3 Information-Processing Explanations of Understanding

#507
In P.W. Jusczyk & R.M. Klein (Eds.), The nature of thought: Essays in honor of D.O. Hebb. Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

Herbert A. Simon
Carnegie-Mellon University

The explanation of human mental processes that Donald O. Hebb provided some 30 years ago in his *Organization of Behavior* (1949) is now a permanent part of the conceptual equipment of psychology. The cell assembly is as familiar a construct as the reflex. In rereading the book, one is immediately struck again with the insight and imagination of its author in building a coherent picture of mind from a mass of complex and confusing evidence. What may be less evident—unless one has a long memory—are the reasons why *Organization of Behavior* produced such a violent shock wave when it impacted on the hard shell of a behaviorism reluctant to look inside the human head, much less populate it with structures as rich as cell assemblies. The courage of the author in rejecting the dominant S–R psychology and reintroducing central mental processes is quite as remarkable as his psychological insight.

That shock wave continues to reverberate through psychology. We are no longer satisfied with external descriptions of regularities that say nothing about the mechanisms producing them. We want to formulate and test, however indirectly, what is going on inside. I don't know how sympathetic Donald Hebb is with the particular form of explanation that today is known as *information-processing psychology*. Whatever his sympathy or lack of it, his courage in exploring the mechanisms of mind must bear a large share of the responsibility for setting others on this path. Without the example of that courage, few would be foolhardy enough to investigate phenomena as intricate and cloudy as the phenomena of human understanding.

"Understanding" is one of those global terms referring to processes of the human mind whose importance we acknowledge even as its meaning remains vague and ambiguous. No one—certainly no practicing teacher—doubts for a moment that there is a fundamental difference between learning through understanding and learning by rote and that the former is infinitely to be preferred to the latter. There is even a modest body of empirical evidence (Katona, 1940) indicating that material learned through understanding is retained longer and transferred more readily to new tasks than material learned by rote. If the distinction merits the significance we attach to it, then we need to provide it with some operational foundations and to explore the mechanisms that enable understanding to arise.

SOME MEANINGS OF "UNDERSTANDING"

In a recent survey of artificial intelligence approaches to understanding (Simon, 1977), I undertook to provide some definitions of the term, building on an earlier proposal by Moore and Newell (1974). Just as intelligence is defined by specifying tasks the intelligent person is able to perform ("Intelligence is what intelligence tests measure"), so understanding is defined by specifying tasks the understanding person can perform. Let me enumerate what I think are some of the relevant tasks.

Understanding, like intelligence, has many facets: We can speak of someone understanding knowledge or of someone understanding a task. In the first case, following Moore and Newell (1974), we can say that "*S* understands knowledge *K* if *S* uses *K* whenever appropriate." Notice that this definition distinguishes clearly between *having* knowledge and *understanding* it. I may know (i.e., be able to repeat on request) the definition, "the sine of an angle is the side opposite over the hypotenuse," without being able to answer the question, "What is the sine of 45 degrees?". It is precisely this kind of difference we are trying to get at when we distinguish between rote learning and learning through understanding.

Similarly, we would not say that someone understands the game of chess simply because that individual can recite the rules for a legal move. We would want to test further to make sure the individual can actually apply the rules in a chess situation. We may say that *S* understands task *T* if *S* has the knowledge and procedures needed to perform *T*.

Some important ambiguities remain even in these definitions. One may know how ("understand how") to play chess without knowing how to play it well. In fact, we commonly make a rather strong distinction between understanding a problem and being able to solve it. There is a celebrated mathematical hypothesis, Goldbach's conjecture, that any even number can be expressed as a sum of exactly two prime numbers. It is not at all difficult to understand Goldbach's conjecture—that is, to know what it would mean for it to be true or false. However, no one has yet been able, after a century of effort, to prove whether the conjecture is true, or false, or undecidable.

Understanding how to perform a task may or may not entail understanding the principles underlying task performance. Most automobile drivers have only the vaguest idea of how a clutch operates or how carburetor and spark plugs control the injection and ignition of fuel. We presently examine an example of alternative solution algorithms for solving a problem that correspond to different understandings of the problem structure. There are many other nuances to the term "understanding" that I cannot go into here.[1] My main goal is to illustrate, with concrete examples, how processes that mediate understanding, in one or the other application of that term, are represented in information-processing models of human thinking. In the next section, I sketch out the modeling strategy in general terms. In the two succeeding sections I describe in greater detail two examples of information-processing systems that simulate human understanding.

