem and learn while it works. The issue of the generality of the program and its component mechanisms are addressed later in the article.

The theory we propose had its origins in our detailed analysis of a single human problem-solving protocol of about 1½ hours, which provides very specific evidence that one particular person, at least, did learn by using processes closely resembling those that we have incorporated into our computer program. As part of our exposition of the theory, therefore, we give our analysis of this protocol. Finally, we comment on how specific or general the resulting theory is.

The Learning Task

Problem-solving tasks of any complexity allow considerable leeway in solution strategy because, generally, considerable numbers of alternative strategies are available, all of which are adequate for reaching a solution (Amarel, 1968; Bruner, Goodnow, & Austin, 1956; Katona, 1940; Newell & Simon, 1972; Simon, 1975). For this reason, solution strategy becomes an important intervening variable in human behavior in problem-solving situations.

This is not to say that all strategies are equivalent. Some may be far more efficient than others in terms of speed in arriving at the solution, in terms of the load they place on short-term memory, in terms of the ease of retaining them, or in terms of the range of tasks to which they can be transferred. Since the most efficient strategies for a particular task may not be the most obvious ones to someone encountering the task for the first time, we might expect a person to employ a sequence of strategies as he or she gains skill in performing the task. Initially, the solver

This research was supported by Research Grant MH-07722 from the National Institute of Mental Health and by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-73-C-0074), which is monitored by the Air Force Office of Scientific Research.

Yuichiro Anzai is now at the Faculty of Engineering, Keio University, 3-14-1, Hiyosho, Japan.

Requests for reprints should be sent to Herbert A. Simon, Department of Psychology, Carnegie-Mellon University, Schenley Park, Pittsburgh, Pennsylvania 15213.

 0033-295X/79/0000-0000$00.75

might hit on one of the "obvious" strategies, and then gradually progress to more efficient ones with increasing familiarity with the problem domain.

If such progressions take place, they have great importance for understanding how problem-solving skill is acquired. To explain the learning process, we have to show how each of these problem-solving strategies emerges from the one that just preceded it. If the strategies themselves are expressed as programs—production systems, say—we have to explain how one such production system is transformed into another. A complete explanation would presumably take the form of an adaptive production system capable of carrying out the sequence of transformations (Waterman, 1975).

The Tower of Hanoi puzzle provides a convenient task domain in which to study the strategy transformation process. A number of the different strategies that have been identified for solving this puzzle have been described as production systems (Simon, 1975). Hence, we have a good foundation for identifying the strategies used by subjects, and we know that a number of reasonable alternative strategies are available. However, there do not appear to be any published studies that throw light on the processes that subjects use to acquire or transform these strategies.

The present article addresses itself to these transformation phenomena. Its empirical base is an exceptionally rich protocol of a subject who was previously unfamiliar with the puzzle and who developed a sophisticated strategy for solving the puzzle over a period of about $1\frac{1}{2}$ hours. During that time, the subject tried solving the five-disk Tower of Hanoi puzzle four times from the initial disk configuration. On each trial, she used a strategy that was transformed from, and more efficient than, the strategy she used on the previous trial.

The subject's protocol is analyzed to provide a detailed picture of the transformation process—in particular, the sequence of cues she noticed in the problem structure and the use she made of these cues in order to modify her strategy. We first programmed each of her four strategies as a production system so that by comparing these systems, we could see exactly what changes in the program were involved in moving from one strategy to another. After examining the protocol evidence as to how the transformations came about, we next describe an adaptive production system that has the ability to create each new strategy from the preceding one. This adaptive production system is offered as a theory of the basic learning capabilities that the subject exhibited and the basic mechanisms that she used to accomplish the successive improvements in strategy that her protocol exhibited.

Method

Subject and Design

The experiment was carried out in a single session that lasted for about $1\frac{1}{2}$ hours. The subject was an adult female liberal arts college graduate whose native language is Japanese. Prior to the experiment, she had seen the Tower of Hanoi apparatus but had had no experience in trying to solve the puzzle.

The subject was instructed to think aloud while solving the problem, and the protocol of her verbalizations was recorded on tape. An English translation of the problem instructions (including the description of the problem) is reproduced here, and a literal English translation of the complete transcript of the subject's protocol is presented in the appendix of this article. The protocol is divided into 224 statements numbered consecutively from S1 to S224. The starting peg for the problem is Peg A, and the goal peg is Peg C. The disks are numbered from 1 (smallest) to 5 (largest).

The instructions for the thinking aloud protocol for the 5-disk problem were as follows:

> There are three pegs on the board, which are named Peg A, Peg B, and Peg C from the left to the right. There are five disks of different size on Peg A with the configuration that each disk lies above the disks bigger than it is.
>
> Your task is to transfer those disks to Peg C by moving one disk at a time to one of the pegs other than the peg on which the disk lies, and by following the rules: (a) You can not put a disk on any disk smaller than it is, and (b) you can not move a disk on which another disk lies.
>
> The purpose of this experiment is to analyze what and how you think when you solve the problem. Thus, I would like you to tell aloud whatever you think during the solving process. If you think that your solving process would not lead to a good solution procedure, you may give up that process and start from the initial situation. I hope that you can find a good solution procedure for the problem.

The last two sentences of the problem instructions encouraged the subject to find not merely a solution for the problem but a good solution procedure. In pursuance of this instruction, the subject solved the problem three times following one unsuccessful solution attempt. As we shall see, in each successive solution attempt, she used a different strategy.

The shortest solution path for the five-disk Tower of Hanoi problem is 31 moves. The subject made only about three incorrect moves during her first two successful solution attempts, and she corrected all of these immediately. She made no errors during the third successful attempt.

Results

Analysis of the Protocol

The 224 protocol statements can be divided into four episodes (E1–E4), corresponding to the subject's four solution attempts. The first, unsuccessful, attempt (E1), is recorded in S1–S23. Statements S24–S74 record E2, which terminated with the first successful solution of the problem. In E3, which contains S75–S162, she achieved a second solution using a new strategy; and in E4, which contains S163–S224, she achieved a third solution using yet another strategy.

Of the 224 protocol statements, 96 are bare statements announcing a move that the subject was making. An additional 14 statements announce a move but also comment on the reason for it. The remaining 114 statements, a little more than half of the protocol, fall into three categories: (a) specific means–ends reasoning to calculate the correct next move (49 statements), (b) metastatements about the solution method (32 statements), and (c) an explicit analysis of the recursive structure of the problem at the beginning of E3 (33 statements, S75–S107). In our analysis of the protocol, we shall be particularly interested in the statements belonging to the last two categories.

The protocol is so explicit throughout as to leave little or no uncertainty about the strategy that the subject was employing at each moment. On the other hand, there is much less explicit information in the protocol to disclose the subject's learning process—how she acquired each new piece of information that permitted her to alter her strategy adaptively, or just how the adaptation occurred.

