October 1982
Also numbered HPP-82-18

Report No. STAN-CS-82-936

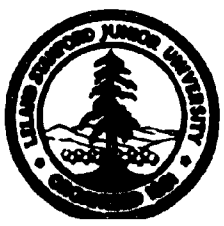
Models of Cognition

by

Paul R. Cohen

Department of Computer Science

Stanford University
Stanford, CA 94305



UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-82-936; HPP-82-18	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Models of Cognition		5. TYPE OF REPORT & PERIOD COVERED technical, July 1982
		6. PERFORMING ORG. REPORT NUMBER STAN-CS-82-936; HPP-82-18
7. AUTHOR(s) Paul R. Cohen (edited by Paul R. Cohen and Edward A. Feigenbaum)		8. CONTRACT OR GRANT NUMBER(s) MDA 903-80-C-0107
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford, California 94305 U.S.A.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Avenue, Arlington, VA 22209		12. REPORT DATE July 1982 13. NO. OF PAGES 87
14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office) Mr. Robin Simpson, Resident Representative Office of Naval Research, Durand 165 Stanford University		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report)
Reproduction in whole or in part is permitted for any purpose of the U.S. Government.
Distribution authorized to U. S. Government agencies and private individuals or enterprises eligible to obtain export controlled [illegible] in accordance with implementing 10 USC 140c. Other requests must be referred to DARPA/IPTO 1400 Wilson Bl. Arlington, VA. [illegible] (X)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Elementary Perceiver and Memorizer
General Problem Solver
Human Associative Memory
Reprints (GRW)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report is reproduced from Chapter XI, "Models of Cognition," of the Handbook of Artificial Intelligence (Vol. III, edited by Paul R. Cohen and Edward A. Feigenbaum). This chapter, written by Paul R. Cohen, discusses AI models of human memory, belief, and planning and problem solving. These programs (e.g., [illegible] and [illegible]) were among the earliest developed in AI and give some insight into the powerful influence of the computer on the development of AI and cognitive psychology.

DD FORM 1473
UNCLASSIFIED

Models of Cognition

by

Paul R. Cohen

Chapter XI of Volume III of the

Handbook of Artificial Intelligence

edited by

Paul R. Cohen and Edward A. Feigenbaum

This research was supported by both the Defense Advanced Research Projects Agency (ARPA Order No. 3423, Contract No. MDA 903-80-C-0107) and the SUMEX-AIM Computer Project under the National Institutes of Health (Grant No. NIH RR-00785). The views and conclusions of this document should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the National Institutes of Health, or the United States Government.

CHAPTER XI: MODELS OF COGNITION

FOREWORD

The Handbook of Artificial Intelligence was conceived in 1975 by Professor Edward A. Feigenbaum as a compendium of knowledge of AI and its applications. In the ensuing years, students and AI researchers at Stanford's Department of Computer Science, a major center for AI research, and at universities and laboratories across the nation have contributed to the project. The scope of the work is broad: About 200 short articles cover most of the important ideas, techniques, and systems developed during 25 years of research in AI.

Overview articles in each chapter describe the basic issues, alternative approaches, and unsolved problems that characterize areas of AI; they are the best critical discussions anywhere of activity in the field. These, as well as the more technical articles, are carefully edited to remove confusing and unessential jargon, key concepts are introduced with thorough explanations (usually in the overview articles), and the three volumes are completely indexed and cross-referenced to make it clear how the important ideas of AI relate to each other. Finally, the *Handbook* is organized hierarchically, so that readers can choose how deeply into the detail of each chapter they wish to penetrate.

This technical report is reproduced from Chapter XI, "Models of Cognition," of the *Handbook* (Vol. III, edited by Paul R. Cohen and Edward A. Feigenbaum). This chapter, written by Paul R. Cohen, discusses AI models of human memory, belief, and planning and problem solving. These programs were among the earliest developed in AI and give some insight into the powerful influence of the computer in the development of AI and cognitive psychology.

A. OVERVIEW

ANTHROPOMORPHISM is a powerful tendency in human thinking—we ascribe personalities and emotions to all kinds of animate and inanimate objects. Thus, it is not surprising that we should do the same with computers, or even that we should reverse the terms of the equation and describe ourselves in terms reserved for the machine. This is not a new trend—it certainly predates the electronic computer (e.g., the Futurists around 1910 extolled the virtues of the machine in their manifestos)—but the comparison between man and machine is particularly compelling in the case of the computer.

However, there is no science and no subtlety in the broad, unqualified claim that we behave like computers or vice versa; the trick is to know enough about how humans and computers think to say *exactly* what they have in common, and, when we lack this knowledge, to use the comparison to *suggest* theories of human thinking or computer thinking. Thus, psychology and AI have a reciprocal, piggyback relationship: What we learn about human intelligence suggests extensions to the theory of machine intelligence, and vice versa.

This reciprocal relationship was most evident during the early years of AI. For example, in 1956, Allen Newell and Herbert Simon developed a theory of problem solving called LT (for Logic Theorist), which they implemented as a computer program. Because the theory was formalized, Newell and Simon could specify *exactly* the problem-solving behaviors they expected to find in human problem-solvers. But when they tested their theory, they found that it failed in one respect: Humans did not use the same control process (working backward from theorem to axioms) as the program. Thus, they revised the theory, and wrote a new program, to incorporate what they had learned about human control processes during problem solving. They called the new program the General Problem Solver (GPS), and the new control process *means-ends analysis*, and found that this process was much more efficient (in terms of computer time) than its predecessor. Means-ends analysis is now an established problem-solving technique in AI.

This example illustrates how, by exploiting the comparison between human and machine problem-solving, it is possible to develop theories of both from relative ignorance of either. The first step was LT, a preliminary theory. The next step was to test LT against human problem-solvers. The third step was to derive a new theory, GPS, from differences between the old one and the experimental data. This theory was tested again and was more successful, both as a theory of human problem-solving and as a technique for AI. Note, however, that this development succeeded not by simply asserting that human problem-solving is like machine problem-solving but, rather, by describing

with precision their similarities and, more importantly for the development of the theory, their differences. Computer programs are precise descriptions of behavior and so are the results of experiments with humans; by using each to complement the other, a theory of behavior develops quickly.

This approach to psychological research is called *information-processing psychology* and, more recently, *cognitive science*. The theories that are developed—computer models of human thinking—are called *models of cognition*. The central idea of information-processing psychology is to bring precision to the seductive comparison between human and artificial intelligence, to benefit our understanding of human cognition. In the next section, we present a historical background to information-processing psychology.

A History of AI and Information Processing

Information-processing psychology has played an important part in the development of American psychology since 1950. It has helped to reinstate the concept of *mind*, which had been abolished by behavioral psychologists because it was unobservable except by introspection. *Methodological behaviorism* condemned introspection as a psychological method because there was no guarantee that the words used by one person to describe his (or her) mental events would mean the same thing to another person. For example, if a person says, "I can't quite think of the word—it is on the tip of my tongue," you may think you know what he is thinking and feeling, but, in fact, regardless of the detail with which he describes his state, you cannot guarantee that your knowledge of his state is completely accurate. A stronger position on introspection is taken by *radical behaviorism*, which holds that knowledge obtained by introspection not only cannot be accurately communicated, but is not even accurately perceived by the introspector: "An organism behaves as it does because of its current structure, but most of this is out of the reach of introspection" (Skinner, 1976, p. 19). Mental events are viewed as side effects of the interaction between an organism and its environment, not causes and thus not explanations of behavior.

These positions—radical and methodological behaviorism—were objective but resulted in a psychology that did not admit the mind. Theoretically, it was possible to explain behavior in terms of stimulus-response pairs, denying any mediating mental structures or processes:

> A person is changed by the contingencies of reinforcement under which he behaves; he does not store the contingencies. In particular, he does not store copies of the stimuli which have played a part in the contingencies. There are no "iconic representations" in his mind; there are no "data structures stored in his memory"; he has no "cognitive map" of the world in which he has lived. He has simply been changed in such a way that stimuli now control particular kinds of perceptual behavior. (Skinner, 1976, pp. 93–94)

In contrast, all the research described in the *Handbook* is concerned with structures and processes that mediate intelligent responses to stimuli. This fundamental change in theoretical positions took place between 1950 and 1960, during which time behaviorism was largely displaced by *cognitive psychology*. The key to the change was the concept of *information*. Following the publication, in 1949, of Shannon and Weaver's "The Mathematical Theory of Communication," information became a concrete, measurable quantity (see Shannon and Weaver, 1963). Initially, the strict mathematical conception of information was explored; theorists tried to fit many aspects of human communication into the general model proposed by Shannon and Weaver (see, e.g., Cherry, 1970). But the model was best suited to communication over electrical channels, and so, by the mid-1950s, a more relaxed, and more appropriate, conception of information was emerging.

An influential paper was "The Magical Number Seven . . ." in which Miller (1956) proposed that the information capacity of mental processes, particularly short-term memory, is best measured in terms of semantic *chunks*—meaningful units of information—not abstract *bits*. For example, words from a sentence and nonsense syllables are considered to be chunks of information and put approximately equal demands on memory, despite the fact that the words contain more information, in the mathematical sense, than the syllables. In the years following Miller's paper, information *structures* such as discrimination nets, associative semantic nets, and frames were developed to *represent* the information used in cognition. The original, mathematical formulation of information has been largely abandoned:

> The problem was that the bit gave a very poorly articulated characterization of the information. . . . As descriptions of the information have become more articulated, the theories composed out of them have become more successful. (Anderson and Bower, 1973, p. 136)

The increasing sophistication of computers and computer science was the most important factor in the development of information-processing ideas. During the late 1950s, there was the realization in information-processing psychology that the computer was not simply a device for shifting bits or "crunching numbers," but was more generally capable of any kind of symbol manipulation, of any kind of *information process*:

> An entirely different use of computers in psychology . . . has emerged. This . . . stems from the fact that a computer is a device for manipulating symbols of any kind, not just numerical symbols. Thus a computer becomes a way of specifying arbitrary symbolic processes. Theories of this type, which can be called information processing theories, are essentially nonquantitative (they may involve no numbers at all), although neither less precise nor less rigorous than classical mathematical theories. (Newell and Simon, 1963, p. 366)

And in cognitive psychology, the computer and the emergence of programs like LT had a profound effect, even though cognitive psychology does not share the enthusiasm of information-processing psychology for computer models:

> The activities of the computer itself seemed in some ways akin to cognitive processes. Computers accept information, manipulate symbols, store items in "memory" and retrieve them again, classify inputs, recognise patterns, and so on. Whether they do these things just like people was less important than that they do them at all. The coming of the computer provided a much-needed reassurance that cognitive processes were real.... Some theorists even maintained that all psychological theories should be explicitly written in the form of computer programs. (Neisser, 1976, pp. 5–6)

These theorists were Newell, Simon, and J. C. Shaw. Their position that computer programs can be psychological theories is the point at which cognitive psychology and information-processing psychology part company. For most cognitive psychologists, information processing is a metaphor for human thought, a means of focusing attention on new and interesting questions about the mind. Very few cognitive psychologists have implemented information-processing models—programs—of their theories. Even among those who have, the strong position that the program is itself a theory is not universally accepted; for example, Anderson and Bower (1973) explicitly limit the sense in which their model of human associative memory is a theory (Article XI.E2):

> It is important to be clear about the relationship between the theory and this simulation program. We make no claim that there is any careful correspondence between the step-by-step information processing in the simulation program and in the psychological theory.... The claim is sometimes made... that the program is the theory. This is not the case for HAM, and we wish to make this denial explicit. HAM represents a very complicated set of speculations about human memory. Only some of these are represented in the simulation program. Moreover, the simulation program does not serve as an embodiment of this subset of the theory; rather, it is but one test of the adequacy of that subset. (pp. 142–143)

(The relationship between cognitive psychology and information-processing psychology is discussed in more detail in Newell, 1970, and Miller, 1978.)

To complete this historical overview, we should note the relationship between AI and information-processing psychology. It was summed up nicely by Minsky (1968) in his own historical discussion in which he identified three extensions to early work in cybernetics:

> The first was the continuation of the search for simple basic principles.... The second important avenue was an attempt to build working models of human behavior incorporating, or developing as needed, specific psychological theories.... The third approach, the one we call *Artificial Intelligence*, was an attempt to build intelligent machines without any prejudice toward making the system simple, biological, or humanoid. (p. 9)

In other words, AI does not require that an intelligent program demonstrate *human* intelligence, but information-processing psychologists insist that the correspondence be proved.

This overview is almost current; we have discussed the common roots of AI, information-processing psychology, and cognitive psychology, and we have discussed the points at which they part company. However, we should note that we have presented the strongest version of the information-processing approach, that advocated by Newell and Simon. Their position is so strong that it defines information-processing psychology almost by exclusion: It is the field that uses methods alien to cognitive psychology to explore questions alien to AI. This is an exaggeration, but it serves to illustrate why there are thousands of cognitive psychologists, and hundreds of AI researchers, and very few information-processing psychologists. Recently, the strong position has been relaxed to admit research that does not necessarily *prove* the correspondence between programs and human behavior but that has some avowed concern for understanding human behavior. This research is called *cognitive science* by its practitioners.

The articles in this chapter discuss models of cognition that have, for the most part, been the historical shoulders on which cognitive science now stands. Of the eight articles, five are devoted to models of human memory, two to problem-solving, and one to belief systems. The emphasis on memory has two causes, one historical and one artifactual. Historically, cognitive psychology has concerned itself almost exclusively with memory, so it is not surprising that it should be a major topic in information-processing psychology. However, the proportion of articles would have been different had we included discussions of other cognitive science research in this chapter, rather than elsewhere in the *Handbook*—for example, research on speech understanding (Chap. V); on natural-language understanding, especially the work of Schank and his colleagues (Chap. IV); on planning (Chap. XV); and on learning (Chap. XIV).

The first model discussed in this chapter is, appropriately, Newell and Simon's General Problem Solver program (GPS; Article XI.B). It is some of the earliest research in information-processing psychology. The program introduced *means-ends analysis*, which constrains a problem solver to the task of reducing the differences between the current state of a problem and the goal state, or solution. The problem solver often cannot derive a solution immediately from the problem, so it is necessary to transform the problem into some intermediate state, from which the solution might be derived. GPS was tested extensively as a theory of human problem-solving.

The next article (Article XI.C) is also about problem solving; it discusses a model of *opportunistic planning* designed by Hayes-Roth and Hayes-Roth (1978). Their model is an interesting contrast to those discussed in Chapter XV on planning. Opportunistic processing involves a flexible control strategy (implemented with a *blackboard* control structure) that permits planning

decisions to be made when the opportunity arises, rather than in a strict order. Hayes-Roth and Hayes-Roth suggest that opportunistic processing is necessary for complex problem solving. Their model was developed specifically as a model of human planning abilities; thus, it is discussed in the context of this chapter on models of cognition.

About the time that GPS was being implemented, Feigenbaum was designing his Elementary Perceiver and Memorizer (EPAM) program, the first of the memory models considered in this chapter (Article XI.D). It learns *paired-associate nonsense syllables*, which, since the end of the 19th century, have been used in experiments to reduce the effect on memory of the meaning of the material being remembered. Paired associates allow probing: One of a pair of syllables serves as a cue to invoke the memory of the other syllable. Many things can be learned about memory by varying the speed at which syllables are presented, the number that must be remembered, or the similarity between the syllables. Feigenbaum modeled learning of the syllables as a process of storing just enough information about a syllable to distinguish it from the other syllables in memory at the time it was stored. Often, this did not require storing the whole syllable, which results in performance on a recall test that is less than perfect and strikingly similar to that of humans on similar tests.

In 1968, Quillian developed a model of *semantic memory* that provided the basis for the work described in the next three articles in this chapter (Articles XI.E1, XI.E2, and XI.E3). Conceptually, semantic memory models are very simple. They can be thought of as graphs, where the points (called nodes) represent concepts and the lines represent relations between the points. The meaning of a concept in a semantic net is represented by its connections (or *associations*) with other concepts.

Quillian's model was not developed as a psychological theory originally, but it was the first information-processing model that looked like it might explain recently discovered and curious effects of meaning on memory, for example, the *category-size effect*, whereby it takes longer to classify objects that are members of large classes than those that are members of small classes.

The MEMOD model developed by Lindsay, Norman, and Rumelhart (LNR; see Article XI.E4) is much more ambitious than Quillian's model. In the first place, it is intended to be a model of human memory that captures some of the richness of language. This requires three types of nodes, instead of just the one "concept" node of Quillian. Nodes represent concepts, but also *episodes* and *events*. Episode nodes can be the superordinate nodes of complex events like stories; moreover, MEMOD's interpreter can "run" these events to simulate them. Episode nodes can designate arbitrary procedures that the interpreter can execute. The MEMOD model also permits a large number of relations between nodes, where Quillian had only about half a dozen. Further, relations in this model have a *case structure* similar to that of Fillmore (see Article IV.C4, in Vol. I). Another improvement over Quillian's model was

the introduction of more powerful *interpretive procedures*, since semantic-net models do not actually *do* anything except represent information. Interpretive procedures are required to manipulate this information.