MODELING UNDERSTANDING PROCESSES

The basic strategy for modeling understanding processes is identical with the general strategy for computer simulation of human thinking. A task is selected, successful performance of which is regarded as evidence of some particular form of understanding. On the basis of any knowledge we may have of how people perform this task, together with our general knowledge of human cognitive processes, we construct a computer program capable of performing the task we have selected, and of doing so without using any computer capabilities—rapid arithmetic operations or unlimited short-term memory—that we know people do not possess. After testing the program's performance on the understanding task, we carry out various comparisons with data (e.g., latencies, thinking-aloud data, error statistics) obtained from human performance of the same task. If we regard the computer program as a theory, which it is, this is simply an application of the general procedure for testing theories that is used throughout science. Let us look at a few examples.

One of the earliest understanding programs was described in Robert Lindsay's 1961 doctoral dissertation (Lindsay, 1963), which undertook to understand Basic English in the context of making inferences about kinship relations. Given such sentences as "John is Mary's father" and "Alice is John's sister," Lindsay's system could infer and report that "Alice is Mary's aunt" and "Mary is Alice's niece." This act of understanding required two basic abilities: the ability to carry out syntactical analyses of simple English prose, and the ability to carry out inferences based on lexical definitions like "a parent's sister is an aunt." The demonstrated success of Lindsay's system in performing the language-processing and inference task allows us to hypothe-

[1]For a more elaborate taxonomy of forms of understanding, see Simon (1977).

size that it is precisely the possession of these abilities that enables people to exhibit understanding in these kinds of tasks. Lindsay's system was, of course, only a first step toward the construction of more flexible, general, and powerful language-understanding schemes with inference-making capabilities, but it illustrates how computer models can illuminate the nature of the human processes.[2]

Other aspects of understanding are illustrated by Winograd's well-known SHRDLU program (Winograd, 1972). Understanding "cat" involves more than having a lexical definition for the term. A "real" understanding of what the word means requires the ability to recognize a cat when one appears on the scene. Any system that understands words having real-world referents must have intentional definitions of these words in the form of tests that can be applied to real-world objects.

SHRDLU has this capability in principle, although the "real world" on which it operates is in fact a simulated Block World that exists in a computer memory. The Block World contains blocks of various shapes and colors resting on a table or on each other. SHRDLU is able both to answer questions about the Block World (e.g., "Is there a blue cone standing on a yellow cube?") and to obey instructions that involve moving blocks (e.g., "Place the small cube on the large cylinder."). Executing such commands requires not only an ability to understand what is meant by them, but also a problem-solving ability for finding a sequence of actions that will actually carry them out. The problem-solving component of SHRDLU uses forms of means–ends reasoning very similar to those embodied in the General Problem Solver (Newell & Simon, 1972).[3] By storing the reasons for its sequences of actions, SHRDLU acquires another kind of understanding: the ability to explain why it did certain things. It can carry on a conversation like this: "Why did you move the green pyramid?"; "In order to clear the large cylinder?"; "Why did you want to clear the large cylinder?"; "In order to place the small cube on it."; "And why did you want to do that?"; "Because you told me to."

A major product of the first two decades' research on systems that understand in various task environments, including the systems described briefly above, was to identify a small set of mechanisms that showed up again and again as essential components of understanding systems. The first of

[2]Anderson & Bower (1973) contains an extensive survey of langauge-understanding systems that have been constructed in the intervening years. See also Siklóssy & Simon (1972).

[3]In brief, means–ends reasoning involves comparing the present state of the system with the desired goal state, detecting one or more differences between them, recovering from long-term memory one or more operators that are relevant to reducing differences of the kinds that were detected, and applying one of the operators. The process is then repeated until all differences between present state and goals have been eliminated. For a fuller discussion of this and other terms from information-processing psychology, see Newell and Simon (1972).

these are mechanisms for syntactical parsing of natural language. The second are mechanisms for mapping input information into canonical representations that can be stored in semantic memory. (Lindsay's system, for example, stored its information in family trees.) The third are mechanisms for inferring implied meanings by exploiting the relations among several input sentences, including mechanisms for doing means-ends reasoning. The fourth are matching mechanisms for assessing schemas of semantic information stored in long-term memory and relating these to new information provided explicitly in input strings. The fifth are mechanisms that can recognize instances of concepts and classify objects in terms of those concepts. The sixth are mechanisms for generating objects that satisfy specified relations. The seventh are mechanisms for finding reasons for the system's own actions.