The first episode. From the very beginning, the subject's approach to the problem was far from random. Her strategy during E1 may be described as a search strategy, but a highly selective one. The number of legal moves at each step in the Tower of Hanoi problem was very small, only two or three. At no time did the subject reverse a move just made or move the same disk twice in succession (this would always be inefficient, since the same result could be gained in a single move), although she never mentioned these restrictions. These restrictions reduce the number of legal moves at each step to one (when the smallest disk has just been moved) or two (when the smallest disk is to be moved). With these restrictions, also, the smallest disk will be transferred on odd-numbered moves.

On no occasion during E1 did the subject, on successive odd-numbered moves, transfer the smallest disk and then return it to its previous peg (see S4). This further restriction completely eliminates any choice beyond the initial move—that is, the restrictions taken together constitute a complete strategy, except for the choice of the very first move. It can be seen that the subject's search was very selective.

The subject had a reason (S4, S6) for her initial move, too: She preferred not to move the smallest disk to the ultimate target peg, since the larger disks would have to go to that peg first. This decision, though motivated, happened to be the wrong one for the five-disk problem, accounting for her failure on the first attempt.

The subject pursued her first attempt through eight moves, at which point she noticed that she was moving the fourth disk to block access of the largest disk to the target peg (S12–13). After persevering a little longer (S14–19), she decided to return to the original configuration, ending the first episode.

The strategy that the subject followed during E1 (we cannot tell how consciously or explicitly) will be referred to as the *selective search strategy*. It eliminates all further search after the initial move has been selected.

There is some indication, however, that the subject did not follow the selective search

strategy in pure form. At some points in the protocol, she mentions goals and moves taken in order to accomplish goals (See S8, S13). The fact that she decided she was on the wrong track when she saw that the largest disk could move to Peg B if she continued (S13) is perhaps the clearest evidence that she was forming some goals. Most of the subject's pauses occurred just before she moved the smallest peg (S10, S30, S32, S39). This suggests that she was not executing the strategy automatically, but was engaging in some kind of reasoning about the moves—perhaps already using part of the goal–peg strategy that emerged clearly in the second episode.

From the first episode, the subject retained in long-term memory at least two pieces of information: (a) On the first move, the smallest disk should be transferred to the goal peg and not to peg B, as in her first attempt (S26); and (b) the largest disk should be moved to the goal peg (S48, S72).

The second episode. The second episode started with the correct move, remembered from E1. More precisely, what was remembered (S25–S26) is that the smallest disk (Disk 1) must be moved to the target (Peg C) so that the largest disk (Disk 5) might eventually reach that peg.

In the second episode, unlike the first, the subject guided herself explicitly by mentioning intermediate goals: to move Disk 4 to Peg B (S34), to move Disk 5 to Peg C (S48), to move Disk 4 to Peg C (S59), and to move Disk 3 to Peg C (S63). She summarized this strategy of moving first the largest and then the successively smaller disks to C in S72–S74. We refer to this as the *goal-peg strategy*. The subject did not make explicit how she calculated back from these goals to her next moves, but the principal pauses in the protocol occurred just after each such goal has been achieved, when the plan for achieving the next had to be formulated.

The main information that the subject appeared to retain from the second episode and to bring to the third episode is that the transfer of the biggest remaining disk to the goal peg is a good subgoal (S73–S74).

The third episode. Before undertaking to solve the five-disk problem again, the subject tested her understanding by going through the moves of the one-disk problem, the two-disk problem, the three-disk problem, and the four-disk problem (S75–S107), respectively. In the course of this rehearsal, she explicitly formulated and applied the recursive rule that guided her through the solution of the five-disk problem in E3. A clear example of an application of the rule is found in S82–S84: "Oh yes, 3 will have to go to C first; for that, 2 will have to go to B; for that, um . . . , 1 will go to C." Other examples are found at S78, S86–S87, S99–S101, S121–S123, S130–S132, S140–S143, and S153–S155. The subject summed up this procedure in S162.

In E3, therefore, the subject found the idea of a recursive goal calculation. That is, to achieve a goal (G1), we need to achieve a subgoal (G2); to achieve G2, we must achieve another subgoal (G3); and so on. She arrived inductively at this strategy, which we will call the *recursive subgoal strategy,* by working upwards from the simplest one- and two-disk problems. (See especially S78–S86.)

The fourth episode. The subject was evidently satisfied with the recursive goal strategy and only solved the problem a third time because the experimenter asked her to (S162–S163). She appeared to bring no new information from the third episode but began to apply her strategy exactly as before. Nevertheless, a subtle but significant change began to take place in her solution process. She began, for the first time, to speak of *sets* of disks and to set goals of transferring such sets. In S179, she said, "I only need move three blocking disks to B." In S193, she said "I will move the remaining four from B to C." She then followed with the important comment (S194): "It's just like moving four, isn't it?" From this point on, she repeatedly referred to goals of moving sets of disks (S196, S208, S210, S211, S216, and S223).

We will call this final strategy that the subject attained, which involves solving the whole Tower of Hanoi problem by solving smaller problems of exactly the same kind, the *pyramid subgoal strategy*. The new strategy appears to have been arrived at less systematically—certainly with less awareness—than the goal recursion strategy. Perhaps it would

not be misleading to call the learning process here perceptual. The subject learned to view the goals in a new way—requiring not the transfer of a succession of disks but the transfer of a pyramid of disks. Unfortunately, in this task environment as in others that have been studied, the verbal protocols give us only the slightest hints of the perceptual processes and the perceptual learning that may be going on.

Summary. We have now accounted in some detail for the content of the subject's protocol and for the gradual transformation of her strategy. In summing up, we may claim the following:

1. With varying degrees of awareness, the subject definitely generated at least four successive strategies.

2. Information stored in long-term memory in the previous trial appears to provide the basic cues for strategy transformation, except in the last case in which perceptual cues may have played the main role.

3. Short-term memory appears to be used mainly to store subgoals during the course of problem solution. Other information is retrieved from long-term memory or by direct perception of the problem situation. Rehearsal occurs only when information gathered in the previous episodes is required in the current one. Examples of information obtained from perception are the set of admissible moves (e.g., S4, S26) and noticing the disk next smaller to a given disk among the disks on some peg (e.g., S82–S83).

4. During each episode, we see evidence of at least four kinds of processes: (a) applying the current strategy, (b) gathering information that will later be used to modify the strategy, (c) using information gathering in previous episodes, and (d) deciding to terminate the solution attempt (successfully or unsuccessfully.