The HAM model of Anderson and Bower (Article XI.E2) is also a model of human long-term memory (Human Associative Memory; thus, HAM), but it differs in a number of important respects from MEMOD. Although it has a network knowledge base, relations in the network are much simpler than those in MEMOD. They are based on the syntactic categories of a simplified grammar of English that is used to interact with the system. Another difference between the two systems is that, in HAM, arbitrary procedures cannot be written and the simple procedures that are used reside outside of the network. Anderson and Bower take the position that experimental data from the memory literature can be explained by a relatively simple *strategy-free* process.

Later work by Anderson on his ACT system is discussed in Article XI.E3. The ACT model uses a propositional semantic-network knowledge base, similar to that of HAM. It has, in addition, a procedural component to operate on the knowledge base. Procedures, represented by a production system, are written by the user of ACT. This feature makes ACT rather like the MEMOD system in that both provide a language for their users to build computer models of psychological processes. The major differences between the systems arise from the way procedures are represented and from the interpreter, which controls the flow of computation in the systems.

The last article in this chapter (Article XI.F) discusses belief systems, in particular, the models of ideological oversimplification designed by Abelson and the PARRY model of paranoia built by Colby and his associates. These models have in common a representation of beliefs that affect interpretations of sentences. For example, a "typical liberal" would interpret a national event, like Congress appropriating money for urban redevelopment, in a different way than would a "typical conservative." The article reviews a recent paper by Abelson, in which he discusses some differences between belief systems and the knowledge-based expert systems that are current in AI.

References

A concise, personal history of the first years of information-processing psychology is given in Newell and Simon (1972, pp. 873–889). Cherry (1970) is a comprehensive and readable account of the very early work in psychology, telecommunications, cybernetics, and computer science; it is a good resource for readers who want to know about the intellectual background that gave rise to AI and its related disciplines. Anderson and Bower (1973) present a detailed review of the history of associationism in memory research, as well as a review and criticism of several memory models. Several books by cognitive scientists give their perspective on the new field: Bobrow and Collins (1975) contains

several interesting papers on the developing topic of knowledge representation. Norman and Rumelhart (1975) discuss their MEMOD system in detail—it is interesting to contrast this book with a "standard" text on memory (e.g., Crowder, 1976) to see what a difference the information-processing perspective can make. Schank and Abelson (1977) discuss their theory of knowledge representation—a theory that is currently very popular. Finally, there is a journal called *Cognitive Science* that publishes current research.

B. GENERAL PROBLEM SOLVER

HUMAN PROBLEM-SOLVING has received intensive examination by Allen Newell, Herbert A. Simon, and their colleagues and students at Carnegie-Mellon University. In their book *Human Problem Solving* (1972), Newell and Simon present thorough analyses of problem solving in three task domains—cryptarithmetic, logic, and chess—and they present and evaluate information-processing systems that accurately simulate human thought in these domains.

There is not the space here to summarize all the work in human problem-solving. In fact, the only system we examine is the General Problem Solver program (GPS); and the only task domain, logic problems. However, the information-processing system that Newell and Simon develop is certainly general enough to provide a framework for problem solving in several other task domains. GPS is not just a logic problem-solver.

Problem solving, and most other intellectual activity, involves general knowledge that applies to many problems and very specific knowledge that is special to a particular problem. For example, a general rule, or *heuristic*, is "If you can't solve the whole problem, try to solve part of it." A specific piece of knowledge that may be useful for solving some word problems is, for example, that a mile is 1,760 yards. The distinction between general and task-specific knowledge is made in GPS, and it was for just this reason that it was called GPS:

> GPS obtained the name "general problem solver" because it was the first problem solving program to separate in a clean way a task-independent part of the system containing general problem solving mechanisms from a part of the system containing knowledge of the task environment. (Newell and Simon, 1972, p. 414)

Accordingly, our discussion of GPS moves from general to specific: First is a simplified discussion of the information-processing system on which GPS is constructed, then a presentation of general problem-solving methods, and finally consideration of methods specific to the task demands of logic problems.

The Information-processing System

Everything that takes place in GPS is an information process, and the environment in which GPS solves problems is called an *information-processing system* (IPS). A central concept is that of a *state*—a momentary snapshot containing what the IPS knows at the time. The knowledge implicit in a state is represented by *symbol structures*.

More formally:

1. There is a set of elements called *symbols;* a symbol structure is a set of instances of symbols connected by *relations.*
2. An *information process* is a process that has symbol structures for all or some of its inputs or outputs.
3. An *object* is a symbol structure, or a program that the IPS is capable of executing, or an external environment of readable stimuli.

States are derived from other states by the application of information processes, often called *operators.* Two important states are the *starting state,* which represents everything that the IPS knows at the beginning of the problem, and the *goal state,* which represents the knowledge of the IPS when it has solved the problem. There may be many goal states, corresponding to various solutions to a problem. For example, the starting state in a game of chess is the familiar double ranks of opposing black and white pieces. From this single starting state an enormous number of goal states representing checkmate can be derived. Each new position is derived from its predecessor by an operator, a legal move of one or two chessmen. A final point about this state-space representation is that symbol structures may be nested in an IPS; within the structure that corresponds to a whole board position, there are a number of smaller structures corresponding to parts of it.

Since an object is defined as a symbol structure, a program to be executed, or external data, no distinction is made between data and programs. This is an important aspect of GPS and of many other AI programs, but for the sake of simplicity we will ignore the possibility that objects can be programs. From here on, *object* refers to symbol structures, and *operator* or *information process* denotes the programs that the IPS executes. As an example of this more restrictive definition, configurations of chess boards are objects and the moves of the chess pieces are operators. Note that an object may represent an entire chess board or just a part of it. A state, then, is composed of one or more objects, and it is transformed by operators.

Elementary Information Processes

Newell and Simon suggest some *elementary information processes* (EIPs) from which all the other operations of an IPS can be constructed. They are:

1. *Discrimination.* The IPS must be able to invoke operators appropriate to the symbol structure it is currently processing.
2. *Tests and comparisons.* The IPS must be able to compare symbol structures.
3. *Symbol creation.* It must be possible to create symbols and allow them to designate other symbol structures.

4. *Designation of symbol structures.* It must be possible to designate various parts of any symbol structure and obtain the designation of any part of any symbol structure.
5. *Input and output.* The IPS must be able to read and write symbol structures internally and externally.
6. *Storing of symbol structures.* It must be possible to store a symbol structure and retrieve it by means of another symbol structure that designates it.

The Problem Space

Newell and Simon define the *task environment*, or *problem space*, of GPS to be the formal specification of the set of symbol structures through which GPS searches for a solution. This may suggest that GPS has a collection of states available to search for a goal state. In fact, search in GPS means that GPS *generates* states by applying operators, first to the starting state (which it is given), then to states it derives from the starting state, and so on. GPS generates states in its problem space as it solves a problem.

The problem space used by GPS varies with the problem. It is a formal specification of the knowledge needed to solve a problem. Consider, for example, the famous cryptarithmetic problem

$$\begin{array}{r} \text{DONALD} \\ +\ \underline{\text{GERALD}} \\ \text{ROBERT} \end{array} \quad \text{Given } D = 5$$

where the object is to assign digits to letters so that the sum of the numbers denoted by DONALD and GERALD equals the number denoted by ROBERT. A problem space for this example is:

⟨letter⟩	:= A|B|D|E|G|L|N|C|R|T
⟨digit⟩	:= 0|1|2|3|4|5|6|7|8|9
⟨expression⟩	:= ⟨letter⟩ has-value ⟨digit⟩
⟨knowledge state⟩	:= ⟨expression⟩ | ⟨expression⟩ & ⟨knowledge state⟩
⟨operator⟩	:= Assert(⟨expression⟩).

All knowledge about this problem is made up of expressions of the form *letter* has-value *digit*. The initial knowledge state is the single expression *D* has-value *5*. Subsequent knowledge states are conjunctions of expressions. The single operator required to solve the problem is to assert that a letter has a particular value, that is, to assign it the value. This problem space is complete in the sense that application of the operator is enough to generate all the expressions needed for a solution.

In addition to the problem space, the IPS needs a program, or set of instructions, to dictate how digits are to be assigned to letters and to test if

a solution has been found. This will be discussed further for the domain of logic problems.

A distinction must be made between *search in the problem space* and the *search space*. The former refers to all the solutions and paths leading to them that the problem solver actually generates, while the latter refers to all the solutions and paths *that exist*. For problems of any complexity, it is necessary to keep the problem space smaller than the search space. To rephrase a point made in Chapter II: Search in the problem space involves generating just enough of the search space to find a solution to the problem. In GPS, two methods are used to accomplish this. One is a general heuristic called *means-ends analysis*, and the other is a form of *planning*. We will not consider planning here; the interested reader should see pages 429–435 of Newell and Simon (1972) and Article XV.A in the *Handbook*.

General Problem-solving Methods: Means-ends Analysis

Problem solving in GPS is a matter of transforming the start state into a goal state. Thus, at any point during problem solving, GPS has two goals:

1. Transform state 1 to state 2 by the application of operators.
2. Apply some operator to state 1 (or some intermediate state).

These goals do not specify *which* operator should be applied to any object. There are numerous strategies for deciding this. One is to apply *all* legal operators to the first object, then apply all legal operators to all the results of the first application, and so on. This method, called *exhaustive search*, generates the entire search space. It is guaranteed to find a solution eventually but is much too costly to be used for problems of any complexity. Means-ends analysis is a powerful heuristic that constrains search by anchoring paths in the search space to the current state and the desired state; it implies a third problem-solving goal for GPS:

3. Reduce the *difference* between state 1 and state 2 by modifying state 1.

This rules out directionless expansion of possible solutions:

> By taking account of the characteristics of the goal object it is seeking to reach, the problem solver extracts from the situation an enormous amount of information about the direction in which it should explore, and almost immediately rules out of bounds all but a tiny portion of the problem space. (Newell and Simon, 1972, p. 426)

Means-ends analysis is incorporated into GPS as follows:

1. If the current state is not the desired one, *differences* between it and the desired state will be detected.

2. Operators can be classified according to the differences they eliminate.
3. It may be necessary to modify the current state to make it compatible with a desired operator.
4. "Difficult" differences between states might be simplified by transforming the current state, even if this results in more, though simpler, differences.

The IPS, problem space, search, and means-ends analysis are domain-independent ideas. The GPS program was designed to separate them from any given problem-solving task. In the next section, we look at an example of GPS in the task-domain of logic problems.

Task Demands of Logic in GPS

Symbolic logic problems provide an ideal situation to study problem solving because one can describe the *task environment* of these problems in great detail. One such problem is:

Translate the expression $R \,\&\, (\neg P \rightarrow Q)$ into $(P \vee Q) \,\&\, R$.

It is unimportant what the connective symbols ($\rightarrow$, $\neg$, $\&$, $\vee$) mean. (In fact, the human problem-solvers who provided data for Newell and Simon were told nothing about them except that they were a set of transformations for turning one expression into another.) Each transformation reduces a difference between two expressions. The problem is to use these transformations to turn the first expression, $R \,\&\, (\neg P \rightarrow Q)$, into the second one, $(P \vee Q) \,\&\, R$. The available transformations were the following (in which ":" means "translates to" and A and B are arbitrary expressions):

$\neg\neg A : A$	$A \,\&\, A : A$
$A \,\&\, B : A$	$A \,\&\, B : B$
$A \vee A : A$	A and $B : A \,\&\, B$
$A \,\&\, B : B \,\&\, A$	$A \vee B : B \vee A$
$A \vee B : \neg(\neg A \,\&\, \neg B)$	$A \rightarrow B : \neg A \vee B$
$A \rightarrow B : \neg B \rightarrow \neg A$	$A \rightarrow B$ and $A : B$
$A \vee (B \vee C) : (A \vee B) \vee C$	$A \,\&\, (B \,\&\, C) : (A \,\&\, B) \,\&\, C$
$A \vee (B \,\&\, C) : (A \vee B) \,\&\, (A \vee C)$	$A \,\&\, (B \vee C) : (A \,\&\, B) \vee (A \,\&\, C)$
$A \rightarrow B$ and $B \rightarrow C : A \rightarrow C$	$A : A \vee X$ (X is any expression)

Consider how these rules can be used to translate from the original to the goal expression:

Expression	*Transformation*
$R \,\&\, (\neg P \rightarrow Q)$	$A \,\&\, B : B \,\&\, A$ yields $(\neg P \rightarrow Q) \,\&\, R$
$(\neg P \rightarrow Q) \,\&\, R$	$(A \rightarrow B) : (\neg A \vee B)$ applied to left part yields $(\neg\neg P \vee Q) \,\&\, R$
$(\neg\neg P \vee Q) \,\&\, R$	$\neg\neg A : A$ applied to left part yields $(P \vee Q) \,\&\, R$

$(P \vee Q) \,\&\, R$ is the goal expression. Q.E.D.

One can now see how GPS works in the task environment of logic problems. Exhaustive search would eventually generate the goal state but is wasteful here because it ignores the information provided by the goal state. Means-ends analysis directs GPS to reduce the difference between the starting state and the goal state. For example, comparing the start state to the goal state, it is immediately obvious that the former needs to be turned around: R must appear on the right of the parentheses instead of on the left. This is a *difference* between the two states; it can be reduced by the transformation $A \,\&\, B : B \,\&\, A$. Instead of applying *all* applicable transformations to the starting state, GPS might simply apply this one, which will yield the state $(\neg P \rightarrow Q) \,\&\, R$.

Continuing this reasoning, one might try to reduce the differences between $(\neg P \rightarrow Q)$ and $(P \vee Q)$. There are two differences: P has a "$\neg$" prefix in one case but not the other, and the connective between P and Q is "$\rightarrow$" in one case and "$\vee$" in the other. One transformation will reduce the latter difference, namely, $A \rightarrow B : \neg A \vee B$. Application of this transformation yields $(\neg\neg P \vee Q) \,\&\, R$.

The final problem is to get rid of the "$\neg\neg$" prefixing P. One transformation is available to do this, $\neg\neg A : A$, which yields the goal state $(P \vee Q) \,\&\, R$ when it is applied.

(The reader who wants a "real life" example of problem solving with means-ends analysis is encouraged to read Article XV.B on the STRIPS planner, in the *Handbook*.)

The reasoning of the last paragraphs is a simplified version of the operation of GPS. Means-ends analysis is demonstrated here in its simplest form: At each step in solving the problem, a transformation is chosen that will reduce one difference between the current state and the goal state. GPS is able to do this because each of the transformations it uses in a task domain is *classified* according to the differences it reduces. For the logic task domain, there are six differences that can be reduced by transformations. In GPS these are summarized in a *difference table*. Three of the reducible differences are:

1. A difference in position of components of the expression. Several transformations will eliminate this difference:

$$A \vee B : B \vee A, \quad A \& B : B \& A, \quad A \rightarrow B : \neg B \rightarrow \neg A, \quad \text{etc.}$$

2. A difference in the symbol that appears between letters. Transformations to eliminate this difference are:

$$A \vee B : \neg(\neg A \& \neg B), \quad A \rightarrow B : \neg A \vee B,$$
$$A \vee (B \& C) : (A \vee B) \& (A \vee C), \quad \text{etc.}$$

3. A difference in the number of "¬" prefixes of a letter. Several transformations change the number of prefixes:

$$\neg\neg A : A, \quad A \rightarrow B : \neg A \vee B, \quad A \rightarrow B : \neg B \rightarrow \neg A, \quad \text{etc.}$$

To solve the problem above, GPS determines the differences between the starting state and the goal state and then applies transformations that reduce them. However, the problems solved by GPS are rarely so simple; several complications must be considered. First, if several transformations are applicable to a state, GPS must choose between them. To do so, it consults a *ranking* of differences that tells it which differences to reduce first.

Another complication arises when GPS cannot find an operator to reduce a particular difference. In this case, it must transform the current state into an intermediate state from which it can reduce the difference. For example, consider adding the transformation rule $A \circ B : A \vee \neg B$ and solving the problem defined by the starting state $R \circ (\neg P \rightarrow Q)$ and the goal state $(\neg P \& \neg Q) \vee R$. In this case, GPS sets up the goal of moving R to the other side of the expression, as it did in the last problem, but it has no transformations available to accomplish this. Instead, it must defer this goal and transform the starting state into a state from which it can accomplish the goal. To do this, it transforms $R \circ (\neg P \rightarrow Q)$ into $R \vee \neg(\neg P \rightarrow Q)$ and then into $\neg(\neg P \rightarrow Q) \vee R$. Thus, GPS has the ability to set up *nested subgoals*.

The design of GPS is dictated by the heuristic of means-ends analysis and by the *task demands*. The general part of GPS is means-ends analysis and the information-processing system in which it operates. The remainder of the system follows from the task of solving logic problems. There are a limited number of differences possible and a limited number of operations to reduce them.

Empirical Tests of GPS

GPS was proposed as a psychological theory of human problem-solving. In this section we give evidence for the theory. Recall that the most general aspect of GPS is means-ends analysis, which is used to guide the generation of states in the problem space. Some general behaviors are a natural consequence of means-ends analysis; for example, GPS works forward from the

current state to the goal state, as opposed to working backward from the goal. Another general characteristic of GPS is the repeated application of transformations to states. This refers to the situation in which GPS finds a transformation it wants to use, but the current state is not in a form that will accept the transformation; the state must be altered and the transformation reapplied.