This is certainly not a complete inventory of the range of mechanisms that will be required to simulate human understanding in all of the task environments where it manifests itself. What is shows, however, is that of the substantial number of "understanding" programs that have been constructed, each one is usually capable of operating in only a very restricted range of task environments, and all rest on a common base of quite general processes.

I have referred to these programs as simulations of *human* understanding. Many of them were written, of course, not with the specific goal of simulating human processes, but with the aim of exploring and extending the capabilities of computers for intelligent action. In spite of that, I believe that most of the mechanisms they use are essential also to human intelligence. In the two examples I discuss in the following sections, we have a certain amount of empirical evidence that this is so.

THE *UNDERSTAND* PROGRAM

In an earlier section of this chapter, Goldbach's conjecture was used to illustrate the distinction between understanding a problem and understanding how to solve it. This is simply one instance of the vast number of situations we encounter where we have to understand a set of instructions expressed in natural language before we can carry them out. More homely and practical examples are the instructions on a can of soup or medicine bottle, or instructions for filling out an income tax form.

In these situations, and others like them, we are engaged in a two-stage process: first to extract the meanings of the written instructions, and then to use our problem-solving abilities to follow those instructions. There are two classes of such situations (really a continuum, but I consider the extreme possibilities). On the one hand, the instructions may be entirely self-contained, so that, once we understand them, we possess all the knowledge that we need in order to solve them. In that case, we may conceive of the second stage as being carried out by some kind of general problem-solving process that incorporates one or more general procedure for attempting to solve problems but no specific knowledge or information about any particular problem domain. We call problems that, once understood, can be attempted by a general problem solver, "puzzlelike" problems. Most of the problem-solving tasks that have been studied in the psychological laboratory fall in this category: e.g., the Tower of Hanoi problem, the Missionaries and Cannibals problem, water-jug problems, and so on.

The second class of problems are those that, even after they are understood, require domain-specific knowledge as well as general problem-solving capabilities for their solution. For example, solving a textbook problem in physics stated in natural language may require the ability to understand not only general English-language text, but also the specific meanings of physical terms like "acceleration" and "lever," and knowledge of some physical laws—the law of mechanical advantage for levers or the law of falling bodies—that govern the behavior of physical systems. We call problems that require specific knowledge for their solution, "semantically rich" problems.

The UNDERSTAND program (Hayes & Simon, 1974) was designed to simulate the processes for understanding puzzlelike problems. UNDERSTAND takes the written instructions in natural langauge as its input and produces as output the information that would be needed, in turn, as input to the General Problem Solver (GPS) so that the latter program could go to work on the problem (without guarantees, of course, that it would be successful in solving it).

The input for GPS that UNDERSTAND produces consist of: (1) a representation for problem states,[4] which permits a description of the objects appearing in the problem instructions and their relations; and (2) routines for legal move operators that can transform one problem state into another. We claim that the program understands because it uses its knowledge appropriately to construct the representation and the routines.

What knowledge does UNDERSTAND call on in its performance? Its only external source is the set of written problem instructions. Its internal sources, embodied in the program, include: (1) a natural language parser; (2) capabilities for transforming the parsed text into an internal list structure representation of objects and relations; (3) stored programs capable of modifying list structures in a variety of ways and of making tests on list structures; (4) capabilities for mapping the legal move descriptions in the instructions on appropriate moves and tests selected from among these stored programs; and (5) capabilities for executing the stored programs and tests

[4]Problem states are situations reachable by applying "legal move" operators from the initial problem situation.

interpretively so that they operate correctly on the particular representations constructed by the processes that transform the parsed text.

I must digress to explain some of the computer science terminology I have used in this description and to motivate its use in the context of simulation of human thought processes. In particular, what do "list structures" have to do with human memory? A list is simply an ordered set of elements—which could be used, for example, to represent a simple chain of associations. A list structure is an assemblage of lists—which could be used to represent an assemblage of associations (more precisely, of *directed* associations, in the Wuerzburg sense of the term). A more complete explanation of the use of list structures to represent an associational memory may be found in Newell and Simon (1972, Chapter 2), or in Anderson and Bower (1973).