5. The most deliberate (explicitly mentioned) changes in strategy took place between the first and second episodes and between the second and third episodes. Less conscious (unverbalized) strategy changes took place during the first and the fourth episodes. Perhaps this difference provides further evidence of a plurality of learning processes—cognitive and perceptual.

Simulation of Protocol by a Fixed Production System

To gain a deeper understanding of the subject's learning processes, we modeled her behavior in two stages in detail. First we built a production system that embodied the four strategies that she had used in successive trials and showed that we could model most of the fine detail of her behavior in each trial. This production system did not, however, model the learning process. Essentially, it contained a "big switch" that shifted from one strategy to the next at points corresponding approximately to the points in the protocol where the same shift occurred for the subject.

The initial production system was compartmentalized into substructures, each representing one of the strategies and consisting of from six to nine productions, including productions that gathered information for later use (i.e., for switching strategies) and productions that used information gathered in previous episodes. Apart from the information-gathering and information-using productions, the actual strategies each consisted of five or six productions. Some characteristics of the structure and output of the system are summarized below:

1. The five main strategic modules (corresponding to search, selective search, goal, peg, recursive subgoal, and pyramid subgoal strategies) share a common local structure. Roughly, productions for gathering information for later use come first, those for using information gathered in previous episodes come last, and those for applying the current strategy are placed in-between. The productions in a strategic module are scanned from top to bottom. In the first stages of executing a strategic module, productions near the bottom tend to fire, rather than those near the top. Hence, each of the observed strategies begins by utilizing information gathered in the previous episode, then proceeds to execute the algorithm, and finally collects information for use by subsequent modules.

2. All five main modules except the first include the following subset of productions:

1. (GOAL X P Q) (LEGAL X P Q)
→(REMOVE (GOAL X P Q)) (MOVE X P Q)
2. (GOAL X P Q) (ON (NEXTSMALLER X P)P)
→(GOAL (NEXTSMALLER X P) P (OTHER P Q))
3. (GOAL X P Q) (ON (NEXTSMALLER X Q) Q)
→(GOAL (NEXTSMALLER X Q) Q (OTHER P Q))

Production 1 means that if the subgoal is to transfer Disk X from Peg P to Peg Q and the move is admissible, then the subgoal is removed from short-term memory and the move is performed. Production 2 means that if the subgoal is to transfer X from P to Q but there lies on Peg P the disk next smaller to X, then the new subgoal of moving the next smaller disk from P to the peg other than P and Q is set up. Production 3 is the same as 2, except that the peg of the next smaller disk is Q, the target peg of the current goal.

Productions 1–3 construct a recursive sequence of subgoals that try to attain the current subgoal. The existence of these productions in most of the main modules implies that recursive subgoal calculation plays an important role in all the strategies. Only the initial stage of selective search is an exception, so that the main productions in the first two strategic modules have a form quite different from Productions 1–3.

3. Even though Productions 1–3 are embedded in most of the strategic modules, they are not used in the same way in the successive episodes. In the second strategic module, in which they appear for the first time, they are used only for the disks smaller than Disk 4. In the third module, which simulates the goal–peg strategy, they are used for the disks bigger than Disk 3, and then their use is generalized to all disks so that in the fourth recursive subgoal strategy module they are fully used and form the main part of this module. Finally, in the pyramid goal module, their use is again restricted to smaller disks. Bigger disks are transferred using the recursive subgoals of moving subpyramids.

Thus, the main trend of strategy transformation can be diagrammed simply as:

selective research → recursive calculation for smaller disks → recursive calculation for all disks → recursive calculation for larger sets of disks.

4. Important information, 14 items in all, stored and recalled through long-term memory is tabulated in Table 1. The table shows that such information (except Item 1, which asserts that the top goal is to move all disks from Peg A to Peg C) is recalled only in the episode immediately following the episode in which it was stored. Item 1, stored before the solving process starts, is recalled mostly in the earlier episodes and rarely in the recursive subgoal or pyramid subgoal modules.

As a consequence, the strategy transformation process is globally simple, for new strategies are dependent only on the strategies that immediately precede them. It must not be presumed, however, that information acquired in the preceding module is the sole basis for learning the recursive strategies. It is also necessary to assume that the subject had a number of kinds of procedural knowledge prior to entering into the experiment—knowledge, for example, that could lead to the selective search in Trial 1 and the recursive calculation in Trial 2.

5. The recursive subgoal strategy and pyramid subgoal strategy used by the subject bear a close similarity to the goal-recursion and inner–directed goal-recursion strategies analyzed formally by Simon (1975), thus providing some empirical support for his theoretical exploration of the range of strategies available to human subjects in this task environment.

Although the initial production system did not provide a simulation of the learning process, it did give us a minute description of each of the strategies that had to be learned and allowed us to detect in the protocol much of the detail of information that the subject fixated in long-term memory and carried over from one trial to the next to be used for learning the next strategy. We turn next to some observations of her learning processes and, particularly, on the prior knowledge about

Table 1
Information Stored and Recalled Through Long-Term Memory

Episode	Corresponding strategy[a]	Information stored[a]	Information recalled (item no.)[a]
E1	PROBLEM–INSTRUCTION SELECTIVE SEARCH	1. (TOPGOAL MOVE ALL FROM PEGA TO PEGC)	
		2. (MOVE 1 FROM PEGA TO PEGC)	1
		3. (MOVE 4 OFF PEGA)	1
		4. (MOVE 5 FROM PEGA TO PEGC)	1
E2	GOAL PEG	5. (SUBGOAL: MOVE 5 TO PEGC)	2, 4, 3, 1
		6. (SUBGOAL: MOVE 4 TO PEGC)	1
		7. (SUBGOAL: MOVE 3 TO PEGC)	1
		8. (SUBGOAL: MOVE 2 TO PEGC)	1
		9. (SUBGOAL: MOVE 1 TO PEGC)	1
E3	RECURSIVE SUBGOAL	10. (REMEMBER: BLOCK 3)	5–9
		11. (MOVE PYRAMID 4 TO PEGC)	5
E4	PYRAMID SUBGOAL	12. (REMEMBER: SMALL BLOCK 2)	10
			1

[a] Inferred from the protocol; the table shows at approximately what points in the protocol the information was initially noted and subsequently recalled and applied.

possible strategies that she brought to the learning task.

Knowledge Accumulation in Learning

In order to form new and more effective strategies, the subject extracted knowledge about the problem structure from her solution attempts. Her ability to do this, however, depended fundamentally on her having already available in long-term memory some sophisticated learning capabilities and some prior knowledge of possible types of strategies (e.g., means–ends analysis). From her protocol, we can infer at least five important ways in which prior knowledge was combined with new information gathered while solving the problem to contribute to the learning process.