If GPS is a theory of human problem-solving, one would expect humans to use means-ends analysis and exhibit the behaviors that derive from it in situations where GPS exhibits these behaviors. In the case of logic problems, this is easily tested. Task demands are equated by ensuring that GPS and the human subjects have the same transformations to work with and the same problems to solve. GPS is programmed to print out its goals as it tries to solve the problem, and the humans are instructed to talk out loud as they solve the problem. The subjects' comments are recorded and the resulting record is called a *protocol*, which is broken down into phrases:

"I'm looking at reversing these two things now."

"Then I'd have a similar group at the beginning..."

"I could easily leave something like that 'til the end."

These are classified as evidence of goals and applications of transformations.

Breaking down the protocols is a painstaking process, but it is expedited by a structure called the *problem behavior graph*, a graphic display of the problem solver's progress. The nodes of the graph represent the knowledge of the problem solver at a given point in time, and the arcs represent the transformations that lead to new nodes (states). There is also provision for returning to parts of the problem that were left dormant while a particular line of reasoning was being explored. The protocol of each subject is mapped onto a problem behavior graph. Newell and Simon do not expect that any problem behavior graph will precisely match the output of GPS on a problem. Their claim is, rather, that patterns of behavior will be common to GPS and all their subjects. The problem behavior graph provides an explicit record of the behavior, from which patterns can be abstracted if they exist.

The following is a summary of an analysis of the problem-solving behavior of seven human subjects on a single problem. Newell and Simon classify the behavior of both GPS and their subjects into patterns and compare them for overlap. (This analysis is taken from pp. 489–502 of Newell and Simon, 1972.) Mnemonics for these patterns and the percentage of their occurrence in the protocols of each subject are shown in Table B–1. Total percentages are shown for the pooled sum of utterances in all seven protocols. Table B–1 has three horizontal divisions, or tiers, representing (a) patterns exhibited by both GPS and the subjects, (b) patterns exhibited by the subjects and *not* by GPS, and (c) uninterpretable behavior on the part of the subjects.

TABLE B-1
Percentages of Particular Problem-solving Patterns in Protocols of Individual Subjects

	Subject							
	A	B	C	D	E	F	G	TOTALS
Tier 1. Behavior exhibited by subjects and by GPS								
Means-ends analysis (toward goal object; operator applicability)	37	47	48	38	52	50	45	39
Working forward	17	0	13	14	2	1	9	7
Repeated application (after subgoal: implementation)	46	44	37	39	39	44	42	38
							Subtotal	84
Tier 2. Behavior exhibited by subjects and absent in GPS								
Means-ends analysis (consequence avoidance)	0	0	0	<1	<1	5	7	>3
Working backward	0	2	0	0	0	0	0	<1
Repeated application (review)	0	0	5	15	6	8	9	7
							Subtotal	11
Tier 3. Uninterpretable behavior								
	0	3	2	9	7	5	4	5
							TOTAL	100

In the first tier of the table, means-ends analysis has two manifestations in which states are transformed to achieve the goal expression or are transformed into a form compatible with a desired transformation. A second pattern of behavior is *working forward*, that is, searching through transformations for one that will apply to the current state. A third pattern is repeated application of a transformation on the same state. This event arises mostly when a desired transformation is incompatible with a state. A goal is set up to transform the state, and the original transformation is then successfully reapplied. Another type of reapplication found here is to try out consequences of a transformation before committing the system to it. Table B-1 shows clearly that the great

majority of the utterances of the seven subjects conform to these patterns of behavior—84%, in fact.

Tier 2 represents human behaviors that were not implemented in GPS at the time. The greatest percentages were obtained for the reapplication of transformations for the sake of review (refreshing the memory). Working backward from the goal was another behavior that had not been implemented in GPS. A third is a complex behavior in which a transformation is applied before the application of the desired transformation, because the latter has undesirable consequences (as well as the desired ones) if applied before the intermediate transformation. These behaviors constitute 7%, 1%, and 3% of the protocols, respectively.

Tier 3 of the table accounts for 5% of the subjects' protocols and represents uninterpretable behavior that could not be assigned to any pattern. These behaviors include grunts and yawns, and unfinished and ambiguous phrases such as *Well, this looks like, uh ... I dunno.*

Conclusion

From this and other analyses, Newell and Simon conclude that GPS is an explicit, operational, and sufficient model of some human problem-solving. In GPS, a separation is maintained between general components, such as the information-processing system and means-ends analysis, and task-specific components, such as details of the problem space. Newell and Simon claim that the general components apply in a wide range of task domains. Chess and cryptarithmetic were examined in addition to logic problems, and these analyses certainly support Newell and Simon's argument of generality. Moreover, since GPS, means-ends analysis has been used in several other problem-solving programs (see Article XV.A).

Some problems are not solved efficiently with means-ends analysis. For example, the heuristic can lead one down a long path of problem-solving operators that dead-ends, forcing the problem solver to back up to a previous decision point and try a different path. Also, means-ends analysis may construct a series of problem-solving operators that will, in fact, solve the problem, but that is much longer than necessary. Lastly, means-ends analysis can be inefficient when there are interacting subgoals to be achieved; if accomplishing one subgoal prevents accomplishing another, the problem-solver can do no more than return to the beginning of the problem to try the subgoals in a different order (see Article XV.A for a detailed discussion of this problem).

However, the efficiency of problem-solving is a big concern for computers, but perhaps not a serious concern for humans. The fact that means-ends analysis can be inefficient does not detract from the empirical fact of its generality in human problem-solving. This is not to say that means-ends analysis is the only problem-solving strategy used by humans; the following

article (Article XI.C) will discuss a planning problem that is best solved by a process called *opportunistic planning*.

References

The most comprehensive and exhaustive information-processing analysis of human problem-solving is Newell and Simon (1972).

C. OPPORTUNISTIC PROBLEM SOLVING

THIS ARTICLE discusses a theory of planning developed by Barbara Hayes-Roth and Frederick Hayes-Roth (1976; B. Hayes-Roth, 1980). The theory is specifically of human planning, and the authors and their colleagues have run several experiments to test it. For this reason, the theory is discussed here rather than in Chapter XV, on planning.

Hayes-Roth and Hayes-Roth have implemented their theory in a model that, due to its complexity, will be sketched later in this article but not presented in detail. The first part of the article discusses an exploratory experiment with human planners in which subjects were required to think out loud while planning. This technique is familiar from the work of Newell and Simon (Article XI.B). A transcript, or *protocol*, is broken down into phrases that are interpreted as evidence of particular planning or problem-solving operations.

In the planning experiment (Hayes-Roth and Hayes-Roth, 1978), subjects were given a map of a small town marked with points of interest such as movie theaters, the veterinarian, stores, and restaurants. They were asked to plan a day's activity that included 10 errands, such as *Get medicine from the vet* and *Buy fresh vegetables at the grocery*. A couple of errands included explicit constraints, such as the showtimes of movies. Constraints about other errands were implied; for example, fresh vegetables should probably be purchased in the evening, rather than leaving them in a car all day.

With the map and list of errands in hand, subjects talked about their developing plans for the day. What they said was recorded and transcribed; Table C-1 shows samples of one subject's comments as he planned his activities. These paragraphs are excerpted from a longer protocol of 47 such paragraphs; the numbers in parentheses indicate the position of each paragraph in the protocol. The paragraphs illustrate a number of important characteristics of human planning. In the first, the subject uses his knowledge to assign importance to each errand and, thus, to order them. World knowledge is also used to order plan steps in the later paragraphs, in which the subject tries to schedule the purchase of groceries to avoid spoilage.

The second and third paragraphs illustrate two styles of control of planning. In the second paragraph the subject is motivated by a number of individual goals; his thinking is *bottom-up*, or driven by what he perceives to be the immediately attainable goals of the problem. In the third paragraph, however, he starts planning at a different level of abstraction. From the goals previously articulated, he abstracts a higher level goal, to *do the errands in the southeast corner*. For three more paragraphs in the protocol (not excerpted here), the subject tries to fit errands into the general plan of heading southeast.

TABLE C-1

Excerpts from a Planning Protocol (from Hayes-Roth and Hayes-Roth, 1978)

1. (1) Let's go back down the errand list. Pick up medicine for the dog at the veterinary supplies. That's definitely a primary, anything taking care of health.... Buy a toy for the dog at the store. If you pass it, sure. If not, the dog can play with something else.
2. (7) The appliance store is a few blocks away. The medicine for the dog... isn't too far away. Movie theaters—let's hold off on that for a little while. Pick up the watch. That's all the way across town. Special-order a book at the bookstore.
3. (8) Probably it would be best if we started in a southeasterly direction.... I can see later on there are a million things I want to do in that part of town.
4. (23) Third item will be the newsstand since we are heading in that direction. Often I like to do that. I know buying a gardening magazine is hardly a primary thing to do, but since I'm heading that way, it's only going to take a second...
5. (31) I would like to plan it so I can see the movie, pick up the vegetables, pick up my car, and then go home. Vegetables would rot.
6. (35) Now we do have a problem. It's 2:00 and all we have left to do is see a movie and get the vegetables. And that's where I think I've blown this plan. I've got an hour left there before the movie...
7. (40) If I go get the groceries now, it's not really going to be consistent with the plans throughout the day because I've been holding off on the groceries for rotting. If I take them to the movie... vegetables don't really perish like ice cream.

When immediately attainable errands are pointed out to the subject, he says, *I can still do that and still head in the general direction.* In contrast to the earlier mode of planning, driven bottom-up by immediate goals, he now attempts to incorporate these goals into an abstract plan. This illustrates the ability of human planners to reason at many levels of abstraction and to move freely between them. Hayes-Roth and Hayes-Roth call this *multidirectional* processing.

The fourth paragraph illustrates one of the most interesting and fundamental characteristics of planning, and indeed of other aspects of cognition: It is *opportunistic*. The subject realized that he could fulfill one of his obligations "for free," and promptly did so. Goals that fit into a developing plan are integrated, and goals that belong together are clustered into subplans, often without regard for how the subplans will integrate with the overall plan. For example, early on in the protocol (not shown above), the subject plans to

end his day at the movie and then walk to a parking lot where his car is parked. This subplan is constructed when the subject notices the proximity of the movie and parking lot. There is a strong parallel between this process and *island driving*, in which a problem solver finds part of a solution that he thinks is correct—an island—and extends the solution from there, possibly toward another island. Subplans can be regarded as islands that are linked by sequences of planning actions. (For a detailed discussion of island driving in speech understanding, see Article V.C1, in Vol. I.)

The fifth and sixth paragraphs of the protocol show the subject summarizing his current state and realizing that the plan is flawed because he has too much time for what he has to do. At this point, he relaxes one of his requirements, that he purchase vegetables after the movie, to fill in the hour before the movie.

Opportunistic, multidirectional planning is very different from that practiced by the planners discussed in Chapter XV; human planning can be significantly more complex than that of current AI planners. Before we discuss the Hayes-Roths' model, we consider some of these differences.

Opportunistic processing has a bottom-up component; planning processes are instigated by something the problem solver notices about the state of the world. In human planning, steps are introduced into a plan whenever the opportunity arises to do so. This contrasts with the *least-commitment* strategies in NOAH and MOLGEN (see Articles XV.D1 and XV.D2), in which planning steps are refined only when there is evidence that they will not have to be abandoned later. In human planning, the carefully controlled introduction of plan steps implicit in NOAH and MOLGEN is abandoned for the advantage of introducing steps in a plan wherever they are convenient.

A closely related issue is that human planning is multidirectional: that is, it takes place at several levels of abstraction simultaneously. This contrasts with the *hierarchical* planners (discussed in Articles XV.B, XV.D1, and XV.D2), which develop detailed plans from abstract plans in a purely top-down fashion. NOAH and MOLGEN do not include detailed steps in a plan unless they have been refined from more abstract ones. This strategy helped them to avoid interactions between plan steps; MOLGEN would *post constraints* summarizing the implications of refining an abstract plan step for other parts of the plan, and NOAH used *critics* to check for interactions between plan steps as its plan developed. Both approaches rely on developing abstract plans into detailed ones in a top-down manner.

The major advantage of the least-commitment strategy of hierarchical planning is that it allows the planner to avoid subgoal interactions and, thus, plan constructively with a minimum of backtracking. Opportunistic planning leaves the planner susceptible to these interactions; an opportunistic, multidirectional planner is more likely to need to rewrite parts of its plan or change its goals than is a hierarchical planner. In fact, Table C-1 showed the planner committing himself to a plan that does not fulfill all his goals—he is

left with too much free time. However, instead of backtracking to a previous point in the plan and replanning, the planner instead relaxes one of his goals and decides to buy groceries before the movie.

Hayes-Roth and Hayes-Roth argue that opportunistic, multidirectional planning is more efficient than hierarchical planning when the problem to be solved is very complex. They say that hierarchical planning restricts the problem solver and does not permit organizing parts of a plan around interesting possibilities that emerge bottom-up (as can be done in island driving).

The relative efficiencies and advantages of hierarchical, least-commitment planning and multidirectional, opportunistic planning are issues for AI. However, our chief concern in this article is not with efficiency but, rather, with how human planners plan. The remainder of the article summarizes the Hayes-Roths' model.

The Control of Planning

Hayes-Roth and Hayes-Roth propose a *blackboard model* to represent the complex control structure of human planning. Blackboards have been used primarily to facilitate *interpretation* of noisy signals such as speech (see discussion in Article V.C1. in Vol. I) and data from sensors (see Article VII.C3, in Vol. II, on CRYSALIS; also, Nii and Feigenbaum, 1978). A blackboard model for signal interpretation typically has a number of *specialist* programs that produce hypotheses about aspects of the signal. For example, a speech-understanding program has specialists for dividing the speech signal into phonetic units, for guessing the syntax of the spoken message, for predicting the next word given those that have been spoken already, and so on. The hypotheses produced by each specialist are accessible to all, since they are posted on a central *blackboard*. Hypotheses posted by one specialist are data for others: for example, if the syntactic specialist posts the hypothesis that the next word is a verb, the lexical specialist can use this information to narrow the search for the exact word.

Theoretically, the control of processing in a blackboard model is *asynchronous* and *opportunistic:* Specialists post hypotheses in no particular order, and they use hypotheses posted by other specialists whenever they appear helpful. Although human planning, involves *generating* behavior rather than interpreting it, it does seem to be an asynchronous, opportunistic process. Plans are not developed all of a piece, but, instead, clusters or islands of planning actions are constructed, and they are linked to other clusters when an opportunity arises.

The Hayes-Roths' planning model involves a blackboard with five *planes* of planning decisions and many specialists that generate tentative decisions and record them on the blackboard. Planes are organized to reflect characteristic processes in planning. One is the *plan plane*, a plane of operations.

Decisions to execute the processes discussed in the protocol—going to the veterinarian, seeing a movie, and so on—are recorded in the plan plane. More general goals and general plans to accomplish them are also recorded in the plan plane.

At the level of the *meta-plan plane*, the planner makes decisions about how to solve the problem at hand. As we note in the discussion of MOLGEN (Article XV.D2), a planner can do a lot of reasoning about a problem before proposing so much as a single action to solve it. Decisions recorded on the meta-plan plane capture some of this reasoning. For example, the planner must represent the problem to itself and decide what *type* of problem it is, so that it can pick out a *problem-solving model*, or *strategy*. The way a problem is represented by the problem solver can affect the ease with which it is solved (Amarel, 1968); thus, identifying a problem and finding an approach to solving it are two very important decisions. Most planning programs have a single representation of a problem and a single, implicit strategy for solving it; for example, some *nonhierarchical* planners (discussed in Article XV.C) represent problems as a collection of propositions to be made true, and they solve the problems by pattern-directed invocation of procedures with backtracking. It is possible, even likely, that a human planner might adopt means-ends analysis with backtracking as a method; the choice between this and other possibilities is recorded on the meta plan plane.

Another kind of planning decision represented at the meta-plan level involves the *policies* followed by the problem solver: What constitutes a good solution? Is it to be quick and dirty or painstaking and elegant? Again, most AI planning programs do not make such decisions, which obviously lend power and flexibility to human problem-solving. We can usually decide when a solution is good enough (what Simon, 1969, calls *satisficing*); those who are never satisfied and those who are too easily satisfied—the compulsive and the slob—are often inefficient problem solvers.

The meta-plan plane records global decisions about how to approach a problem; between this level and that of individual planning operations, the Hayes-Roths place the *plan-abstraction plane*. Decisions recorded at this plane motivate operations recorded on the plan plane. For example, the decision to do all of the "primary" errands first is a formulation of an abstract plan; it motivates the decision—recorded on the plan plane—to divide errands into "primary" and "secondary" groups.

A fourth plane in the Hayes-Roths' model contains world knowledge. For the errand-planning task, the *knowledge-base plane* includes a list of the errands and a representation of the map. A point made earlier—that the representation of the problem affects the efficiency with which it is solved—holds also for the representation of knowledge pertinent to the problem. The Hayes-Roths represent the map in several ways to enhance problem-solving efficiency. At one level, the map is represented as sectors, for example, the southeast corner; at another level, neighbors are recorded, for example, a

movie house neighbors on a parking lot. A third level of information about the map represents routes between points of interest. This knowledge base would, of course, change for another kind of problem.