To account for storage of new information in such an associational memory, for the modification of existing information, or for the accessing of information, only a few basic processes need be postulated, and—most important—these processes depend only on the structure of memory, and not on its content. They include, for example, processes for adding an item to a list, for deleting an item from a list, for copying an item from one list to another, for comparing two items for identity, and the like. Once represented as list structures, all problems look alike to such a system—that is to say, they all can be attacked with processes that are indifferent to real-world content, and responsive only to the organization of the abstracted list structures. The "stored programs" referred to are simple combinations of these basic processes.

Consider the familiar puzzle of moving three missionaries and three cannibals across a river, with the conditions that they must be rowed across in a boat holding at most two persons and that missionaries must never be left alone with a larger number of cannibals. The list-structure description of this problem might consist of a list of missionaries and cannibals on the left side of the river, a list of those on the right side of the river, and a list of those in the boat. A "move" operator would transfer members of these lists from one location (list) to another, as the boat is loaded, moved across the river, and unloaded. Another simple operator would compare the relative number of missionaries and cannibals on each list. Of course, the system would need no knowledge of the meanings of "missonary" and "cannibal"—they could as well be labeled "hobbits" and "orcs." Nor would "boat" and "river" have any but abstract properties, unrelated to real-world knowledge about boats and rivers.

UNDERSTAND parses the natural-language text of the problem instructions and discovers in the parsed text the classes of objects that are being talked about ("missionaries," "cannibals") and the properties and relations of those objects (location, number). By examining the formal properties of the legal moves described in the instructions (the number and types of arguments), it assembles from the basic operators stored in its memory a legal move operator (in this case, an operator for transferring objects from one list to another, and checking that the legal move conditions are satisfied). This is essentially all the information that GPS needs in order to conduct a heuristic search for a problem solution.

The UNDERSTAND program is demonstrably capable of understanding the instructions for simple, puzzlelike problems. J. R. Hayes and I (Hayes & Simon, 1974; Hayes & Simon, 1977; Simon & Hayes, 1976) have carried out several empirical studies to compare the program's behavior with that of human subjects faced with the same task. Human thinking-aloud protocols show generally good correspondence with the processes postulated by the program. Moreover, the program predicts correctly the alterations that can be induced in human subjects' problem representations by manipulating the text of the problem instructions. I think it fair to say that, on the basis of the available evidence, UNDERSTAND provides a good first-order theory of human understanding processes in abstract problem environments.

Some progress has been made, also in simulating the understanding process for semantically rich domains, but much remains to be done. The central difficulty, of course, is that in understanding such problems, people make use of the knowledge of the domain they already possess, and therefore that knowledge must be represented in the memory of the problem-solving system. This means that we must make a reasonably complete inventory of such knowledge for a particular domain before we can build an understanding system for that domain.

Such knowledge bases have now been constructed, for example, for some restricted domains of chemistry (Buchanan & Lederberg, 1971) and for broad areas of medical diagnosis (Pople, Meyers, & Miller, 1975; Shortliffe, 1976), but, in these cases, for the purpose of constructing artificial intelligence systems, and without concern for detailed simulation of human processes. However, several knowledge bases have also been constructed for limited domains of physics that simulate in various ways the organization of human knowledge in these same domains. I mention just one of them.

Novak (1976) has constructed an ISAAC system that can understand and solve physics word problems in elementary statics, relating to levers and the equilibrium of weights. The system's domain-specific knowledge is stored in *schemas* that describe common components of such systems: schemas for pivots, for levers, for surfaces with and without friction, for weights, and so on. Concrete objects (ladders, walls, men standing on ladders) are abstracted to the corresponding appropriate schemas. A ladder is a lever, a wall is a surface, and so on. When a problem is presented to ISAAC, it is analyzed in terms of the elementary schemas represented in it and their relations. A new complex schema is built up in memory to represent the whole physical situation of the problem. This schema is used in turn to specify the equations that govern the behavior of the system, and, finally, the equations are solved.

Similar systems have been built for simple problems in kinematics (Simon & Simon, 1978), dynamics (de Kleer, 1977), and chemical engineering thermodynamics (Bhaskar & Simon, 1977). They have been compared with human behavior to a limited extent only but show considerable initial promise of casting light on the human processes.