Knowledge From Selective Search

Acquisition of essential subgoals was not at all trivial for the subject. As a preliminary step, the search space had to be simplified on the basis of knowledge permitting the abandonment of possible but uninteresting moves (e.g., moves that returned to previously attained positions). In the Tower of Hanoi puzzle, after these repetitive moves are eliminated, only a single admissible move remains at each choice point after the first. The requisite prior knowledge is simply that move repetitions are inefficient and avoidable. Using this knowledge requires remembering the most recent moves and the next-to-most-recent moves.

With the elimination of move repetitions, and as a consequence, the elimination of branching in the search tree, it becomes much easier for the problem solver to look ahead (i.e., to imagine a sequence of future moves without actually carrying them out), since no information about subgoals or the like now needs to be kept in short-term memory, and hence memory capacity is available for stor-

ing anticipated moves. The looking-ahead procedure plausibly leads to the generation of subgoals.

Organization of Subgoals

If a subgoal of moving a bigger disk to the goal peg is remembered, the subtree of the coal tree that contains the subgoal as its root can be detached from the original tree and handled independently of its larger context. Thus, it is important to have prior knowledge of this kind of means–ends analysis. Remembering good subgoals helps encode the hierarchical organization of the original structure and reduces the load on short-term memory.

Even if such encoding succeeds, the complicated subgoal structure still needs to be organized for encoding a strategy in efficient form. The subgoal hierarchy formed by moving Disk X to the goal peg (X = 5, 4, 3, 2, 1) is such an organization of subgoals constructed by the subject. It seems to have been first conceived when the subject encountered the problem instructions. (S6 in the protocol provides some evidence.) But fixation of this organization within a particular strategy had to wait for the subject's experience of applying these subgoals (e.g., S73).

As mentioned in the previous section, the kernels of the main strategic modules all have similar structures, which similarity may be derived from the subject's prior general knowledge of means-ends analysis and her selective, looking-ahead search in the first episode. These kernels facilitate generating a sequence of subgoals, from future to present, looking backwards. After they are generated, moves are made to attain the subgoals consecutively starting with the immediate one.

Chunking Moves

Successive moves without pauses of Disks 1 and 2 or Disks 1, 2, and 3 occurred frequently. A sequence of moves made without a pause and without a reference to subgoals may come to be regarded as a single chunk. These chunks are of great help for encoding a strategy, for they save effort in making decisions about the moves of the smaller disks. The subject's ability to form chunks facilitated her learning overall strategies.

The protocol shows that the concept of three-disk chunks was not yet formed in the earlier part of E3 (see, e.g., S114–S126). The first evidence for it appeared in the middle stage of E3 (S130–S138, S144–S151). The concept of two-disk chunk moves became apparent also in E3 (S114–S117, S124–S126, S132–S134, S136–S138). In these instances, the moving of Disk 3 was still driven by a subgoal. It was after the concept of blocking disks was discovered (S179) that the concept of 3-disk chunk moves was completely fixated (S180–S186). (S197–S207 still used a subgoal for Disk 3.)

Concepts of a Goal Stack and Moving a Set of disks

The subject's discovery of the pyramid subgroup strategy (S193–S194) may have been inspired by discovery of the concept of a set of blocking disks (S179), also a chunking ability. The concept of moving a set of disks was apparently discovered perceptually. Any subpyramid has a pyramidlike shape similar to the initial (and goal) configuration of disks. But this perceptual discovery might not have occurred if the subject had not obtained the concept of a set of blocking disks.

To obtain the pyramid subgoal strategy, it is crucial to discover the concept of a goal stack and the concept of retaining several subgoals in the stack at one time. This can be done by replacing a current subgoal with a sequence of subgoals functionally equivalent to it. Chunking subgoals should make this equivalence transformation easier. Evidently, the chunking of subgoals required a long learning process, as did the chunking of moves, because the pyramid subgoal strategy was acquired only toward the end of the whole problem-solving sequence.

Perception in Learning the Strategies

The production system program is based on the ability of the system to make three basic perceptual discriminations: (a) that two pegs are distinct, (b) that one disk is larger than

another, and (c) that a particular disk is on a particular peg. Of these, only the third is time dependent, changing as successive moves are made. Some verbalizations from the protocol indicating perceptual processing are: "no place else" (S4), "open" (S37, S191) and "bottom" (S78). These traces are related directly to the third class of perceptual tests.

Analysis of the program structure indicates that most of the information-gathering productions include a perceptual primitive of the third class on the condition side, whereas the information-utilizing productions, with one exception, do not include such a primitive. Moreover, every information-gathering production includes a subgoal condition or a condition referring to information stored in long-term memory. From these data, we may infer that the information-using process is usually a conceptual process, whereas the information-gathering process involves cooperation of conceptual and perceptual processing.

An Adaptive Production System

On the basis of our analysis of how the subject went about devising new strategies, we proceeded to build an adaptive production system that uses similar processes to progress from an initial primitive search strategy to a strategy that avoids repetition of moves, a means–ends strategy, and an inner-directed recursive strategy. Although the system does not simulate in detail the actual temporal course of the subject's learning, it demonstrably uses the same kinds of information that the subject used and arrives at three of the strategies that she generated. The structure of the adaptive production system and the representation it employs are discussed from an artificial intelligence viewpoint in Anzai (in press), and computational results are described in detail in Anzai (1978).

Basic Assumptions and Procedure

Before describing the adaptive production system in detail, it will be helpful to outline the basic assumptions about learning that it incorporates.

1. At the heart of learning is the ability of the system to acquire knowledge about the effectiveness of its choices of moves (knowledge of results) and to use that knowledge to modify itself. The knowledge of results is of two kinds: (a) recognition that a move or sequence of moves has had unfavorable consequences and (b) recognition that a move or sequence of moves has improved the situation. More specifically, under a, the system is able to recognize that it has moved the same disk twice or that it has returned to a situation that it had already reached a move or two earlier. A double move by one piece and a move sequence that causes "looping" can always be improved. Under b, the system is able to recognize when it has achieved a subgoal.

2. When the system has recognized a bad or good outcome of a sequence of moves, it is able to create one or more new productions and insert them to modify its behavior in a direction that will tend to avoid bad consequences and to enhance the likelihood of good ones. To do this, it must reason backwards from the bad or good outcome and find a pattern of preceding moves that can be interpreted as having caused the outcome.