The fifth plane, the *executive plane*, schedules the planning decisions made by *specialists* that are recorded on other planes of the blackboard. We have characterized the kinds of decisions that are found in human planning, for example, decisions about specific planning actions, about approaches to a problem, and about abstractions of planning actions. The decisions are tentatively proposed and recorded on the appropriate plane of the blackboard by specialist programs that are sensitive to particular kinds of decisions. For example, a *proximity detector* specialist would note when two points of interest are nearby on the map; it would record pairs of *neighbors* on the knowledge-base plane of the blackboard. Specialists operate independently and asynchronously, as mentioned above. Consequently, a scheduler is needed to decide on a sequential order (since most present-day computers are sequential machines) for all the actions of specialists. Scheduling might be queue oriented, that is, first come first served, but, in general, humans do not schedule actions this way. Instead they schedule them according to their perceived efficiency, productivity, and the like.

Conclusion

Hayes-Roth and Hayes-Roth present a detailed example explaining how their model accounts for the protocol of a subject planning a day's activities (excerpted above). Rather than discuss the model in detail, we have presented its planes and specialists in quite general terms, attempting to characterize the types and levels of decisions that are necessary for planning. One general conclusion of this article is that human planners are much more sophisticated than any of the programs discussed in Chapter XV on planning. Multidirectional, asynchronous, and opportunistic processing is proposed to model this sophistication.

References

Hayes-Roth and Hayes-Roth (1978; B. Hayes-Roth, 1980) give accounts of their experiments and the model developed from them.

D. EPAM

EPAM (Elementary Perceiver and Memorizer) was developed in the period 1956–1964 by Edward Feigenbaum and Herbert Simon. This program was the first information-processing model of a number of well-known human verbal-learning behaviors. Though it sounds simple, rote learning of nonsense material has provided much evidence about the characteristics of short-term and long-term memory. Nonsense material is useful in that it avoids the effect of the meaning of a stimulus on how well it is learned; for example, familiar stimuli or stimuli that "fit in" with previously learned material are relatively easy to learn. When Ebbinghaus first used nonsense syllables in the 1870s, these factors were not understood. His method limited their effects, which, he felt, obscured the fundamental characteristics of memory. (An interesting sidelight on the topic of nonsense syllables is that Anderson and Bower, whose work is discussed in Article XI.E2, used meaningful sentences for their experiments on *strategy-free memory* because they felt that their subjects were likely to employ mnemonic strategies to remember nonsense stimuli.) EPAM provides an explanation of some of these characteristics, among them oscillation and retroactive inhibition, forgetting, and stimulus and response generalization.

Verbal Learning Behavior

To simplify the study of human verbal learning, psychologists have developed a number of experimental techniques (for a survey, see Baddeley, 1976). Most are based on the following procedure: The subject (whether human or EPAM) is required to memorize nonsense syllables in serial lists or associate pairs. The syllables are typically comprised of three letters, beginning and ending with a consonant, and are supposed to be meaningless for most subjects (e.g., XUM, JUR, FAZ). In paired-associate learning experiments, the first syllable of a pair is called the *stimulus* and the second is called the *response*.

EPAM was designed for paired-associate and serial learning, but in this article we will consider only the former. In a typical experiment, a set of nonsense syllable pairs is used. For each pair in the set, the stimulus syllable is displayed to a subject, who then attempts to say the associated response. Any errors made by the subject are recorded. The response syllable is then shown, so that both stimulus and response are in view, and the subject is able to refresh his (or her) memory of the association (or learn it, if this is the first presentation). After a few seconds, the next pair of syllables is displayed. This continues until all of the pairs have been displayed. The entire sequence is

called a trial. Trials are repeated until the subject is able to give the correct response for each stimulus. This is called *learning to criterion*. There is a relatively short period of time between trials, and the sequence of pairs is randomized from trial to trial.

A number of behaviors are typical in a paired-associate verbal-learning experiment:

1. *Stimulus and response generalization.* Overt errors in recall are often attributable to confusion by the subject between similar stimuli or similar responses. When similar stimuli are confused, their responses may become interchanged: when two responses are similar, the wrong one may be given to a stimulus.
2. *Oscillation.* Associations that are recalled correctly over several trials are sometimes forgotten only to reappear and then later disappear again.
3. *Retroactive inhibition.* When the paired-associate task is modified to include an intervening learning task, so that one list of syllables is learned and then another, and the retention of the original list is tested, the subject's ability to give correct responses is reduced by the intervening learning. Moreover, overt errors in recall are usually *intrusions* from the second list. The phenomenon disappears rapidly, however, and the subject's memory of the first list is refreshed during the next trial.

The EPAM Model

The EPAM program was written in IPL-V, one of the first list-processing languages. EPAM is a two-part system, with performance and learning components. In the performance mode, EPAM attempts to produce responses to stimulus syllables. In the learning mode, EPAM learns to discriminate and associate stimuli and responses. The model is easier to understand if the performance mode is discussed first.

The Performance System

After EPAM has learned a set of stimulus-response pairs, it is tested in a standard paired-associate task. The test, which is identical to that given to a human, involves presenting stimulus syllables to EPAM, which then must produce the associated response syllables. The performance system proceeds as follows. A stimulus syllable is encoded into an *input code* that directs the search of EPAM's memory, called a *discrimination net*. This search leads to a node in the net that contains a *cue*. Cues are information with which to search for a response syllable. Using the cue, EPAM searches the net again for a node containing the response, called a *response image*. The cue does not always hold enough information to find the response syllable. If it does, the response is given; otherwise, EPAM makes an error.

EPAM codes each stimulus syllable into an internal representation called the *input code*. This is based on certain features of the input characters, such as the "openness" of a letter (e.g., C versus O) and whether the letter contains crossed straight lines (e.g., X). Different sets of features have been used, but in all cases they must satisfy two criteria: They must be related in some way to features of letters, and they must be highly redundant (having many more features than are required to distinguish letters).

For the remainder of this discussion, to simplify the examples, we will assume that letters themselves and *not* features of letters are used as input codes. Thus, when EPAM is tested with the stimulus MUR, features of the letters M, U, and R are actually used as the input code, but for simplicity we assume here that the input code is MUR.

The primary memory structure of EPAM is the *discrimination net*. It is constructed during EPAM's learning mode and searched during the response mode. The input code is used to traverse the discrimination net, which normally contains a dozen or more pairs. The net is simply a binary search tree, with internal nodes representing tests of features of stimuli. The leaf nodes represent either cues or response images. A diagram of a discrimination net that has been constructed in response to the associate pairs DAX–JIR, PIB–JUK is shown in Figure D–1.

An example. Imagine that the input code to EPAM is the syllable PIB. EPAM will sort down the tree until it gets to the node representing PIB. It does this by going left or right at each internal node contingent on the results of the test at that node. At the PIB node it will find a cue, J–K, which will be used to traverse the tree again, from the root node down the right branch to the next node, then down the left branch to the JUK node. At this point, it will respond with the syllable JUK. Note that it is only necessary to store

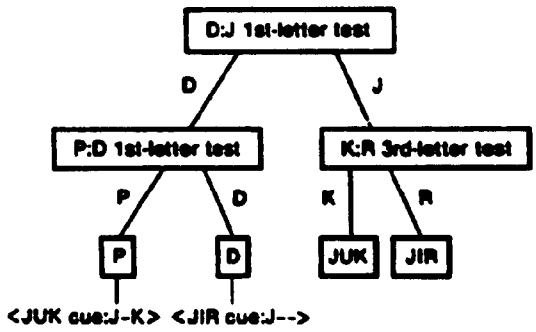


Figure D–1. A discrimination net for the associate pairs DAX–JIR, PIB–JUK.

enough features of the cue to direct EPAM to the response syllable at the time the cue is created. The method of constructing cues will be discussed later.

We have seen how EPAM performs when it gives correct responses to stimuli in the paired-associate task. To understand how EPAM fails at the task in ways that are characteristic of human memory, we will consider how it learns.

The Learning System

The discrimination learning system operates by constructing a discrimination net from a set of stimulus-response pairs. Initially the net is empty, and only a set of simple processes for growing nets and storing images at leaf nodes is available.

Suppose that the first stimulus-response pair is DAX–JIR and has already been learned. The discrimination net at this point is shown in Figure D–2.

The full response image must be stored in order to produce the response, but only partial stimulus-image information need be stored to recognize the stimulus. In this simple net, a single letter is enough to *discriminate* between the two syllables: therefore, the test at the root node is on a single letter and no other tests are necessary. Moreover, the cue to find the response need be only a single letter. The amount of information that needs to be stored at internal and leaf nodes is determined by the program as the net grows.

Suppose the second syllable pair to be learned is PIB–JUK; see Figure D–3. The net, as it stands, does not know about PIB; therefore, another test must be added to discriminate between the input codes for DAX and PIB. This new test is placed at the point in the net where there is a failure to discriminate.

Let us assume that the test is placed so as to discriminate between PIB and DAX, as shown earlier in Figure D–2. (The test could have been between PIB and JIR; EPAM is able to determine where the failure to discriminate occurs.)

Figure D–3 does not include a response image for the second syllable, JUK, or a cue at the leaf of the PIB branch to help EPAM find JUK later. The input code JUK is used to traverse the net until a discrimination failure occurs. In

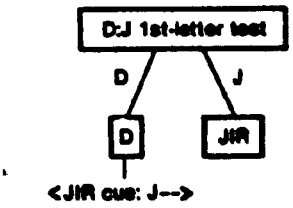


Figure D–2. Discrimination net for the associate pair DAX–JIR.

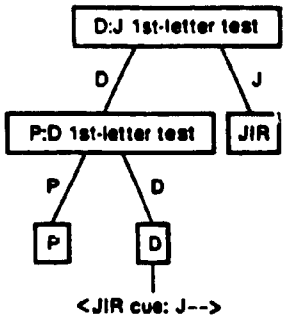


Figure D–3. Discrimination net for the associate pair DAX–JIR, which also discriminates PIB from DAX but does not include a cue or response image for the PIB–JUK association.

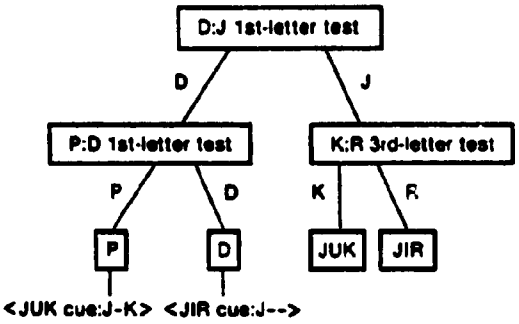
this case, the D:J test takes the J-branch and again a new discrimination must be added to distinguish JUK and JIR. Human subjects generally consider final letters before middle letters and EPAM does the same: It notes that the last letters of JUK and JIR differ, and a test is added to reflect this.

A cue to lead from the end of the PIB branch to the JUK response image is still lacking. It is constructed by trial and error. Each time a letter is added to the potential cue, it is used to traverse the net; see Figure D–4. Information is added to the cue as necessary until it leads to the correct response image. This method ensures that a cue contains the minimum information required to find the appropriate response image *at the time of memorization.*

It is now possible to see EPAM's source of errors on the paired-associate task: Cues are constructed to guarantee correct retrieval of the appropriate response image at the time the association is formed. If at some later time the net incorporates other images and cues, the cue might no longer be sufficient to perform that task. Thus, responses are forgotten temporarily. No information is destroyed, but some becomes inaccessible. This can be seen by comparing Figures D–2 and D–4. When the DAX–JIR association was first constructed (Fig. D–2), the cue for JIR, J--, was sufficient to find the response to DAX. However, when JUK was added to the net (Fig. D–4), J-- became inadequate to discriminate between JIR and JUK.

The DAX–JIR association is not necessarily lost forever. If the association is repeated (typically during a later trial), it will be reconstructed in the net with the information necessary to maintain the association at *that* time.

There is another aspect of the cue-construction method that results in inadequate cues. This has nothing to do with the discriminability of a cue changing due to the expansion of the net; rather, it derives from a single



random decision made by EPAM while it is constructing a cue. For example, if J- - is proposed as a cue for JUK, when the cue is tested, it will lead to a branch in the tree that has JUK on its left branch and JIR on its right. At this point, EPAM chooses one of the branches at random. If it goes left, it will find JUK and conclude that the cue is sufficient to find JUK in future, when in fact, this is not so.

EPAM's Verbal-learning Behavior

EPAM behaved very much like a human subject in classical rote-learning experiments. It provided a parsimonious explanation of rote-learning behavior, since retroactive inhibition, oscillation, stimulus and response generalization, and forgetting can all be seen to stem from a single mechanism. As items are learned, the discrimination net grows to accommodate new stimulus-response pairs. However, the cues that associate the stimuli with their responses guarantee correct response retrieval just at the time of the association. A cue that leads to the appropriate response image can fail to do so at a later time.

The oscillatory behavior exhibited by EPAM serves as a basis for an alternative explanation of forgetting. The usual explanation is that the information is destroyed over time, typically by overwriting or decay. Forgetting in EPAM occurs not because the information is physically destroyed but because it becomes inaccessible in the growing network of new associations. Furthermore, forgetting in EPAM is only temporary: Lost associations can be recovered by updating the appropriate cue with more information during another trial.

This process accounts for the fact that more than one trial is usually required to *learn to criterion*, that is, to give the correct response to each

stimulus. During the first trial, each cue is constructed with enough information to find the correct response at the time it is stored: a subsequent stimulus-response pair may be added such that the original cue can now no longer discriminate between its correct response and the new one. This was shown in Figure D–4: The J-- cue was sufficient to produce a response when DAX–JIR were the only elements in the net, but as soon as PIB–JUK were added. J-- was ambiguous with respect to JIR and JUK. Thus, on the next trial, EPAM might respond to DAX with JUK; this would be incorrect and an example of *response generalization*. However, the correct association is always shown after a stimulus-response test, so EPAM has the opportunity to update the J-- cue to make it discriminate JIR and JUK. On the next trial, it will not confuse the two. Thus, in the course of a number of trials, EPAM gradually learns to discriminate all stimuli and their responses.

If stimuli and their responses were initially very discriminable, EPAM would require less time to learn them. This is because there is less chance of response generalization. Operationally, this means that when EPAM constructs a cue with the minimum information needed to find a response image, it is less likely that a subsequent stimulus-response pair will render the original cue ambiguous.

If the same discrimination net is used for two trials, that is, two different sets of stimulus-response pairs, the discrimination net that was sufficient to respond correctly to all stimuli during the first trial may now be unable to discriminate between responses for trial 1 and responses for trial 2. This produces the phenomenon of *retroactive inhibition*, which is the deleterious effect of learning an intermediate list on recall of the original list. It also predicts the result that errors are likely to be *intrusions* from the second list, rather than confusions between responses in the first list.

One problem with EPAM was that it had no mechanism to model *proactive inhibition*, the situation in which learning one list of stimuli interferes with the learning of the next list. Typically, when a subject is tested on the second list, intrusions from the first result. Both proactive and retroactive inhibition are evident in verbal-learning experiments, but EPAM exhibited only the latter. EPAM has since been extended to deal with proactive inhibition by Hintzman (1968) in his SAL (Stimulus and Association Learner) program. This was accomplished by having a push-down stack at each leaf node in the discrimination net. Instead of a single image and cue at a leaf node during an experiment, the associations from multiple experiments were allowed to accumulate by being pushed onto the appropriate stacks. Thus, the most recently learned association would be on the top of each of the stacks. If the stacks were randomly disrupted, the responses that "spontaneously rise" to the top of the stacks might be responses from previous experiments. Another accounting of proactive inhibition given by Anderson and Bower (1973, pp. 74–75) in their review of EPAM is that instead of a stacklike structure, a list of cues is

kept, and the ordering of elements in the lists gets reshuffled, possibly as a consequence of the subject thinking about the material he has learned.

References

Feigenbaum (1963) and Simon and Feigenbaum (1964) are interesting treatments of EPAM and the empirical studies done with it. Feigenbaum and Simon (1962) is a discussion of an important verbal-learning effect—the serial-position effect. Anderson and Bower (1973, pp. 69–76) review and criticize the EPAM theory, and Simon (1979, pp. 99–100) provides a rebuttal to each of their criticisms.

E. SEMANTIC NETWORK MODELS OF MEMORY

E1. Quillian's Semantic Memory System

THERE are numerous intelligent behaviors of computers that depend on knowing the meanings of words, for example, machine translation, summarizing text, and speech understanding. The *semantic net* formalism developed by Ross Quillian was the first attempt to provide an operational representation of word meaning. The basis for Quillian's model is remarkably simple, namely, that the meaning of a word can be expressed by relating it to other words. This leads to the concept of *word senses*—a word may have many meanings that depend on the context in which it is used.

Quillian found that to *recognize* the meanings of words it is adequate to find the relations between them. However, for another task this conception of meaning might be less appropriate. For example, in the game "Twenty Questions" one may know many things about a word—that it denotes a common household item, the item is wooden, and so on. One may know everything about a word that would go into defining its meaning but still be unable to guess what it is, that is, to recall it. Quillian makes the distinction between *recognition memory* and *recall memory* for the meaning of words. His model is concerned with the former; recall memory is not considered.