UNDERSTANDING PROBLEM SOLUTIONS

Space does not permit a survey of all the meanings of "understanding"—the range of tasks to which knowledge can be applied. Consider, for example, the HEARSAY speech recognition system (Reddy & Newell, 1974). Its task is to produce a translation of a sound stream into written language, a task that would not appear to require understanding of the meaning of the sound stream at all. Attempts to build speech-understanding systems have taught us, however, that it is very difficult to decode a sound stream without bringing to bear on it all the syntactic and semantic information we can marshal. No one has been able to show how simple phonemic encoding and word recognition can be made to work without the use of rich syntactic and semantic information. As a consequence, although understanding meanings is not one of the direct goals of HEARSAY, the program makes extensive use of semantics and indeed requires virtually all the kinds of mechanisms I have described as constituting an understanding system.

Chess-playing programs illustrate a rather different point. Here, depth of understanding might be measured by quality of play. Yet existing programs achieve a given quality of play by a whole continuum of methods ranging from large-scale brute-force search of all legal moves to some depth, to highly selective search making use of feature recognition and other knowledge-interpreting devices. Looking at these programs, we might decide that strength of play does not capture what we usually mean by "understanding" and that the programs that play more selectively somehow understand more deeply the situations in which they find themselves than do the programs that rely on brute-force search.

Every experienced teacher makes a similar distinction between a student's depth of understanding and a student's ability to perform on particular tests. The teacher knows that some students learn by rote whereas others learn through understanding and that this is an important difference even if it does not show up in the grades on the Friday quiz. How can we characterize the distinction, and what is its significance?

Returning from humans to computers for a moment, there is a temptation to say that everything a computer knows it knows by rote, for "it can only do what it is programmed to do." The vast differences among chess programs noted previously should warn us, however, against equating "doing what the program says" with "doing by rote." In any event, the question deserves a deeper examination.

In order to have a simpler environment than chess in which to explore the issue, let us consider the Tower of Hanoi puzzle (Simon, 1975). This puzzle consists of three vertical pegs and a number of doughnutlike disks of different sizes, which are initially stacked in a pyramid on one of the pegs, say *A*. The task is to move all the disks to another peg, say *C*, under the constraints that (a) only one disk may be moved at a time, and (b) a disk may never be placed on top of another smaller than itself. The minimum number of moves for a solution of the *n*-disk problem is 2^n-1.

Three examples illustrate the wide variety of relatively simple programs that solve the Tower of Hanoi problem. Moreover, these different programs exhibit quite different kinds of understanding of the problems. I think these programs are simple enough that I can describe their fundamental characteristics in English, without getting into programming formalisms.

The first possibility is simply a *rote* solution. The correct sequence of moves is stored in a list in memory. A simple interpreter retrieves the moves from memory in the order of storage and executes each in turn. When the list is exhausted, it halts, with the problem solved.

To execute the rote strategy, only a single place-keeping symbol need be retained in short-term memory, and no perceptual tests need be made in order to determine what move to make next. The reasons for regarding this as a rote strategy are obvious: No analysis is involved, but simply recall of a list of items in order; the solution only works for a specified number of disks; and it only works for a specified initial problem situation and a particular goal situation. Hence, the program has no capability for transfering what it knows about the problem to variants involving a different number of disks, or even trivially different starting and goal points.

In sharp contrast to the rote program is a program incorporating the usual *recursive solution* for the Tower of Hanoi problem. This solution rests on the observation that in order to move a pyramid of *n* disks from Peg *S* to Peg *G*, one need only: (1) move a pyramid of (n-1) disks from *S* to Peg *O;* (2) move the largest remaining disk from *S* to *G;* and (3) move the pyramid of (n-1) disks from *O* to *G*. To execute this solution, however, the program must be able to store in short-term memory, in a sequence corresponding to their order in the goal hierarchy, all the goals that at the given time have not yet been completely executed. The maximum number of goals that have to be held simultaneously in short-term memory is ($2n$-1) for an *n*-disk problem. Hence, it is unlikely that a human subject would be able to carry out this strategy unmodified form with more than three or four disks.