To simplify the structure of the production system and, in particular, to make it easier to analyze its behavior, the full range of a priori knowledge that it needed to acquire the three strategies was not given to it at the outset (Table 2). Instead, in Run 1 it was given enough prior knowledge to enable it to learn to avoid repetitions of moves; then its behavior while using the productions it had learned was tested in Run 2, without additional learning. In Run 3, it was given the prior knowledge that it needed to acquire productions that use means–ends analysis. Again, its performance using the newly acquired productions was tested in Run 4 without additional learning. In Run 5, it was given the prior knowledge that it needed in order to make use of the idea of pyramids as chunks and to incorporate new productions based on this idea. Finally, its performance was tested again in Run 6, after this last piece of learning had been accomplished. Hence, the system learned during the odd-numbered runs, and its

Table 2
Results of Computer Simulations and Their Relation to Protocol Data

Run	Productions executed	Strategy	Learning	Corresponding episode	Corresponding protocol segment
1	914	→ selective move	yes	1	1– 24
2	656	selective move	no	2	
3	371	→ recursive subgoal	yes	2	25– 70
4	280	recursive subgoal	no	3 (part)	108–162
5	431	→ pyramid subgoal	yes	4	163–224
6	296	pyramid subgoal	no	—	—

learning was tested during the even-numbered runs.

Effort Measurements

Run 1 of the program was roughly equivalent to Episode 1 in the subject's thinking-aloud protocol. Runs 2 and 3 were roughly equivalent to Episode 2 and the first portion of Episode 3. Run 4 was roughly equivalent to the remainder of Episode 3, and Run 5 was roughly equivalent to Episode 4. Nothing in the protocol corresponded to Run 6. As a crude measure of the improvement in problem-solving performance produced by the learning, we may compare the number of productions that had to be executed in each run to solve the problem. These numbers are displayed in Table 2. Looking first at the even-numbered runs, we see that the recursive subgoal strategy required less than two fifths as many production executions as the selective search strategy with avoidance of repetition (280 as compared with 656). On the other hand, from Runs 4 and 6, we see that the pyramid goal strategy required the execution of slightly more productions than the recursive subgoal strategy (296 as compared with 280).

Although the general implications of these numbers are clear, we should not assume that the number of productions executed in carrying out a strategy is exactly proportional to the time or energy a human subject would expend in carrying out the same strategy. In particular, the chunks of moves created by the pyramid goal strategy might be executed by humans in compiled rather than interpreted fashion and with much decreased effort because compilation reduces the number of symbols that must be held in short-term memory during execution of the program. Hence, for humans, the pyramid goal strategy may be significantly more efficient, in time and in memory load, than the recursive subgoal strategy, even though this was not the case for our production system.

From the comparison of the corresponding odd-numbered runs with even-numbered runs, we get some measure of the additional effort that was required to solve the problem while learning. Although it went only halfway to the solution, Run 1 required nearly half again as many executions as Run 2 (914–656); Run 3, a third more than Run 4 (371–280); and Run 5, a third more than Run 6 (431–296). Not all of this difference can be attributed to the cost of the learning processes themselves because during the early portions of the learning (odd-numbered) runs, the production system was still using the previous, usually less efficient, solution method. However, the learning processes require a substantial expenditure of effort, but by no means an excessive expenditure compared with the gains they produce in the efficiency of performance.

Task Independence of Prior Knowledge

Let us look next at the specific learning strategies used in Runs 1, 3, and 5 and, particularly, at the prior knowledge that is embedded, implicitly or explicitly, in these strategies. From "prior knowledge" we exclude, of course, knowledge that the subject might have gained either from reading the problem instructions or during the course of her problem-solving efforts.

Here a small apology is in order. In the version of the program actually implemented, we did not succeed in making a complete separation of these two kinds of knowledge. Hence, some of the productions in the learning part of the system contain information about the representation of the Tower of Hanoi problem. As a result of this programming exercise, we now know how to make the boundary sharper and the separation cleaner between the program's general learning capabilities, which make no reference to the task environment of the Tower of Hanoi and its knowledge of the way in which that particular task environment is represented. We will describe the learning program in this cleaned-up form, and hence as a task-independent general learning system, comparable in this respect to the general problem solver.

The system begins with capabilities for selective search: It can *make* (legal) moves, and it can *consider* moves without necessarily making them. Its path through the problem space can be represented as an alternation of *states* or *positions* (in the Tower of Hanoi puzzle, particular arrangements of disks on pegs) and *actions* (moves, applications of operators). Each action leads to a new position, and each position permits a new action (from the set of legal actions for that position) to be taken. All learning that we will consider is based on retaining in memory and using information about a segment of the search path.

Avoiding Bad Moves (Selective Search)

The method of avoiding bad moves uses the following schema: Let P1, P2, and P3 be successive positions along a search path and let A1 and A2 be the actions that take P1 into P2 and P2 into P3, respectively. Then, if P1 = P3 (the two positions are identical), create a production of the form:

(PAST-MOVE A1) → (EXCLUDE A2)

and insert it in the program so that it will be executed while moves are being considered and before they are made. Notice that the subsystem that creates this production need have no knowledge of the internal structures of states or moves of the task representation. It simply refers to them by their labels or names.

Now, with a little more sophistication, we can go a step further. Suppose A1 is represented in the Tower of Hanoi task as

(MOVE 1 C B)

("Move disk 1 from peg C to peg B")

and A2 as

(MOVE 1 B C).

Then a *generalization process* can replace the constants in these actions by symbols that denote variables of the appropriate types—making the same replacements in both A1 and A2—and thus producing

(PAST-MOVE X P Q) →
(EXCLUDE (MOVE X Q P)).

Clearly, this generalization requires no knowledge of the representation other than an ability to identify the types of the arguments of the two functions (i.e., the types of the symbols 1, B, and C).

Similarly, if the system can notice that the path (P1 A1 P2 A2 P3) is longer than (P1 A3 P3) but reaches the same position, it can learn to avoid the longer path. Suppose A1 = (MOVE 1 A B), A2 = (MOVE 1 B C), A3 = (MOVE 1 A C). If the system notices that DISK(A1) = DISK(A2), the desired production, suitably generalized, is

(PAST-MOVE X P Q) →
(EXCLUDE (MOVE X Q R)).

These particular pieces of learning are based on the idea that it is inefficient to visit the same state twice or to move the same disk twice in a row and on the ability to detect identity between two states or two disks (the latter abilities obviously being task dependent). Variants of the same idea are possible. Suppose the system had both the (task-independent) knowledge that it was undesirable to return the same object to the same location on two successive moves of that object (without necessarily restoring the entire state) and the

(task-dependent) knowledge of which arguments in the move operator designated objects and which ones designated positions. The latter knowledge would permit it to perform the test, INVERSE A1 A2, to discover that MOVE 3 A C followed by MOVE 3 C A creates a loop in this generalized sense. Then such loops could be avoided by creating the production

(PAST-PAST-MOVE A1) (INVERSE A1 A2) →
(EXCLUDE A2).

Again, if the production were created for specific moves, A1 and A2, these could be generalized by application of the generalization operator, exactly as before.

The capabilities we have just described are essentially the capabilities (although in less elegant form) that the adaptive production system employed in Run 1 to create new productions for avoiding bad moves, and the productions we have described are equivalent to those that were created.