The tasks Quillian chose to implement using semantic memory were *comparison* of word meanings and *expression* of the comparisons in English. Both were motivated by linguistic theory contemporary with Quillian's research, which subordinated meaning to syntax in search of rules to produce "all and only" grammatical sentences. In contrast, Quillian regarded semantic memory as primary to language production and syntax as secondary. Thus, he chose tasks to show that this new conception of language production could, in fact, both produce language and understand it.

The Associative Structure of Quillian's Semantic Network

Quillian's model is an associative network of nodes that represent concepts and arcs that represent relations between the concepts. When one is asked to say all one knows about a concept, for example, *machine*, a string of associations often results: A machine does work, has moving parts, is used to convert energy, and so on. *Machine* is associated with *energy* via the concept *convert*.

Word definitions have an associative structure. The set of associations and concepts that make up a definition is called a *plane* (see Fig. E1-1). The

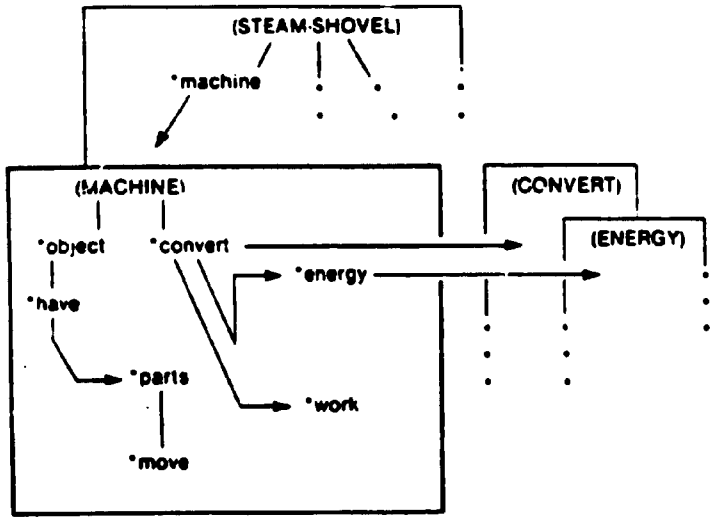


Figure E1-1. Illustration of planes for *Machine*, *Steam-shovel*, *Convert*, and *Energy*, showing type-token links.

concept being defined, called the *type node*, appears at the top, and the starred words beneath it are called *token nodes*. They are *instances* of the type nodes of other planes that are connected to their type nodes by arcs. (In Fig. E1-1, these arcs are not filled in for all token nodes; *steam shovel* is a subclass of *machine*, ∗*machine* is an instance of *machine*, and ∗*convert* and ∗*energy* are instances of their associated type nodes.) Every plane contains only a single type node and enough token nodes to define the concept it names. Every plane represents a new concept defined by associations to those previously defined. Planes are linked together, type node to token node, throughout the associative memory.

The utility of the type-token distinction is that it saves space in computer memory. Imagine the size of a memory in which every definition of a particular machine included the entire plane for *machine*, and the planes of its other defining concepts, within its own. A more efficient organization is to have a single plane define *machine* and to connect it to token nodes in all the planes that include *machine* as part of their definition.

Quillian believes that semantic memory should have a large enough selection of arcs to represent the richness of relations between concepts in English, but not so many that the mechanisms required to process the arcs are very complicated. Six kinds of arcs were used, representing the following relations:

1. Subclass/superclass,
2. Modification (adverb, adjective),
3. Conjunctive (a *and* b *and* c),
4. Disjunctive (a *or* b *or* c),
5. 6. Two other relations representing unspecified binary predicates.

Other, more complex schemes for associating nodes have been proposed (see Article XI.E4). In a later publication, Collins and Quillian (1972) describe several other kinds of arcs, representing *proximity* (or *adjacence*), *consequence*, *precedence*, and *similarity*.

Meaning-dependent Tasks in the Model

One important contribution of Quillian's work was providing a simple model of semantic ambiguity. There are two sources of semantic ambiguity: A word may have different *meanings* (e.g., the noun and the verb forms of *plant*), and it may have different *senses* depending on context (e.g., *animal* in the context of species or *animal* in the context of untamed). Quillian's model is able to find many of the senses of words.

When the model is presented with two words to compare, it starts to search outward from the planes representing the words in its memory. The type nodes of the planes are called the *patriarch* nodes. The program alternately examines nodes emanating from each patriarch. Each node is tagged with a double label, one part containing the name of the patriarch and the other the name of the last node examined (the immediate ancestor). Searching continues until the path from one patriarch "bumps into" a node labeled with the name of the other patriarch. At that point, a path from one patriarch to the other has been completed. Its nodes represent the concepts that relate the two patriarch concepts, the raw material of a comparison. A program that expresses this conceptual pathway in English is summoned and produces a crudely expressed comparison.

It is likely that there is more than one path between two words. In fact, Quillian estimates that in a network of the 850 words of basic English, at least 10 nontrivial paths could be found relating any pair. Each of these constitutes a *sense* in which one word is used in the context of another. For example, the pair *man*, *business* yields the following comparison:

```
Man1 is person, and
Business can be activity which person must do work.
```

Also, the program discovers the generic sense of *man*:

```
Man2 is man as group, and
Business is question for attention of group.
```

Thus, *man* used in the context of business has two meanings. In the context of *live*, another sense emerges:

```
Man is animal, and
To live is to have existence as7 animal.
```

Also:

```
Man is a live+being2.
```

Although this version of the model contains less than 60 definitions, it still produces interesting comparisons.

Quillian notes that the breadth-first search (Article II.C1, in Vol. I) between nodes is a form of inference. The relations between nodes *within* a plane are entered by the coder who defines it. In constructing a definition, the coder makes pairwise associations between a plane and (through type-token links) the other planes that define it. Any path between planes that encompasses more than a single type-token pair is a novel conceptual link discovered by the model:

> While a path lying completely within one plane (except for its terminal points) amounts only to a representation of some piece of the information put into memory, a "plane-hopping" path represents an idea that was implied by, but by no means directly expressed in, the data that were input. (Quillian, 1968, p. 240)

Empirical Tests of Quillian's Model

Inference was an important concept to Collins and Quillian (1969) in their research on the psychological validity of the semantic network model. They sought to prove that human memory, like their semantic memory, obeyed the organizational principles of hierarchy and economy. Figure E1-2 represents a hierarchical tree of information about animals. The lower nodes constitute proper subsets of upper nodes; this is the principle of hierarchy. Note that properties of nodes are not repeated at each node at which they apply but at the highest possible node above all the subsets to which the property applies. The properties of subsets are then *inferred* from the superordinate nodes at which they are stored. This is the principle of economy.

For example, although a canary is feathered, this information is stored with the ancestor of the set of feathered things, the concept *bird*. Higher still, stored with the concept *animal*, is the information that a canary is ambulatory. The knowledge that a canary is ambulatory is achieved by inference: A canary is a bird; a bird is an animal; animals are ambulatory; thus, by inference, canaries are ambulatory.

Collins and Quillian reasoned that predictions can be made to test whether the principles of hierarchy and economy hold for human memory. The first of

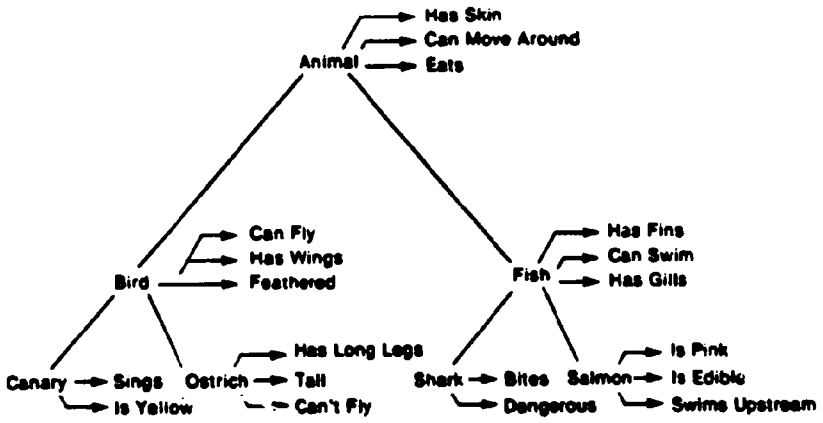


Figure E1-2. A hierarchical memory structure (from Collins and Quillian, 1969).

these concerns the hierarchy principle: Since it requires more inferential steps to confirm a proposition like *A canary is an animal* than a tautology like *A canary is a canary*, humans should require more time to confirm the former than the latter. They should need intermediate amounts of time to confirm propositions requiring intermediate-length chains of reasoning, such as *A bird is an animal* or *A salmon is a fish*. In fact, *reaction-time* data support this prediction:

Proposition	*Time to confirm (in seconds)*
A canary is a canary.	1.0
A canary is a bird.	1.17
A canary is an animal.	1.35

These reaction times have been replicated for similar tasks (Conrad, 1972) and support the hypothesis that semantic memory is organized hierarchically.

Collins and Quillian used a similar experiment to test the economy principle. They predicted that *A canary can sing* should require less time to verify than *A canary has skin*, with intermediate propositions requiring intermediate time. They found:

Proposition	*Time to confirm (in seconds)*
A canary can sing.	1.31
A canary can fly.	1.38
A canary has skin.	1.47

They presented this evidence in support of the economy principle. The alternate hypothesis, that a property common to a superset is stored with each member of each subset, is ruled out by the reaction-time data: If a superordinate property, like having skin, is stored with each subordinate node, for example, *canary*, it should take no longer to verify that a canary has skin than that it can sing.

Although Collins and Quillian's data support the economy principle, there is evidence that the increasing reaction times can be explained in other ways. Conrad (1972) found that the time required to verify a property was proportional to its familiarity, *not* to the hierarchical distance between a property and the noun it is associated with in a proposition. An alternative to the economy principle is that "properties are stored in memory with every word which they define and can be retrieved directly rather than through a process of inference" (p. 154). Conrad explains the differences in reaction time as a function of the familiarity of the words. When familiarity was controlled, and the experiment run again, no differences in reaction time as a function of the presumed hierarchical placement of the property could be found. However, the effect of position in hierarchy for superset-subset sentences was replicated.

The status of the economy principle is unsure. The hierarchy principle has more support, but Collins and Quillian's model of sentence verification leaves a number of phenomena unexplained. For example, it does not account for how false sentences (*Fish can play hopscotch*) are disconfirmed. Unfortunately, the reaction times obtained for disconfirming negative sentences are difficult to interpret. It is difficult to tell whether this is because of a failing in the model or because reaction time is an inappropriate tool for examining this kind of model.

Conclusion

Since Quillian's pioneering work, semantic nets and other associative representations (e.g., frames) have become part of the language of AI. Although Quillian developed his model as a representation of linguistic knowledge and was motivated largely by issues in linguistics, semantic nets have been generalized to representations of many other kinds of knowledge. Several issues raised by Quillian have been examined in detail in AI. The issues of modes of inference, inheritance of properties, and the numerosity and semantics of arcs are discussed in the domain of knowledge representation (see Chap. III, in Vol. I: also, Brachman, 1978). In psychology, the model was subjected to empirical analysis and several other associative models were developed. Three will be discussed in the succeeding articles.

References

The best paper on Quillian's model is his own in Minsky (1968).

E2. HAM

IN THEIR BOOK *Human Associative Memory* (1973), psychologists John Anderson and Gordon Bower present an associationist theory of human long-term memory (LTM). Aspects of their theory have been implemented in a computer simulation called HAM that parses simple propositional sentences and stores the parsed sentences in its memory. HAM also answers simple questions. Its abilities are limited, but intentionally so, in that Anderson and Bower have eliminated the mnemonic strategies and tricks that result in smart memory performance in humans. Their goal was to model the *strategy-free component* of human long-term memory and to explain the vast experimental data on the subject. With respect to this goal they write:

> Why not add some more inferential routines to increase the intelligence with which it (HAM) answers questions? We started down this enticing, seductive path; but we slowly came to the realization that this was no way for experimental psychologists to proceed.... The end product of such an enterprise would appear to be thousands of lines of program that described the countless heuristics, procedures, tricks, and rules that the human has learned in his lifetime. We would have translated one incomprehensible mass of particulars, the human mind, into another incomprehensible mass, a computer program. But the task of science is surely to reduce particulars to general laws rather than translate particulars from one idiom to another. (p. 145)

Anderson and Bower assume that long-term memory is strategy invariant; the strategies that are obviously used to remember things are, they assume, imposed by an *executive component* of cognition. LTM is thought to be much simpler than the experimental literature suggests, because much of the literature does not separate out the effects of mnemonic strategies on memory performance. Memory experiments that use single words or nonsense material as stimuli are considered especially likely to have their results complicated by mnemonic strategies, because these materials are more easily remembered with some strategy than without. Consequently, most of Anderson and Bower's research concerns memory for sentences or phrases that are apparently less likely to evoke mnemonic strategies.

Anderson and Bower chose question-answering as the task environment for HAM. This may be the simplest task on which to examine a memory model, since it requires only storage, retrieval, and rudimentary parsing functions.

HAM accepts two kinds of inputs, facts and questions, which it parses into input structures (described below). To facilitate parsing, inputs are made only in a natural subset of English. We will not consider HAM's parser in this article other than to say that it is a top-down, left-to-right, predictive parser;

we refer the reader to Chapter 8 of Anderson and Bower's book (1973) for more details.

The parameters of memory that interest Anderson and Bower are:

1. The set of possible memory structures,
2. The set of possible inputs to memory,
3. The set of possible outputs from memory in response to probes,
4. The set of possible probes,
5. The encoding process by which the structure of memory is modified to record new information,
6. The decoding function by which the structure of memory is probed to determine what is recorded there.

Some assumptions are made about these parameters. First, the only allowable input structures are facts and questions. The latter are called probes. It is assumed that probes are always parsed into the same input structure, that the encoding function always matches the input structure to memory in the same way, and that the same output will be generated to a probe.

Representation of Knowledge in HAM

All knowledge in HAM is represented as *propositions*, encoded in binary trees. For example, the structure of *In a park a hippie touched a debutante* is shown in Figure E2-1. The numbers identify nodes in memory; the labels are interpreted as follows:

Label	*Interpretation*
C	Context in which Fact is true
F	Fact
L	Location
T	Time
S	Subject
P	Predicate
R	Relation
O	Object
E	Set membership

A proposition tree may also consist of a fact without a context. In this case, it always has the subject-predicate form; sometimes the predicate is just a single concept (see Fig. E2-2).

The relation-object pair is used to express implicit or explicit causality, among other things. *Cause* is illustrated as a relation in Figure E2-3. This

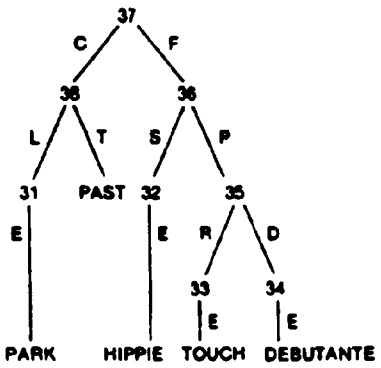


Figure E2-1. HAM structure for *In a park a hippie touched a debutante.*

structure represents the sentence *John opened the door with the key.* It includes an implicit cause, namely, turning the key caused the door to open. This tree is more abstract than the one shown in Figure E2-1, because it does not show the *terminal quantifiers* leading to the terminal nodes of the tree. *Set membership*, labeled "E" above, is one of three terminal quantifiers. It is used when the terminal concept is a member of a set, such as the set of debutantes. A *generic* link is used when the terminal node denotes all members of a set, for example, the entire class of dogs in *All dogs chase some cats.* A subset link is used to indicate that the terminal node denotes neither an entire class nor a single member, but a subset of a class. *Cats* in the previous sentence takes a subset link. These links give HAM the representational power of second-order predicate calculus (Anderson and Bower, 1973, pp. 167–169; however, the reader is referred to Anderson, 1976, pp. 165–[illegible] for a criticism and reworking of the terminal quantifications of HAM).

Properties of HAM's Knowledge Representation

Anderson and Bower (1973) specify the properties of their memory structure as a set of postulates:

> *Symmetry:* If an associative link exists between two nodes, then an inverse link also exists. Concretely, if one knows a relation between two objects, one also knows the inverse of that relation.
>
> *No-forgetting:* Once a structure is incorporated into memory, it cannot disappear from memory. Therefore, forgetting must occur by losing *access* to the information in the structure, not the information itself. (For more on this view, see Article XI.D.)

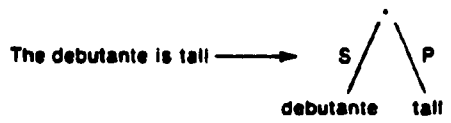


Figure E2-2. Simple subject-predicate structure.

First Empiricist Postulate: There is no innate knowledge in the form of associations between memory nodes. All associations are formed in response to inputs.

Second Empiricist Postulate: Concepts (nodes) similarly are acquired only through inputs. However, this can lead to the idea that HAM is initially empty, which Anderson and Bower explicitly reject. They postulate a *base set* of *simple ideas* that are present in HAM at its birth and upon which more complex ideas are built.