From the standpoint of understanding, however, the important difference between the rote and recursive solutions of the problem is that the latter, but not the former, possesses and makes use of implicit knowledge of important aspects of the problem structure. This additional knowledge is knowledge that the problem is recursive—that the *n*-disk problem can be dissected into problems of exactly the same form, but with fewer disks. Notice that "possession" of knowledge here means that it is built into the program, hence available for solving the problem, not that it is available in explicit declarative form. A subject having and using such a program may, or may not, be able to state the recursive rule.

The recursive strategy is knowledgeable about the Tower of Hanoi problem and can solve problems of any number of disks, and with any starting point and number of goal pegs. It must start from the beginning, however, and if it loses its place (loses the stack of subgoals), it cannot recover. It is very introverted, paying no attention to the current arrangement of disks on pegs but deciding what to do next solely on the basis of which goal lies uppermost in its goal stack in STM. It can easily be fooled by an outside intervention that changes the disk arrangement between two of its moves and is then not able to continue.

There is a very different strategy, the *perceptual strategy,* that makes considerable use of its knowledge of the current problem situation in order to decide on its next move. The perceptual strategy first notices the largest disk that has not yet reached the goal peg. It determines whether this disk can be moved legally to the goal. If not, it notices the largest disk that is blocking the move and determines the peg to which this latter disk must be moved to get it out of the way. Taking the move of the blocking disk as its new subgoal, the strategy now repeats the same cycle: checking the legality of the move, noticing the largest blocking disk if it is not legal, and setting up the goal of moving that disk. Eventually, the strategy finds a move that can be made and makes it. Now it starts at the beginning again and repeats.

The perceptual strategy solves not only the problem of moving an *n*-disk pyramid from one peg to another, but it does this starting from any arbitrary (legal) arrangement of the disks. Moreover, it need retain in short-term memory a maximum of only two symbols, no matter how large *n* is, instead of the entire goals stack required by the recursive strategy.

This difference in performance capabilities derives from differences among the means the various strategies have available for conceptualizing the problem. In the recursive method, the successive moves are governed by the goal structure in short-term memory; in the perceptual method, they are governed by the current state of the stimulus. The rote strategy knows only of individual legal moves. The recursive strategy employs the concept of a macromove: moving a subpyramid of disks. The perceptual strategy lacks this latter concept, but it has the notion of "the largest disk blocking the move of disk X," is able to determine perceptually which disk this is, and can use this knowledge to progress toward its goal.

Even this simple puzzle environment reveals additional dimensions to the meaning of understanding. The understanding that different subjects, who have learned to use different strategies to solve the Tower of Hanoi problem, have acquired is quite different in each case. Each has touched a different part of the elephant.

UNDERSTANDING AND AWARENESS

Let us return again from computer programs to human behavior. In order to test a student's understanding of the Tower of Hanoi problem:

1. We could test the speed with which the student learned a solution strategy, how well the student retained it over time, and to what range of similar tasks the student could transfer it. Research by Katona (1940) has shown that "meaningful" solutions of problems are learned faster, retained better, and transferred more effectively than rote solutions.
2. We could infer the nature of the strategy the student used to solve the problem and evaluate it with respect to its parsimony and generality, just as we have been evaluating the computer programs.
3. We could measure the student's ability to evoke the strategy when it was appropriate for solving a problem. (This is the criterion that comes closest to fitting our initial definition of understanding.)
4. We could test the student's ability to paraphrase or describe the strategy in alternative expressions.
5. We could test the student's ability to explain or prove how or why the strategy works.

The final two tests in this list raise entirely new issues beyond those discussed up to this point—issues of what might be called "awareness." But why are we interested in a student's ability to paraphrase, explain, or prove? And how can we give an information-processing explanation of being aware in these latter senses? We take up the second question first.

Because the three strategies have not been described in formal detail but have already been paraphrased in the text, we can take those paraphrases as examples. Note that they are expressed in natural language; they omit much of the detail of the formal strategies; and they are in a form that a program having the general characteristics of UNDERSTAND might use as input for generating the programs themselves. Stated otherwise, a paraphrase of a program corresponds closely to an informal description of the top levels of that program. Hence, it might serve as input to an understanding program. A system that could paraphrase programs and create programs from paraphrases would have ways of understanding its knowledge that go beyond those previously described. It is not hard to think of contexts in which people appear to use this kind of understanding (e.g., in learning to perform tasks from brief English-language descriptions).