Goal Recursive Strategy

When the system has learned to avoid bad moves, its search is narrowed. If it has acquired the full selective search strategy, it even becomes linear. It is then in a position to learn other strategies.

To acquire a goal recursive strategy, the system must first be able to posit goals. Our implementation assumes that the subject is able from the problem instructions to infer goals such as: GOAL I FROM A, where I = 1, 2, 3, 4, 5, and GOAL I TO C, where I = 5, 4, 3, 2, 1. Once the system has such goals and has used its selective search strategy to reach one or more of them, it can use a priori knowledge to create new recursive subgoals by means of the following strategy.

Suppose that the system in state P1 has GOAL A2 and discovers, by search, the successful path segment P1, A1, P2, A2 (to be read: "P1 is operated on by A1 to produce P2, which is operated on by A2"). When A2 has been reached in this segment, GOAL A2 has been attained. Hence, it is known that applying A1 to P1 creates a new state, P2, in which A2 can be applied. Therefore, in P1, GOAL A1 can be taken as a subgoal to GOAL A2 and a task-independent learning system could create a production such as

(CURRENT-STATE P1) (GOAL A2) →
(SET (GOAL) (GOAL A1)).

Of course this production is too specific to be very useful because it would be evoked only if the system were precisely in state P1. Let us see how we might generalize it using the system's knowledge of the requirements for a legal move in this task environment. Let the tests for the legality of move A be represented by T1(A), T2(A), and T3(A). All of these tests are satisfied for A2 in P2 because A2 was made in that situation. They were not all satisfied for A2 in P1 because although the system then had the goal of taking action A2, it in fact took action A1. As the condition for our production, instead of specifying the entire state P1, we simply provide a list of the tests that are and are not satisfied to obtain the modified production

(CURRENT-TESTS T1 T2 NOT (T3))
(GOAL A2) → (SET (GOAL A1)).

Again, this production is readily generalized by replacing all constants by variables of the same type. However, the full generalization requires the system to notice appropriate relations between the components of A1 on the one hand and of A2 and the tests on the other—for example, to notice that the disk of A1 in the above production is the next smaller to the disk of A2. This too is task-dependent knowledge, requiring a sufficiently powerful perceptual system for acquiring it.

The new productions created during Run 3 were of the kind that we have just described. As can be seen from their form, given both a goal and a situation, they create a subgoal that is appropriate for progressing toward the goal in that situation.

Chunking Moves

We have seen that our subject learned to move her three smallest disks as a single chunk. This chunk corresponds to the produc-

tion

(GOAL 3 X Y) → (MOVE 1 X Y) (MOVE 2 X Z)
(MOVE 1 Y Z) (MOVE 3 X Y).

We did not implement the learning of this production in our simulation, but it would proceed in much the same way as the learning of the pyramid subgoal chunks, which we will discuss next.

Pyramid Subgoal Strategy

Since the term *pyramid* is not task independent but refers to a particular arrangement of disks in the Tower of Hanoi puzzle, a procedure for learning the pyramid subgoal strategy cannot be quite as general as the learning procedures described previously. Nevertheless, we can describe it in a form that will clearly separate the task-dependent from the task-independent components.

Suppose the system has (task-dependent) knowledge that leads it to notice the feature F(P), in any state P where that feature is present. Suppose that in a segment of the solution path, the feature F is present at the beginning of the segment and the feature F* is present at the end. Suppose further that Goal F* is present at the beginning of the segment. And suppose, finally, that the sequence of actions in this segment is A1, A2, A3. Then, it is a simple matter to create the production

(CURRENT-FEATURE F) (GOAL F*) →
(MOVE A1 A2 A3).

This production can be generalized in the same manner as the others by replacing all constants by variables of the same type. All of the task-dependent information that it requires is embedded in the productions that recognize the feature F and the productions that know enough about the types of arguments to be found in goals and moves to replace constants by appropriate variables.

Discussion

The adaptive production system described in the previous section would have only limited interest for psychology if it merely described a set of idiosyncratic learning processes used by a single subject on one occasion to improve her performance in a problem-solving task. We make a stronger claim for the system: It provides a psychological theory of some generality explaining how a person can learn in the course of performing a task of this general kind.

It may be objected that a general psychological theory cannot be supported by a single case. One swallow does not make a summer, but one swallow does prove the existence of swallows. And careful dissection of even one swallow may provide a great deal of reliable information about swallow anatomy. Although the generality of the theory we have constructed remains to be tested, we undertook to model accurately the learning mechanisms that we observed in the behavior of our single human subject, modeling them in such a way that their applicability would not be limited to the specific task environment, the Tower of Hanoi puzzle, in which we discovered them. As we have seen, the key mechanisms are completely independent of this particular task and fully applicable to other problem-solving environments in which heuristic search can be employed. In this sense, the theory is generalizable over subjects and tasks.

Knowledge of results obtained through search of the problem space provided the foundation for all of the learning. If the search led to bad results (e.g., repetition), new productions were formed that detected the circumstances in which repetition had occurred and avoided the offending moves. When good results were attained (e.g., a known subgoal reached), new productions were formed that could make the appropriate preparatory moves when it was noticed that the subgoal was nearly within reach. When it was noticed that a brief sequence of moves always attained a subgoal of a given kind, a production was formed to execute this sequence of moves whenever the subgoal was evoked (chunking). These are all general processes that make no reference to the Tower of Hanoi or any other specific task environment.

One way to characterize learning processes of this sort is to observe that if a person can

solve a problem by any method—however inefficient or crude—then the correct solution path can be used as a template on which to form new productions capable of discovering the solution more efficiently. David Neves (1978) has used a similar idea to construct an adaptive production system that learns from examples that have already been worked out (i.e., in a textbook). Given a single step–by–step example of the process of solving a linear algebra equation, Neves's system constructs an algorithm with fairly general capabilities for solving such equations. The idea underlying the learning program of Neves and the program presented here is that knowledge of the correct solution path provides information not only about the steps that have to be taken to follow that path but also about the cues present in the successive situations reached that indicate what next step may be appropriate. The new productions generated by the adaptive system link the cues with their actions.

Finally, there is some empirical evidence that other naive subjects confronted with the Tower of Hanoi problem will have learning experiences similar to those of our subject. Six subjects have been observed by Andy Kahn (Note 1) to achieve a subgoal strategy in much the same way as our subject did. Although five of Kahn's subjects then went on, like ours, to attain the pyramid subgoal strategy, one gradually shifted from the simple subgoal strategy to the move pattern strategy described in Simon (1975).

Conclusion

One particular subject's protocol in solving the Tower of Hanoi puzzle was analyzed in some detail to study strategy transformation processes in problem-solving tasks. The subject successively generated four different strategies with varying degrees of awareness.