An Example of HAM in Operation

HAM accepts input sentences (indicated by ** below), builds associative structures of them in memory, and answers questions about them:

```
** In a park a hippie touched a debutante.
```

HAM responds by building and printing the structure (shown in Fig. E2-4) that corresponds to this assertion. It is the same structure as shown in Figure E2-1.

```
** Who was touched by the tall hippie?
The tall hippie--which one?
```

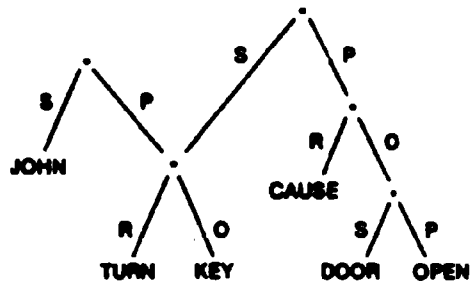


Figure E2-3. Implicit *cause* in the sentence *John opened the door with the key.*

HAM does not know of any tall hippies. It is told that the current hippie is tall.

** The hippie was tall.

HAM incorporates this new knowledge and prints out its structure. The new knowledge structure is illustrated in Figure E2-5; for clarity, we have shown it connected to the structure from Figure E2-4, although HAM would not print all these nodes, but only the new nodes—45, 46, 47, 48, and 32—and the associative links between them.

** Who was touched by the tall hippie?

HAM can now answer the question.

The debutante.

This example illustrates HAM's operation. When input sentences are typed in, they are parsed into tree structures. If the input material is an assertion like *The hippie was tall*, HAM incorporates it into memory by finding and merging common nodes in memory and in the input. In this case, part of the input structure matches node 32 of HAM's memory, corresponding to the *hippie* concept. HAM incorporates the input structure into memory by joining it to node 32, as shown in Figure E2–5. Thus, HAM learns by associating new knowledge in the form of input trees with old knowledge already in memory.

If the input sentence is a question, the parser generates an input tree that may be missing a part. This kind of tree is called a *probe*. For example, the question *Who was touched by the tall hippie?* is parsed into a probe of the form *The* [blank] *was touched by the tall hippie;* see Figure E2–6. To answer questions, HAM quite literally fills in the blanks. It searches its memory for a structure like the probe that has a node instead of a blank. This node is the answer to the question. In this case, the probe in Figure E2–6 matches the memory structure in Figure E2–5, and the node corresponding to the blank is *debutante*.

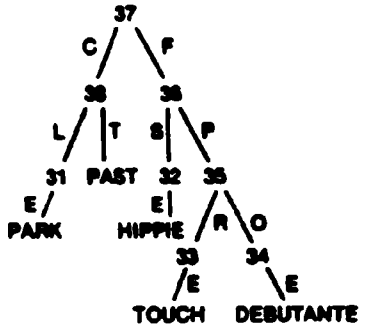


Figure E2–4. HAM structure for *In a park a hippie touched a debutante.*

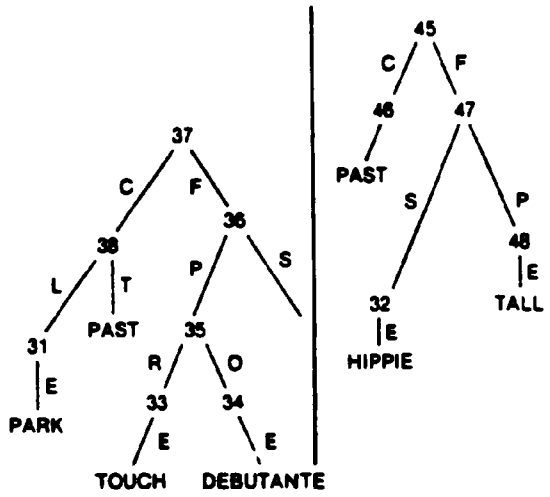


Figure E2–5. Illustration of how HAM incorporates the fact *The hippie was tall* into its memory.

HAM matches input trees to extant memory structures to associate new information with old and to answer questions. Its operation becomes more complicated when *partial matches* are involved. The 1973 version of HAM was run in two modes. The mode illustrated in Figures E2–4 and E2–5 has HAM *not* accepting a partial match in the case of the tall hippie. The program wants to be told explicitly that the tall hippie in the input tree and the hippie in memory are the same hippie. In the other mode, HAM accepts partial matches. For example, it would answer the question *Who was touched by the tall hippie?* by matching the probe tree in Figure E2–6 to the memory

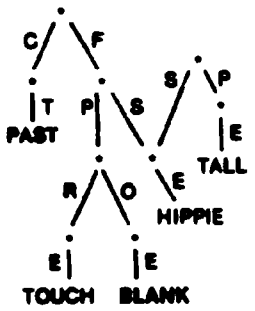


Figure E2–6. Probe tree for *Who was touched by the tall hippie?*

structure in Figure E2–4. It would not be necessary to spell out that the hippie was tall, as in Figure E2–5. (Partial matching is discussed further in Anderson and Bower, 1973, pp. 242–246.)

From these examples, one can see that the matching process, MATCH, is fundamental to HAM's operation. MATCH is simple in conception. First it finds nodes in memory that correspond to the terminal nodes of an input structure, and then it attempts to find links in memory that correspond to the links in the input structure. In other words, MATCH finds *paths* between input terminal nodes that correspond to paths in memory. A memory path and an input structure path are considered equivalent if they have the same number of links and the same *sequence* of relations labeling the links.

HAM searches for paths in memory from all of the input terminal nodes in parallel. For example, after matching the terminal nodes of the probe (Fig. E2–6) to nodes in the memory structure (Fig. E2–5), MATCH would search from each of the nodes (*past, touch, hippie, past, tall*), in parallel, to determine whether the paths that connect them are identical in memory and in the probe. However, if a node has more than one path emanating from it (*hippie* has two), they are searched sequentially. Consequently, the time required to establish that a node falls on a path is proportional to the number of associations it has—the number of paths it belongs to. This is called the *fan effect*.

HAM knows many facts, and a given terminal node like *hippie* is likely to be part of several trees. In this case, *hippie* is associated to nodes 36 and 47 in memory by means of a *subject* link. The nodes associated with each node by a link are stored in a *GET-list* for the node and link. The *hippie* node in Figure E2–5 would have a single GET-list with two members for the *subject* relation. One can imagine other associations made with other links (e.g., *object*) resulting in other GET-lists. To reduce search, MATCH follows only the links from a node in memory that have the same label as the links from the corresponding node in the input tree. If the *hippie* node in memory were connected to other structures by an *object* link, MATCH would not search them, since the input structure it is matching to memory has only *subject* links emanating from *hippie*.

Search is further speeded by using *recency* information. The members of the GET-list are examined in the order of most recent mention. Moreover, HAM will not necessarily search all members of a GET-list; it may be too long. This leads to the sole mechanism of *forgetting* in HAM: An association between two nodes that has not been mentioned in a long time will drop farther and farther down the GET-lists for both nodes, thereby increasing the probability that HAM terminates its search from one of the nodes without finding the association with the other.

Search can be speeded to some extent by these methods, but a node may still be a member of many paths. *Hippie* could be the subject of dozens of sentences, and MATCH would have to check each, serially, to see if an input

structure corresponds to one of them. The number of associations a node has is called its *fan-out;* since fanning nodes are searched sequentially, the fan-out contributes to the amount of time required to answer a question. This property is the basis for reaction-time experiments with human subjects. HAM predicts that it should take humans longer to process memory concepts with a high fan-out than those with a low one. See the following article on ACT for an explanation of the Sternberg effect in terms of the fan effect.

To summarize, MATCH associates terminal nodes in the input structure with corresponding nodes in memory and then starts a parallel search from these nodes for paths between them that are equivalent to the paths between the terminal nodes of the input structure. To do this, it examines the label of each link emanating from a node in the input structure and searches the appropriate GET-list associated with the corresponding node in memory. The GET-list may not be searched completely and thus associations between nodes may appear to be lost, which accounts for forgetting in HAM. The position of a node on the GET-list is a function of how recently it was mentioned, so that old associations are more likely to appear to be forgotten than recent ones. Lastly, the nodes on a GET-list are searched serially, so that a large GET-list can take a long time to search.

Conclusion

Anderson and Bower have a strong commitment to empirical data about human memory. The HAM model was designed as a parsimonious and operational explanation of a wide range of results. It also made a number of predictions that were tested with the standard experimental methods of cognitive psychology. The individual results are voluminous and of interest primarily to cognitive psychologists; none of the particulars is presented here. However, the general result is especially important: A wide range of memory tasks can be modeled by a *strategy-free* process. Although humans use sophisticated strategies to remember difficult (often meaningless) material, the study of long-term memory is simplified by assuming that the strategies overlay a relatively simple mechanism common to all memory performance. The MATCH process is such a mechanism, and in experiments in which the utility of mnemonic strategies is reduced, it predicts many interesting empirical results.

References

Anderson and Bower's 1973 book *Human Associative Memory* provides a detailed account of the HAM model and of empirical tests of the model. The first four chapters of the book are interesting reading, although they are background to HAM, not a discussion of HAM itself. They discuss philosophical approaches to the study of memory, linguistic theory, and other models of memory.

E3. ACT

THE ACT system was built by John Anderson following his work on HAM (see Article XI.E2). There are many points of overlap between HAM and ACT, but there are also fundamental differences. Most significantly, ACT is intended as a general model of cognition, while HAM is a model of human memory. HAM answers questions and learns new information; ACT does more, in that it can be programmed to perform a wide variety of cognitive tasks. In addition to its long-term memory, ACT has a short-term *working memory* of active concepts and a programmable *production system* that brings about changes in working and long-term memories. Common to HAM and ACT are certain features of long-term memory; for example, strategy invariance has been carried over to ACT, and so has the propositional representation of knowledge, although modified in some details.

Overview of ACT

ACT has a long-term memory component and a user-programmable procedural component. The memory is a propositional associative network made up of nodes representing concepts and arcs representing relations between the concepts. ACT's memory is not very different from HAM's (discussed in Article XI.E2), so it will not be described in detail here.

An important feature of ACT's memory is that only parts of it are *active* at any time. Activation can spread through the network as nodes activate adjacent nodes. The time required to activate the neighbors of an active node depends on its *fan-out*, that is, the number of nodes connected to it. ACT attends to a limited number of active nodes. Those that are not marked for attention are eventually made inactive; those that are marked for attention are put in a first-in, first-out buffer called the ALIST. They may displace older nodes, because the ALIST has a capacity of just 10 items. In this article, the ALIST will be called the *working memory*.

The programmable, procedural component of ACT is a *production system*. Each production has a condition part as well as an action part that is invoked if the condition is true. In ACT, all conditions test for a conjunction of features of memory, and all action parts specify a change to be made to memory. The conditions of productions can examine only the active part of memory. A number of productions may be activated by the state of memory, in which case each of them has a probability of being executed.

An Example of ACT

Anderson shows how ACT can be programmed to perform the Sternberg *memory-scanning* task (Sternberg, 1969). In this task, subjects are presented

with a list of numbers, for example, 4 9 1 3, and a *probe* number, which may or may not be on the list. Sternberg's result is that, if the probe number is in the list, then the amount of time required to confirm it increases, by .038 seconds, for each number on the list. Curiously, the serial position of the matching digit is irrelevant; the time required to confirm the presence of a probe in a list of numbers is independent of where the probe occurs in the list. Sternberg originally explained this effect in terms of a *serial exhaustive scanning model*, in which the list is kept in working memory and a *comparator* compares the probe digit to each list element. The comparison process was thought to be exhaustive, meaning that all list elements are scanned, even if a match to the probe has already been found. (This paradigm is discussed in detail in Crowder, 1976, pp. 354-366.)

Anderson offers a different explanation in terms of ACT. When the list of numbers is presented, a structure is built in memory to represent it. In the case of the list 4 9 1 3, a node called LIST is connected to four nodes, 4, 9, 1, and 3, by the relation CONTAINS, as shown in Figure E3–1. In ACT's memory, the LIST node is connected to four others and ultimately to four numbers by the relation CONTAINS. The LIST node has a fan-out of four, since four links emanate from it.

The first production for the Sternberg task is:

P1. State = Ready → State and List. .

It says that if ACT is in the *ready* state, the next step is to rehearse the state and the list. In the context of memory, rehearsal means repeating something over and over to keep it in memory, much as we do with telephone numbers. Production P1 brings about rehearsal behavior by the simple device of putting on the ALIST the condition to satisfy P1 again. Production P1 is satisfied whenever state = *ready*; when it is executed, it sets the state to *ready* and

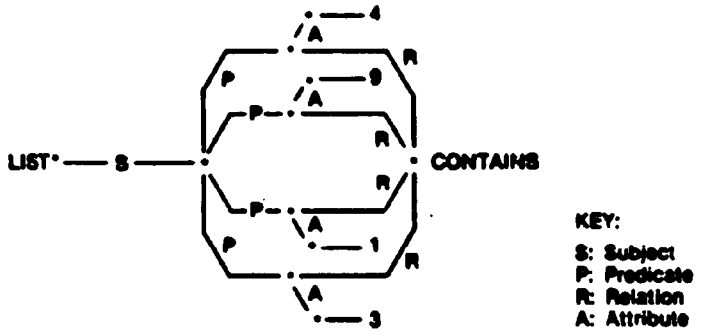


Figure E3–1. ACT memory structure for a list of numbers.

puts the LIST node on the ALIST again. This potentially infinite iteration continues until another production is satisfied.

The second production, P2, tests whether a probe digit has been given. If it has not, then P2 cannot have any effect and ACT will continue to rehearse. If it has, then P2 changes the value of the state variable from *ready* to *test* and puts the probe digit on the ALIST with the state variable:

P2. State = Ready and Probe given
→ State = Test and Probe digit.

The third and fourth productions check for the presence of the probe digit in the list and signal their findings. They then reset the state variable to *ready* for the next problem:

P3. State = Test and List contains Probe
→ Signal "Found it" and State = Ready.

P4. State = Test and List does not contain Probe
→ Signal "Not there" and State = Ready.

This simple production system and the idea of spreading activation in memory account for the Sternberg result that reaction time to identify a digit increases with the number of digits in the list. At the beginning of a trial, ACT has encoded the list in memory as described above, and the LIST node is put into working memory (see Fig. E3-2). It is active, but the nodes emanating from it are not; they must be activated by following links from the LIST node in working memory into long-term memory.

With working memory in the state shown in Figure E3-2, production P1 applies. It will rehearse the contents of working memory until a probe digit is given. When this happens, the value of the state variable is changed to test,

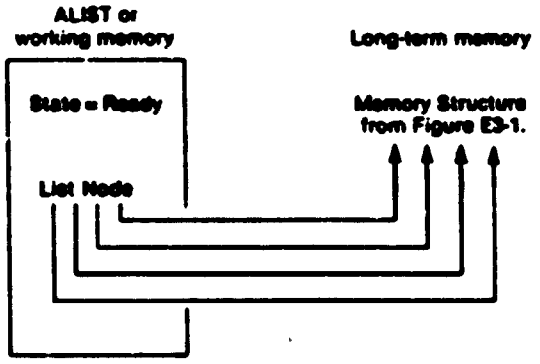


Figure E3-2. Illustration of working memory, showing links into long-term memory.

the state necessary for P3 or P4 to apply. The other condition for P3 is that the list contains the probe digit. Since the numbers on the list reside in long-term memory, they must be activated for P3 to check them. The fan of the LIST node determines how long it will take to activate the nodes connected to it: as the number of links from the LIST node increases, it takes ACT longer to search them. This is a slightly different explanation of the Sternberg effect.

Instead of serially scanning a list of numbers in working memory, ACT activates the memory structure representing the numbers; the amount of time required to do this depends on the fan-out of the LIST node. This determines the reaction time in the Sternberg task.

Performance in ACT

ACT is a highly dynamic system. Its *focus of attention* changes as nodes in long-term memory are activated and put in working memory, as other nodes are pushed out of working memory, and as nodes are *damped* in long-term memory and become inactive. There is a constant fluctuation of activity that is complex and nondeterministic due to the probabilistic nature of spreading activation.

Limitations were imposed on ACT to make it resemble human cognition more closely. Some of these are:

1. The parameters that affect spread of activation are extremely important to the operation of the system because nodes must be active to get to working memory, where productions operate on them. The fan of a node is one such parameter. Others include how far activation must travel in the memory network, how frequently nodes in memory are damped, and how *strong* the links are between nodes.
2. The ALIST, or working memory, is of limited size, so ACT attends to just a few concepts at a time.
3. Only some productions in ACT are applicable at any given time. There are strategies for determining which are applicable and for deciding among them. The strategies affect the amount of time ACT takes to perform a task.
4. When new information is added to ACT, it has only a probability of being remembered.

Learning in ACT

There are four methods for learning in ACT. *Designation* refers to telling ACT something, for example, a proposition or a production rule. *Generalization* and *discrimination* are two methods for automatically generating new production rules. The fourth learning method, *strengthening*, is a *reinforcement* procedure.

The first method, *designation*, is the simplest means of adding information to ACT. It is the method that was used in HAM. The second method, *generalization* of productions, works by replacing constant terms in the conditions of two productions by variables. To avoid creating terms that are too general to be interesting, ACT will not replace more than one-half of the constants of the smallest condition. *Discrimination*, the third learning method, produces two or more productions from one with too many variables in its condition. It does so by instantiating the variables. Discrimination applies whenever ACT gets feedback that a production is too general.