Explaining a program or proving its efficacy (the fifth of the tests listed earlier) is similar to paraphrasing but goes a step further in paying attention to the performability of each step and to the effectiveness of the procedure in attaining its goal. Knowledge in the form of paraphrase, explanation, or proof of strategies is knowledge about knowledge, which is how I define awareness. To the extent that a system can examine its own programs, analyze them, and modify them, it has additional forms of knowledge that have their own appropriate uses. It appears plausible that these uses are especially closely related to the system's capacity for learning, and for acquiring new forms of understanding.

CONCLUSION

There appears to be no end of "understanding" problems. Research in artificial intelligence and information-processing psychology has identified and attacked at least a half-dozen such problems during the past decade. Understanding was initially viewed in this research as the ability to extract meanings from natural language inputs and to make explicit by inference information contained implicitly in those inputs. At a later stage, emphasis shifted to capabilities for combining information derived from multiple sources (two or more input sources, or input information with information already available in semantic memory). Just appearing on the horizon, and likely candidates for the next decade's research, are questions about the forms of understanding called "awareness."

In surveying a considerable range of tasks calling for understanding, we have found a much smaller variety of basic mechanisms for performing these tasks. This is encouraging and comforting if we have, as most of us do, a taste for parsimony in explanation. However, it should not make us complacent in thinking that we have already discovered the main mechanisms required for all forms and types of understanding of which people are capable. The understanding of the process of understanding continues to be an important, even central, research target in cognitive psychology today.

ACKNOWLEDGMENT

This research was supported by Research Grant MH-07722 from the National Institute of Mental Health.

REFERENCES

Anderson, J. R., & Bower, G. H. *Human associative memory,* Washington, D.C.: V. H. Winston, 1973.

Bhaskar, R., & Simon, H. A. Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science,* 1977, *1,* 193–215.

Buchanan, B. G., & Lederberg, J. The heuristic DENDRAL program for explaining empirical data. *Proceedings of the International Federation of Information Processing Societies.* Ljubljiana, 1971, 179–188.

Hayes, J. R., & Simon, H. A. Understanding written problem instructions. In L. W. Gregg (Ed.), *Cognition and knowledge,* Potomac, Md.: Lawrence Erlbaum Associates, 1974.

Hayes, J. R., & Simon, H. A. Psychological differences among problem isomorphs. In G. Potts (Ed.), *Indiana Cognitive Symposium,* Potomac, Md.: Lawrence Erlbaum Associates, 1977.

Hebb, D. O. *Organization of behavior,* New York: Wiley, 1949.

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

de Kleer, J. Multiple representations of knowledge in a mechanical problem-solver. *Proceedings, Fifth International Joint Conference On Artificial Intelligence.* Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1977, 299–304.

Lindsay, R. K. Inferential memory as the basis of machines which understand natural language. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill, 1963.

Moore, J., & Newell, A. *How can MERLIN understand?* In L. W. Gregg (Ed.), *Cognition and knowledge.* Potomac, Md.: Lawrence Erlbaum Associates, 1974.

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

Novak, G. S. *Computer understanding of physics problems stated in natural language.* (Tech. Rep. NL-30) Austin Tex.: Department of Computer Sciences, The University of Texas, March 1976.

Pople, H., Meyers, J., & Miller, R. DIALOG, a model of diagnostic logic for internal medicine, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence,* Tiblisi, USSR: 1975, 848–855.

Reddy, R., & Newell, A. Knowledge and its representation in a speech understanding system. In L. W. Gregg (Ed.), *Knowledge and cognition.* Potomac, Md.: Lawrence Erlbaum Associates: 1974.

Shortliffe, E. H. *Computer-based medical consultation: MYCIN.* New York: American Elsevier, 1976.

Siklóssy, L., & Simon, H. A. Some semantic methods for language processing. In Simon, H. A. & L. Siklóssy (Eds.), *Representation and meaning,* Englewood Cliffs, N.J.: Prentice-Hall, 1972.

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

Simon, H. A. Artificial intelligence systems that understand. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.* Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1977, 1059–73.

Simon, H. A., & Hayes, J. R. The understanding process: Problem isomorphs. *Cognitive Psychology,* 1976, *8,* 165–190.

Simon, H. A., & Simon, D. P. Individual differences in solving physics problems. In R. S. Siegler (Ed.), *Children's thinking: What develops.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1978.

Winograd, T. *Understanding natural language,* New York: Academic Press, 1972.