The first strategy involved selective forward search for a proper move while avoiding repetitions of moves. The second used a mixed subgoal hierarchy. The remaining two strategies incorporated recursive subgoals. The latter of these two involves the concepts of a goal stack and moving a set of disks. The three strategies that use subgoals look backward to construct subgoals and then generate actual moves. Thus, the strategy transformation proceeded from looking ahead to looking backward.

Learning the strategies seemed to involve both conceptual and perceptual information. Particularly, information gathered in the immediately preceding episode played a main role for generating a new strategy. In this sense, the learning process had, globally, a simple sequential structure.

An initial production system was constructed to simulate the subject's execution of the strategies that she used in successive trials. Careful comparison of the behavior of the production system with the subject's protocol provided hints as to the information she used to learn each new strategy in the context of the preceding one. This information was of at least three different kinds: (a) task-dependent information extracted from the task instructions, (b) task-dependent information obtained by observation in the course of solving the problem, and (c) task-independent prior information about classes of possible strategies.

A second, adaptive, production system was then constructed that incorporated a set of learning capabilities. Presented with the Tower of Hanoi task, this system learned three of the four strategies that the subject acquired, requiring a portion of a trial, as the subject did, to learn each new strategy. This simulation gave a precise specification of both the task-independent prior knowledge and the task-dependent information that is required for such learning.

We have indicated how the adaptive production system we have described, although derived from the evidence of a single problem-solving protocol, is constructed from mechanisms of some generality. It is proposed here as a theoretical model of the processes that people use to learn in the course of solving problems—to learn by doing.

Reference Note

1. Kahn, Andy. Personal communication, April 15, 1978.

References

Amarel, S. On representations of problems of reasoning about actions. In D. Michie (Ed.), *Artificial intelligence 3*. New York: American Elsevier, 1968.

Anzai, Y. How one learns strategies: Processes and representation of strategy acquisition. *Proceedings of the 3rd Conference on Artificial Intelligence and Simulation of Behavior,* Hamburg, West Germany/ Gesellschaft fur Informatik, ~~in press.~~ 1978a,

Anzai, Y. Learning strategies by computer. *Proceedings of the 2nd Conference on Computational Studies of Intelligence*. Toronto: Canadian Society for Computational Studies of Intelligence, 1978.

Bruner, J. S., Goodnow, J., & Austin, G. *A study of thinking*. New York: Wiley, 1956.

Katona, G. *Organizing and memorizing*. New York: Columbia University Press, 1940.

Neves, D. M. A computer program that learns algebraic procedures by examining examples and by working test problems in a textbook. *Proceedings of the 2nd Conference on Computational Studies of Intelligence*. Toronto: Canadian Society for Computational Studies of Intelligence, 1978.

Newell, A., & Simon, H. A. *Human problem solving*. Englewood Cliffs, N.J.: Prentice-Hall, 1972.

Simon, H. A. The functional equivalence of problem-solving skills. *Cognitive Psychology,* 1975, *7,* 268–288.

Waterman, D. A. Adaptive production systems. *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*. Cambridge, Mass.: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1975.

Appendix

Protocol of the Subject

1. I'm not sure, but first I'll take 1 from A and place it on B.
2. And I'll take 2 from A and place it on C.
3. And then, I take 1 from B and place it on C. (If you can, tell me why you placed it there.)
4. Because there was no place else to go, I had to place 1 from B to C.
5. Then, next, I placed 3 from A to B.
6. Well . . . , first I had to place 1 to B, because I had to move all disks to C. I wasn't too sure though.
7. I thought that it would be a problem if I placed 1 on C rather than B.
8. Now I want to place 2 on top of 3, so I'll place 1 on A.
9. Then I'll take 2 from C, and place it on B.
10. And I'll take 1 and . . . place it from A to B.
11. So then, 4 will go from A to C.
12. And then . . . , um . . . , oh . . . , um . . . ,
13. I should have placed 5 on C. But that will take time. I'll take 1 . . .

(If you want to, you can start over again. If you are going to do that, tell me why.)

14. But I'll stay with this a little more . . .
15. I'll take 1 from B and place it on A.
16. Then I'll take 2 from B to C.
17. Oh, this won't do . . .
18. I'll take 2 and place it from C to B again.
19. And then, I'll take 1, and from A . . .
20. Oh no! If I do it this way, it won't work!
21. I'll return it.
22. Ok?
23. I'll start over.

(Go ahead.)

24. If I go on like this, I won't be able to do it, so I'll start over again.
25. Let's see . . . I don't think 5 will move.
26. Therefore, since 1 is the only disk I can move, and last I moved it to B, I'll put it on C this time . . . from A to C.
27. So naturally, 2 will have to go from A to B.
28. And this time too, I'll place 1 from C to B.
29. I'll place 3 from A to C.
30. And so I'll place 1 from B . . . to C.
31. Oh, yeah! I have to place it on C.
32. Disk 2 . . . no, not 2, but I placed 1 from B to C . . . Right?
33. Oh, I'll place 1 from B to A.

(Go ahead.)

34. Because . . . I want 4 on B, and if I had placed 1 on C from B, it wouldn't have been able to move.
35. 2 will go from B to C.
36. 1 will go from A to C.
37. And so, B will be open, and 4 will go from A to B.
38. So then, this time . . . It's coming out pretty well . . .
39. 1 will . . . 1 will go from C . . . to B.
40. So then 2, from C, will go to . . . A . . .
41. And then, 1 will go from B to A.
42. And then, 3 will go from C to B.
43. 1 will go from A to C.
44. What?

45. And then, 2 will go from A to B.
46. And then, oh, it's getting there.
47. 1 will go from C to B.
48. So then, 5 will finally move from A to C.
49. And then, 1 will go from B to A.
50. Oh, I'll put 1 from B to C.

(Why?)