Generalizations and discriminations of productions do not replace the original rules: rather, they exist with them. A generalization will apply whenever either of its original productions applies but will have the same effect as both. However, ACT has a conflict-resolution strategy that favors executing specific rules before more general ones, so discriminations of general rules, or the rules from which a generalization has been formed, have precedence over generalizations.

Associated with each production is a *strength* that is used to resolve conflicts when several productions are applicable. *Strengthening*, the last of the four learning methods, reinforces productions by increasing or decreasing their strength. If a production is found to be applicable, its strength is increased by a constant number. However, its strength is decreased by 25% if its execution leads to a mistaken conclusion. Negative strengthening is therefore more effective than positive. Strengthening also applies to productions that are consistent with other applicable or misapplied productions. (A production is consistent with another if its condition is more or less general but its action is the same.)

Conclusion

ACT is a general framework in which cognitive performance is simulated. It is not custom-built to perform a particular task, unlike most of the systems discussed in this chapter. (MEMOD, another general system, is discussed in Article XI.E4.) Anderson considers ACT's design to be psychologically plausible; he goes to lengths to present the "predisposing biases" that motivated design decisions in terms of the psychological literature. Moreover, ACT makes reasonable predictions about human behavior in experimental situations. ACT can be considered a theory, in the sense that it makes predictions, and a programming language, or package, in the sense that it provides an environment for building psychological models.

References

Anderson has published a lengthy book on the ACT system (1976) that includes chapters on the structure and behavior of ACT, spreading activation

in memory, learning, and language comprehension. It is an exhaustive treatment, in which Anderson presents not only the ACT system but also the theoretical motivations for it. The book is reviewed by Wexler (1978), and a reply to the review can be found in Anderson (1980). The review and reply are worthwhile reading for those interested in cognitive science, since they are two different positions on how a science of mind should proceed.

E4. MEMOD

THE LNR research group, named for Peter Lindsay, Donald Norman, and David Rumelhart, is engaged in the ongoing development of a general model of human long-term memory called MEMOD. Of the five memory models discussed in this chapter, MEMOD may be the most ambitious (ACT, discussed in Article XI.E3, is the other candidate) because of its scope and because of LNR's basic tenet that a single system accounts for cognition:

> One system has to be capable of handling the representation and processing issues in syntactic and semantic analysis of language, in memory, perception, problem solving, reasoning, question answering, and in the acquisition of knowledge. (Norman, Rumelhart, and the LNR Research Group, 1975, p. 160)

It is a major goal of the LNR group that the MEMOD system should be a general knowledge-representation system, that is, one that can represent any kind of knowledge. Until quite recently, however, it was used primarily to represent linguistic knowledge. Accordingly, the MEMOD system has three main components: a *parser*, which is based on an augmented transition network (ATN; see Article IV.D2, in Vol. I); a *node space*, which is a semantic-net representation of world knowledge; and an *interpreter*, which performs operations on the node space. The node space represents both *declarative* and *procedural* knowledge; node-space structures represent facts about the world and also specifications of operations to be performed in the node space by the interpreter. Because it is not a passive repository of knowledge but contains procedures that manipulate knowledge, the node space is called the *active structural network*, or ASN.

In this article, the design of the active structural network is sketched briefly, followed by a more formal discussion of how concepts and events are represented. Here, the role of the interpreter will be more obvious. We will not consider the parser at all, since ATN parsers and case grammars are dealt with in Chapter IV (in Vol. I) on understanding natural language.

The Active Structural Network

The design of the active structural network was constrained by a number of goals arising directly from the natural-language applications intended for the MEMOD model. Briefly, these were:

1. *Completeness.* The model must be able to represent *any* knowledge of any type, including nonlinguistic knowledge.

2. *Extendability.* The model must be extendable whenever new information is available. If, for example, the model learns that to saunter is not merely to walk but is to exhibit some degree of indolence, it must be able to incorporate this information.

3. *Invariance under paraphrase.* Expressions that have the same meaning should have the same underlying representation in the ASN regardless of how they are stated at a surface level.

4. *Preservation of overlap in meaning.* The representation of words and larger units of meaning in the ASN should reflect the possibilities of synonymy, partial overlap, and no overlap in meaning. Meanings that overlap, such as *stroll* and *saunter*, should have common components in their representations. Unrelated words should not.

5. *Continuity.* In a psychological model of knowledge, words with similar meanings should have similar structures, and a small change in meaning should not cause a major change in its representation. Similarly, concepts that have very different meanings should have very different representations.

Semantic Decomposition and Case Structure of Predicates

The technique employed by the LNR group to satisfy these goals is *semantic decomposition* of words (or more generally, concepts) into primitive elements called *predicates* (see Article III.C2, in Vol. I, for a detailed discussion of semantic decomposition). For example, they identify four classes of predicates—*stative*, *change*, *causative*, and *actional*—that can be combined to yield different verb meanings.

Stative predicates. The stative component of a verb indicates that a state of the world holds over some time period. One of the stative predicates in the MEMOD system is LOC. It takes four arguments, the last two of which are optional:

```
LOC[object, at-loc, (from-time), (to-time)] .
```

A semantic-net representation of the LOC predicate shows the LOC node linked to four argument nodes, as shown in Figure E4–1. Here, a network structure is shown to represent the sentence *A stadium was located in the park from 1956 to 1963.* In addition to the LOC node, this figure also shows nodes representing the concepts of *stadium*, *park*, *1956*, and *1963*. A point of notation is that the angle brackets and parentheses in this diagram denote *tokens*—or copies—of concepts and predicates, respectively. A token represents a concept in some context; a dictionary of *type*, or original, nodes is also maintained, and token nodes are linked to them; see Article XI.E1 for a discussion of the type-token distinction.

Change predicates. A verb like *move* can be represented as a CHANGE predicate taking two LOC predicates as arguments, as shown in Figure E4–2,

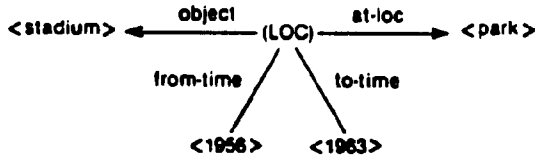


Figure E4–1. The LOC component of the verb *located* in the sentence *A stadium was located in the park from 1956 to 1963.*

which represents the sentence *The team moved from the stadium to the training camp in May.*

Causative predicates. One can imagine how another verb, say, *push*, might be represented by a structure like the one shown in Figure E4–2, but predicated with a causative; that is, to push is to CAUSE to move from one location to another. Figure E4–3 represents the concept of a person causing an unspecified object to move from one unspecified place to another at an unspecified time. It is the skeleton of a *cause-to-change-location* verb; the reader can think of numerous verbs that have this general structure.

Actional predicates. Consider this example in the context of the design goals discussed earlier. Semantic decomposition is a representational tool that guarantees that similar meanings have similar structures. The structure above is common to several verbs with overlapping meanings: *push, shove, carry, pull, transport*, and so on. The *actional* predicate is instrumental in making finer distinctions in meaning; however, LNR has done little work with actionals, and generally the primitive predicate DO is used.

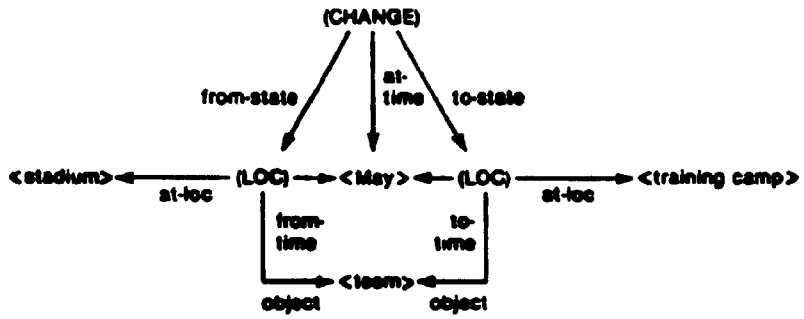


Figure E4–2. ***Move*** **consists of CHANGE and LOC predicates; it is a CHANGE in LOCation—as in *The team moved from the stadium to the training camp in May.***

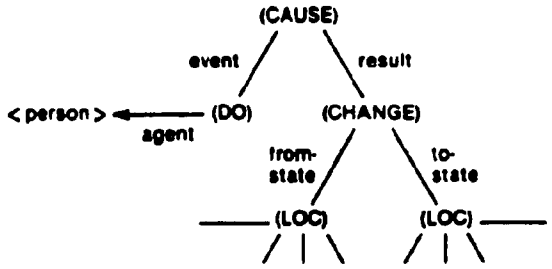


Figure E4-3. Skeleton of a verb with CAUSE, CHANGE, and LOC predicates organized to represent the concept of causing a change in location, as in *push*, *pull*, and *carry*.

Once a word is defined, it is stored in MEMOD's dictionary as a type node and can be used in more complex structures, as in:

```
<lion> ← (CARRIED) → <antelope>
```

A final point, before proceeding to a more formal description of knowledge representation in MEMOD, is that predicates have a *case* structure (discussed in Article IV.C4, in Vol. I). For example, LOC has two necessary and two optional arguments:

```
LOC[object, at-loc, (from-time), (to-time)] .
```

This facilitates parsing. When the parser recognizes a predicate or a verb, it can make predictions about what kinds of words to expect next on the basis of knowing the verb's arguments.

Encoding Concepts, Events, and Episodes

In a 1972 paper, Rumelhart, Lindsay, and Norman specified four categories of rules for constructing complex knowledge structures from the simple ones we have already considered:

1. Rules of formation for concepts,
2. Rules of formation for relations,
3. Rules of formation for propositions,
4. Rules of formation for operators.

Concepts are objects, for example, *lion* and *stadium*. Relations are the names of associations that may hold among concepts, for example, HIT[actor, object, instrument]. Propositions are instantiations of relations, for example, HIT[John, ball, bat]. Operators, the last group, are of two varieties,

prepositional and relational. The former modifies concepts of time or location to generate new concepts:

```
before(noon)  or  under(water) .
```

The latter modifies relations:

```
slowly(walk)  or  very(big) .
```

LNR gives five rules for forming concepts. First, an existing concept can be qualified. This corresponds most closely to the action of adjectives. A qualified concept has a node of its own; for example, the node *lamb* is defined as *young*(*sheep*). Second, quantification of concepts can yield new concepts. as when *crowd* is defined to be *many*(*persons*). Third, new concepts of location and time can be derived from prepositional operators. Fourth, concepts can be conjoined to form new concepts; for example, *and*(*dog, cat*) denotes the concept of the class of dogs and cats. Finally, concepts can denote propositions. For example, in the proposition HIT(*John, ball, bat*), there is a concept *hit*, which corresponds to an instance of the general relation HIT in the context of *John* and his *ball* and *bat*.

There are three ways to generate new relations from old. The first is to modify the relation, as with an adverb. For example, the relation *stroll* is defined as *slowly*(*walk*). Another method is to modify one or more of the arguments of the relation. For example, if the relation of walking is defined as:

```
WALK[actor, path, time] .
```

then a new relation, CLIMB, could be derived by specifying that the path should be uphill. Finally, new relations can be generated by conjoining old ones with special conjunctions. BECAUSE is one such conjunction:

```
FLEE[actor, object, time] is defined as
        quickly( GO(actor, from(object), time))
                BECAUSE
                        FEAR(actor, object, time) .
```

Propositions are formed by instantiating a relation with concepts. For example, the arguments of FLEE might be (*Dorothy, lions, always*). The other method for obtaining propositions is to conjoin them with conjunctions like BECAUSE and AND.

Operators are constructed in some of the same ways. New qualifiers are generated from old by applying relational operators to them; for example, *tiny* is *very*(*small*). Relational operators also apply to each other; for example, *partly* is *not*(*completely*).

Sentences that describe events, such as *The lion chased Mary*, can be encoded in MEMOD. Conjoining events by using conjunctions like BECAUSE, AND, THEN, and WHILE allows one to represent complex *episodes*. Graphically,

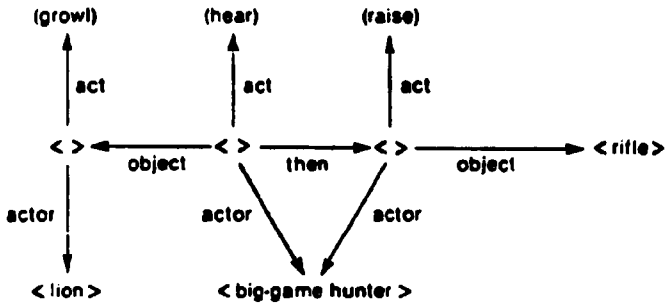


Figure E4–4. A representation of the episode *The big-game hunter heard the lion growl and raised his gun.*

an episode is simply a sequence of event nodes connected by conjunctions. An example of a graphical representation of an episode in which a big-game hunter hears a lion growl and raises his gun is shown in Figure E4–4. (The empty nodes represent tokens of the three events, hearing, growling, and raising.)

A simple propositional sentence can be broken down into a relation and a set of concept arguments. A relation can be broken down further into primitive predicates by semantic decomposition. Rules were discussed here that conjoin and modify concepts, relations, propositions, and operators and that create more complex structures such as episodes. These rules give MEMOD the power to represent episodes of varying complexity. The next section outlines the interactions between these representations and the interpreter.

The Interpreter

Knowledge is supplied to MEMOD in the form of sentences. After these are parsed, the interpreter makes the appropriate changes to the ASN by executing the program associated with each relation in the input sentence. For example, a basic relation built into MEMOD is CONNECT. There is a type node for CONNECT that is linked to a computer program that joins nodes together in the ASN, as shown in Figure E4–5.

If the interpreter encounters a parsed version of the sentence *Connect dog to animal with isa*, it will look up the word *connect;* find that it denotes a built-in program that takes three arguments; *bind* the arguments *dog, animal,* and *isa* to the variables *X*, *Y*, and *Z;* and execute the program. The result is a network structure:

$$\text{dog} \xrightarrow{\text{isa}} \text{animal}\,.$$

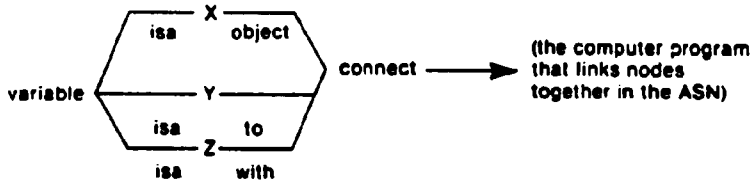


Figure E4–5. Representation of the relation CONNECT.

The CONNECT program was built into MEMOD from the outset. However, it is possible to *define* words by associating programs with them. For example, the LNR group gives the following definition for the word *son*.

> Define son as predicate.
> *the definition frame for son is:* X is son of Y
> *the definition is:*
>
> Connect X to male with sex.
> If age of X is less than 18, then
> connect X to child with isa.
> connect Y to X with parent-of.
> ##

(This represents an interaction with the MEMOD system. The text in ordinary type is entered by the user; MEMOD's replies are in italics.)

The relation DEFINE is itself a built-in procedure that builds structures in the ASN. For example, defining *son* yields a structure that is something like the one shown in Figure E4–6 (which is not exact, since all of the arrows pointing to the node CONNECT would be pointing to the same type node in the ASN).

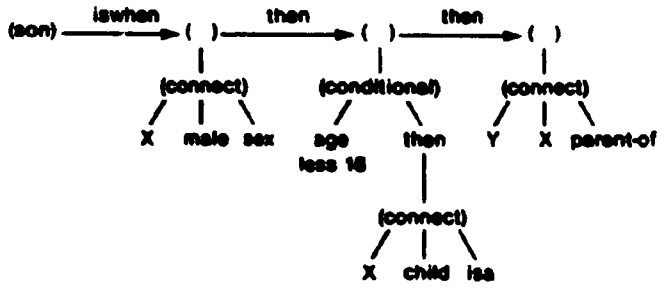


Figure E4–6. Representation of the definition of *son*.

The important thing about this structure is that it invokes changes to the ASN when interpreted. Although its representation is uniform with declarative network structures, it is a representation of a procedure. When the interpreter is given the sentence *Oedipus is the son of Jocasta*, it will create a structure in the ASN representing the facts that Jocasta is the parent of Oedipus and that Oedipus is male. The distinction between procedural and declarative knowledge in MEMOD is obscured by the uniform representation used for both. It appears that there are two kinds of procedural knowledge, built-in programs like CONNECT and definitions that are formally very similar to episodes except for an ISWHEN link. ISWHEN links a node with its definition, and interpreting the definition results in changes to the ASN.

A word like *son*, when defined in MEMOD, carries with it the procedures necessary to make inferences about what it means to be a son; for example, one can infer that a son is male because part of the definition of *son* is a procedure that makes that connection in the ASN. Because definitions carry implicit inferences about what it means to be something, MEMOD can answer many questions. For example, given an appropriate definition, it can say what it means to be a *sandwich*. Here is one definition from the Kitchenworld implementation of MEMOD:

> Define sandwich as recipe.
> *the definition frame for sandwich is:* (subject)sandwich X.
> *the definition is:*
>
> > Place a slice of bread on the counter.
> > Spread preferred spread of X on the bread.
> > Place each ingredient of X on the bread.
> > Place a second piece of bread on the bread.
> > ##

This definition has a network structure similar to those shown above. It is composed of nodes representing simple actions like *place*, which are composed of simpler predicates like CONNECT. To answer questions such as "What containers would be left on the counter after I made a sandwich?" the interpreter executes the sandwich recipe in the ASN. This results in changes to the ASN. For example, containers that were previously associated with *refrigerator* by an IN link may subsequently be linked to *counter* by ON.