51. Because if 1 goes from B to A, 2 will go from B to C . . .
52. Let's try it again, ok?
53. Um . . . it's hard, isn't it?
54. I didn't know it would be so hard . . . It's hard for me to remember . . .
55. And so I guess I have to do it logically and systematically.
56. And 1 will go from A to C.
57. 3 will go from B to A.
58. 1 will go from C to B.
59. Because I want to move 4 to C, and to do that I have to move 2, don't I?
60. And to do that, 2 will go from C to A.
61. And then, I will go from B to A.
62. And then, 4 will go from B to C.
63. This time, if I think of 3 on C, that will be good, so 1 will go from A to C.
64. 2 will go from A to B.
65. 1 will go from C to B.
66. And then, I'll bring 3 from A to C.
67. This time it's easy, and 1 will go from B to A.
68. 2 will go from B to C.
69. And then 1 will go from A to C.
70. All right, I made it.
71. I wonder if I've found something new.
72. I don't know for sure, and little ones will have to go on top of big ones . . . big ones can't go on top of little ones, so first, bit by bit, C will be used often before 5 gets there.
73. And then, if 5 went to C, next I have to think of it as 4 to go to C . . .
74. This is my way of doing it . . .
75. Can I move it like this?
76. First, if I think of it as only one disk, 1 could go from A to C, right?
77. But, if you think of it as two disks, this will certainly go as 1 from A to B and 2 from A to C, then 1 from B to C.
78. That . . . that anyway 2 will have to go to the bottom of C, naturally I thought of 1 going to B.
79. So, if there were three . . . , yes, yes, now it gets difficult.
80. Yes, it's not that easy . . .
81. . . . this time, 1 will . . .
82. Oh, yes, 3 will have to go to C first.
83. For that, 2 will have to go to B.
84. For that, um . . . , 1 will go to C.
85. So, 1 will go from A to C, 2 will go from A to B, 1 will go back from C to B, I'll move 3 . . . That's the way it is!
86. So, if there were four disks, this time, 3 will have to go to B, right?
87. For that, 2 will have to stay at C, and then, for that, 1 will be at B.
88. So 1 will go to B.
89. And then, 2 will go from A to C.
90. And then, 1 will go back to C from B.
91. And then, 3 will move from A to B.
92. And then, I will move 1 from C to A.
93. And then, first, I will move 2 from C to B.
94. And then, I will move 1 from A to B,
95. and then, 4 from A to C.
96. And then, again this will go from A . . . 1 will . . .
97. Wrong . . . , this is the problem and . . .
98. 1 will go from B to C . . .
99. For that, um . . . , this time 3 from B, um . . . , has to go on C, so . . .
100. For that, 2 has to go to A.
101. For that, 1 has to go back to C, of course.
102. And then, 2 will go from B to A,
103. and then, 1 will go from C to A,
104. and then, 3 will go from B to C.
105. So then, 1 will go from A to B.
106. 2 will go from A to C,
107. and then, 1 will go from B to C.

(All right.)

108. I think I can do five now.
109. . . . Ah, it's interesting . . .
110. If it were five, of course, 5 will have to go to C, right?
111. So, 4 will be at B.
112. 3 will be at C.
113. 2 will be at B.
114. So 1 will go from A to C.

(Fantastic.)

115. This is the way I think!!
116. And then, 2 will go from A to B.
117. 1 will go back from B to C.
118. 3 will go from A to C.
119. For that, um . . ., this time, again . . . , as this time 4 will have to go to B . . .
120. Let's move back 1 from B to A . . .
121. If 4 has to go from A to B, it means . . .
122. 2 will have to go to 3.
123. Because 1 will . . .
124. So, 1 will go back from B to A.
125. And then, 2 will go from B to C.
126. And then, 1 from A to C.
127. And then, 4 from A to C.
128. And then, this time . . . , it's the same as before, I think . . . , um . . .
129. Of course, 5 will go to C, right?
130. For that, 3 will have to go to B, so

131. 2 will go back to A,
132. 1 from C to B.
133. 2 from C to A.
134. 1 from B to A.
135. 3 from C to B.
136. 1 from A to C.
137. 2 from A to B.
138. 1 from C to B.
139. And then, finally 5 will go from A to C,
140. and then, this time, um . . . , um, 4 will go to C, so . . .
141. 3 goes to A,
142. 2 goes to B,
143. and then, 1 will go to A.
144. So, anyway, I will move 1 from B to A.
145. 2 from B to C.
146. And then, 1 from A to C.
147. 3 from B to A.
148. 1 from C to B.
149. And then, 2 from C to A.
150. And then, 3 from B . . .
151. 1 from B to A.
152. And then, finally, I have succeeded in moving 4 from B to C.
153. So, this time, um . . . , oh, this time, 3 naturally has to go here, so,
154. for that, 2 has to go to B.
155. So 1 will go from A to C,
156. place 2 from A to B,
157. place 1 from C to B,
158. and then, 3 from A to C.
159. Place 1 from B to A,
160. place 2 from B to C,
161. and then, move 1 from A to C.
162. Oh, yeah . . . , In this way, think bit by bit . . . , think back . . .

(Ok, why don't you try again?)

163. After all, it's the same thing, isn't it?
164. First 1 will go from A to C.
165. Because, 5, at the end, will go to C, so,
166. So, 4 will go to B.
167. And then, 3 will go to C.
168. And then, 2 will go to B.
169. So, 1 will go from A to C.
170. 2 will go from A to B.
171. Move 1 from C to B,
172. move 3 from A to C.
173. Next, 4 will go to B. So . . .
174. move 1 from B to A,
175. move 2 from B to C.
176. Move 1 from A to C,
177. and then, move 4 to B.
178. Next, 5 has to go to C, so . . .
179. I only need move three blocking disks to . . . B.
180. So, first . . . 1 will go from C to B,
181. move 2 from C to A,
182. and then, move 1 from B to A.
183. Move 3 from C to B,
184. 1 from A to C.
185. Move 2 from A to B,
186. and then, 1 from C to B.
187. And then, 5 can go to C . . .
188. It's easy, isn't it?
189. 5 has already gone to C.
190. Next . . . , 5 was able to move, because . . .
191. A and C were open, right?
192. 5 is already at C, so . . .
193. I will move the remaining four from B to C . . .
194. It's just like moving four, isn't it?
195. So . . . I will have to move 4 from B to C . . .
196. For that, the three that are on top have to go from B to A . . .
197. Oh, yeah, 3 goes from B to A!
198. For that, 2 has to go from B to C,
199. for that, 1 has to go from B to A.
200. So, 1 will go from B to A.
201. 2 goes from B to C.
202. 1 will go from A to C.
203. And then, 3 can go from B to A.
204. Then, it'll be good if 1 and 2 go to A, so.
205. . . . first 1 goes from C to B,
206. 2 moves from C to A.
207. And then, 1 moves from B to A.
208. Um, with this, the three at B have moved to A, so . . .
209. move 4 from B to C.
210. Next, if the three at A go to C, I will be done.
211. So first, the top two disks will be moved to B.
212. For that, 1 goes from A to C.
213. 2 goes from A to B.
214. And then, 1 goes from C to B,
215. and then, 3 goes from A to C.
216. Oh! This time, the two on B will be moved to C.
217. Right . . .
218. 1 moves from B to A,
219. 2 from B to C.
220. And then, 1 will move from A to C.
221. I did it!
222. I think I finally got it . . .
223. This time, if 5 goes to C, it'll be just like moving four . . . , but . . .
224. I still don't quite yet . . .

Received July 12, 1978
Revision received October 10, 1978 ■