Conclusion

MEMOD implements a number of powerful ideas, which were reviewed here. Semantic decomposition, for one, ensures that concepts with similar meanings have similar structures. This was illustrated by a general structure for verbs that mean *to cause a change in location*. In MEMOD, the meaning of a concept is reflected in its structure, its composition of simpler units of meaning.

Verbs and other structures in MEMOD have a case structure, which means that MEMOD knows how many arguments a verb takes and what kinds of arguments they are. There is a grammar for building structures in MEMOD, and rules for building concepts, relations, propositions, and operators were discussed.

Events in MEMOD can be linked by conjunctions. The THEN conjunction is particularly important because it orders events of an episode in time and for the interpreter. Another important link is ISWHEN. It links words to their definitions, which are episode-like procedures for building structures in the ASN.

References

The LNR research group wrote a book called *Explorations in Cognition* (1975). It is a collection of articles by Norman, Rumelhart, and their graduate students on experiments with the MEMCD system and is the most complete and recent review of MEMOD.

F. BELIEF SYSTEMS

IMAGINE a conversation with a person who speaks only facts, the kind of conversation you might have with an official who refuses to give a personal opinion or make a prediction about the future or guess at an explanation for a past event. Or consider the testimony of police officers; they say things like "We were called to the scene at 12:07 A.M. and found the suspect holding two hostages. We succeeded in disarming the suspect without injury. The suspect is now undergoing psychological evaluation." What they do not say is that they believe the suspect is guilty, that they believe he is a doped-up crazy, that they were scared stiff while disarming him, that they sincerely hope he gets the maximum sentence, that holding an old lady hostage is a miserable act of terrorism, and so on. Police officers rightly stick to the facts. At least, they do while on duty. Afterwards, we assume they are as full of opinion, belief, innuendo, prejudice, and emotion as the rest of us.

In this example, the distinction between fact and belief has been amplified to emphasize that much human discourse is in beliefs, speculations, predictions, desires, and so on. The research discussed in this article is concerned with the structure of beliefs, how we reason with beliefs, how beliefs function as prejudices to influence interpretation, and how emotions affect reasoning. These questions, and the computational systems that have been implemented to explore them, fall in the domain of *belief systems*.

Abelson (1979) has outlined a number of peculiarities that set beliefs apart from facts and that distinguish belief systems from other systems in AI:

1. Belief systems are not *consensual*. Different beliefs may result in different interpretations of the same phenomena. For example, depending on one's beliefs, the "generation gap" results from insensitive and restrictive parents or from ungrateful and immoral children. One's beliefs can influence interpretation of relatively sure facts; for example, some smokers refuse to believe that smoking causes cancer, and some people insist that concentration camps never existed but are the creation of propagandists.

2. Beliefs deal with *conceptual* entities such as the generation gap, the supernatural, and extrasensory perception. Thus, an entity that exists in one belief system may be absent in another.

3. Sometimes belief systems represent alternative "worlds," typically, "the world as it should be." Ideologies often have implicit alternative worlds.

4. Beliefs have an *evaluative* or *affective* component. Events tend to be good or bad, to evoke pleasure or displeasure. Abelson distinguishes between two aspects of affect. One involves the world divided up into good and

bad things (or into as many categories as there are affects). From this categorization one can infer the goodness or badness of events or objects. For example, if X is bad, and Y helps X, then Y must be bad also. Much of Abelson's early research was devoted to this kind of reasoning. A second aspect of affect is how it influences the operation of a system; for example, Faught (1975) characterizes emotions as leading to motives, and Bower (1981) discusses the effects of emotion on memory.

5. Beliefs may be based on subjective experiences or episodes. Logical, rational deductions may be based on a subjective event. For example, an elaborate theory may be constructed around an event that was believed to occur but that actually did not. An interesting historical example is the mass hallucination of French physicists in the "N-Ray Affair" (Klotz, 1980). It was believed that N-rays could be detected by their effects on the brightness of an electric light-bulb, and for many years, French physicists published reports of the curious properties of N-rays. This research continued (though at a lesser pace) even after it was demonstrated that perceived fluctuations in brightness were entirely illusory. N-rays do not exist and the physics that had been developed to explain them was founded on a hallucination.

6. One does not know, a priori, what knowledge is relevant to a belief. The knowledge pertinent to diagnosis of glaucoma, for example, can be circumscribed relatively easily. It is less easy to decide what is irrelevant to conceptual entities such as *the sexual promiscuity of today's youth*.

7. Credibility and emotion interact in evaluation. One may believe something is true, passionately; or there may be no emotional investment in a belief. For example, it may be true that one brand of pain reliever contains more aspirin than another, but it is hard to achieve the enthusiasm necessary to value one more highly.

These characteristics of belief and belief systems make reasoning from belief more complicated than reasoning from facts or measurable uncertainties. This is for several reasons, all related to what the belief system knows. First, the nonconsensuality argument is that different belief systems house different bodies of knowledge; thus, it may be difficult for one system to explain or predict the behavior of another. For example, it is a difficult task for the BUGGY system (see Article IX.C7, in Vol. II) to derive the inference rules applied by its students in working arithmetic problems. The students make assumptions about arithmetic that are not consensual with the assumptions of the adult community; consequently, they make errors. BUGGY's task is to explain the errors by inferring the students' mistaken assumptions. Another example from the ICAI literature (see Chap. IX, in Vol. II) illustrates the power of assuming consensuality: Several ICAI systems maintain a *student model*—a representation of what the student knows—to facilitate teaching.

Just as nonconsensuality is a problem, so are *existence* and *openness*, and for much the same reason. The existence problem is that reasoning in

one system may be predicated on premises that do not exist in another: for example, one can do little to mollify a person who believes that his (or her) bad fortune is preordained. The belief in preordination is so central to his belief system (though alien to one's own) that he accepts misfortune with resignation and will do nothing to improve his lot. The openness problem is concerned with the relevance of the knowledge used for reasoning; in one system a fact may be central to an argument, while in another it is tangential. For example, one person may attribute the decline of our society to the availability of drugs, while another may believe the cause is inflation and a third may insist that impiety is responsible. The first person constructs the causal argument that society is being destroyed by drugs. He holds this argument with a conviction that is lacking in the second person, who views drugs as a symptom of an inflated economy, not as a symptom of impiety or as a cause of society's ills.

Two other aspects of belief make reasoning difficult. One is the role of affect, or emotion, and the other is the role of confidence, or certitude. It is tempting to make a dichotomy between *rational* and *irrational* thought and to assign emotion to the latter category and ignore it. But there is strong evidence that emotion has powerful effects on human cognition. In a recent and extensive series of experiments, Bower (1981) and his colleagues have shown that emotion influences what we learn, what we remember, and how we make a variety of judgments. Our evaluations of ourselves and others and of events are subtly but strongly biased by what we are feeling. Bower's results suggest that emotion cannot be ignored as a factor in human cognition and that it is at least one factor that argues against a strong rational-irrational dichotomy.

The problem of confidence, or certitude, is that much of the information used in reasoning is not true or false, but somewhere in between, and that one's confidence in the information affects one's reasoning. One attempt to capture this aspect of reasoning is found in MYCIN (see Article VIII.B1, in Vol. II), which attaches *certainty* (*or confidence*) *factors* (CFs) to its conclusions. The initial CFs are supplied to the MYCIN system with its heuristic rules by expert diagnosticians. Then, as MYCIN reasons, it combines the CFs associated with the rules to produce a CF for its conclusion. The CF mechanism is quite crude, however, and very ad hoc. Clearly, MYCIN does not embody a theory of human reasoning under uncertainty. More successful are Tversky and Kahneman (1974), who have identified a number of factors that influence judgments under uncertainty.

Even though reasoning with beliefs involves certain sophistications over reasoning with facts, the two have been modeled in much the same way. Belief systems are formally similar to some of the knowledge-based systems in the *Handbook*. For example, the belief that *If A likes B, then A will help H* can be phrased as a production from which the conclusion *A will help H* follows logically from the premise *A likes B*. This deduction is logically and psychologically valid. Other conclusions may maintain a formal logical

validity but be psychologically odd: for example. *If you are suffering, then you have found true happiness.* It is useful to distinguish the formal logical structure of a belief system from the psychological conclusions that arise from it. The remainder of this article is concerned with both of these factors—with formal representations that facilitate psychological, not necessarily logical, behavior.

Implicational Molecules

Abelson and Reich (1969) described a system based on *implicational molecules*, that is, sets of clauses related by psychological implication. For example:

```
[A does X, X causes Y, A wants Y]
```

or

```
[A likes B, A helps B]
```

Just as a premise implies a conclusion, so does one part of an implicational molecule imply another. Thus, implicational molecules can predict or explain events:

> If *A wants Y*, it is plausible to predict *A does X*. *A does X* because *X causes Y and A wants Y*.

Abelson and Reich used implicational molecules in a system that simulated the extreme right-wing viewpoint of a cold-war ideologue. The system used stereotyped concepts such as *Western-governments*, *situations-helpful-to-the-Communists*, and *prevent*, *promote*, and *control*. These were combined to form *generic sentences* such as *Liberals control Western-governments*. Generic sentences were then combined into implicational molecules that define the conclusions that are reasonable in the system:

```
[Western-governments promote
    situations-helpful-to-the-Communists,
Standing-up-to-Communists prevents
    situations-helpful-to-the-Communists,
Liberals control Western-governments,
Liberals fear standing-up-to-Communists] .
```

A higher order structure was the *master script*, which spelled out several general contingencies for the fate of the free world. Part of the script says that the Communists want to dominate the world and will do so unless the free world exercises its power, in which case the free world will surely prevail. Generic events were considered instances of very general master-script events.

The system could judge the credibility of events; bad events were attributed to the Communists, good to the free world, and never the other way around. It could also predict events and say what should be done if and

when they happened. This was accomplished by associating an event with one on the master script and following it to a conclusion. For example, an event interpreted as Communist domination was predicted to result in world takeover unless the free world flexed its muscles.

The system answered specific questions about real people, not just abstract questions about generic sentences. It did so by instantiating generic sentences with more concrete concepts. For example, *Liberals control Western-governments* might be instantiated in the belief that *LBJ controls the United States.*

One characteristic of belief systems in general is that they perform well with stereotyped beliefs. They reflect what we suspect to be true—that little knowledge is required to hold an oversimplified, dogmatic opinion. (Why let facts interfere with what one knows is right?) Abelson's Cold War Ideologue was not very knowledgeable; it could easily conclude that the Berlin Wall was built by the Red Chinese, since it is just the sort of miserable thing that Communists do. Ideological oversimplification seems to provide a counter-example to the pervasive idea that knowledge is power. To achieve strong dogma, one must ignore the evidence, counterexamples, and qualifications that compromise a position.

The Structure of Belief Systems

Abelson (1973) later developed a hierarchical formalism for beliefs, based on *conceptual dependency analysis* (see Articles III.C8 and IV.F5, in Vol. I). Abelson starts his analysis with three kinds of *atoms: purposes, actions,* and *states.* Purposes encode the wants or desires of actors; for example, *Mary wants John to do his share of housework.* Actions are the things that the actors want to do, and states are the situations that they want to bring about. The next level of Abelson's hierarchy combines these atoms into *molecules;* these are similar to the implicational molecules described earlier.

Molecules represent actions undertaken by actors to produce outcome states. In their simplest form they are (*Purpose, Action, State*) triples, but larger chains and networks are also possible. Among the larger structures are *plans, themes,* and *scripts.* Plans represent action-state sequences, where each state *enables* a subsequent action until a final goal state is obtained. The structure of plans reflects that a set of sequential or parallel actions is usually required to achieve a goal. By assumption, plans are always related directly to the purposes of a main actor. If other actors are involved, they are simple agents or instruments with no autonomy; they cannot enhance or frustrate the plans of the main actor.

While plans represent the purposes of a single actor, interactions of the purposes and plans of autonomous actors are represented in *themes.* Abelson formed a taxonomy of themes based on the possible interactions of two actors (see Table F-1).

TABLE F-1
A Taxonomy of Themes (from Abelson, 1973)

Sentiments toward Other	Influence of Actors: Neither influences Other	One influences Other	Both influence Other
Some positive, no negative	Admiration	Devotion Appreciation	Cooperation Love
One actor negative	Alienation (also Freedom)	Betrayal Victory Dominance	Rebellion
Both actors negative	Mutual Antagonism	Oppression (also Law and Order)	Conflict

Mutual antagonism, for example, refers to agents who are negative to each other, but powerless to inflict harm. When one actor is able to harm the other, oppression results; when each can influence the other, conflict results.

Scripts are sequences of themes that follow each other in some psychologically plausible fashion. (The reader should be aware that this is an earlier and different interpretation of the roles of themes and scripts than is found in Abelson's research with Schank, 1977; see Schank and Abelson, 1977, and Article IV.F8, in Vol. I.) Simple scripts involving two actors are, for example, *blossoming relationships*, wherein a Love theme develops from the themes of Admiration, Cooperation, Devotion, or Appreciation, and *souring relationships*, which happen when Love is complicated by Rebellion and, subsequently, Mutual Conflict.

Differences between individual belief systems are manifest primarily at the theme and script levels. These constructs provide for alternative views of the same events; for example, a relationship might be viewed as alienation by one actor and as mutual antagonism by the other. One may feel he is not at fault for a deteriorating relationship; the other may feel that hostility is involved. The greatest idiosyncrasy of belief is found at the script level, where the repertoire of scripts maintained by an individual defines his ideology (recall the *master-script* that defined the beliefs of the cold-war ideologue).

We now turn from Abelson's designs for general belief systems to a specific kind of belief, namely, paranoid belief.

PARRY

PARRY was one of the earliest and most ambitious simulations of the role of beliefs and affects in cognition. It is a model of what its designers call

the *paranoid mode*, a pattern of behavior motivated by paranoid beliefs and intentions. PARRY's original designer, Kenneth Colby, is a psychiatrist, and PARRY embodies his theory of paranoid behavior. We will discuss this theory shortly, but first we consider the characteristics of paranoia.

Paranoids are suspicious; they think that other people intend to harm them. They believe they are the target of conspiracies. They have a great concern with "evidence," and are likely to treat a random event as significant and intentional (the intentions are held by "them"—those malevolent others). Paranoids are also hypersensitive to criticism:

> References to the self are misconstrued as slurs, snubs, slights, or unfair judgements. He may feel he is being watched or stared at. He is excessively concerned about his visibility to eyes that threaten to see concealed inadequacies, expose and censure them. Cameras, telescopes, etc. that may be directed his way unnerve him. He may feel mysteriously influenced through electricity, radio waves, or (more contemporaneously) by emanations from computers. He is hypersensitive to criticism. In crowds he believes he is intentionally bumped. Driving on the highway he feels repeatedly followed too closely by the car behind. Badgered and bombarded without relief by this stream of wrongs, he becomes hyperirritable, querulous, and quarrelsome. (Colby, 1975, p. 4).

Two other characteristics of paranoia are fearfulness and hostility. One can see how both might arise from the conviction that the self is in a hostile and intentionally malevolent world. A last characteristic, which Colby says makes paranoia very difficult to treat, is rigidity and absolute conviction. Once a paranoid is convinced, for example, that his doctor is in collaboration with "them," it becomes extremely difficult to reestablish rapport because the patient will not compromise his beliefs.

The characteristics of paranoia are so clear-cut that it is possible to simulate the paranoid mode. PARRY was and is an ambitious project because it involves integrating beliefs, intentions, and affects with more "rational" cognition. The manner in which these components interact is dictated by Colby's theory of paranoia.

Paranoid behavior arises, according to Colby, from attempts to avoid humiliation. In the PARRY simulation, humiliation arises, and is intently avoided, during an interview with a doctor. (PARRY has a natural-language front-end, but it is not very sophisticated and we will not be concerned with it here.) Briefly, the paranoid (and PARRY) is hypersensitive to any comment that can be interpreted as reflecting his own inadequacy. Any such comment increases shame and humiliation. (Intense paranoia involves interpreting virtually *all* interactions in this way.) The paranoid seeks to avoid humiliation and shame, since it is intensely painful, so whenever he detects a situation in which the doctor might be making a humiliating comment, he takes three defensive actions: One is to change his opinion of the doctor (e.g., *Anyone who thinks I'm crazy must be really incompetent*); another is to

decrease his level of shame, since he has concluded that the doctor, and not he himself, is at fault; and the third is to take some action, which may be hostile.

To achieve this behavior, PARRY has a number of beliefs, a number of common inferences, and several processes that we will describe briefly. Beliefs include *The doctor is crazy* or *The doctor is friendly*. PARRY also has four beliefs that reflect humiliation: *PARRY is stupid, PARRY is dishonest, PARRY is crazy*, and *PARRY is worthless*. PARRY must avoid concluding that any of these are true, since these conclusions cause pain. Unfortunately, PARRY is always trying to find evidence for them in its interactions with the doctor. This is the problem: To avoid humiliation, the paranoid must constantly search for it; he must catch the insult and deflect it before it harms him.

PARRY has a set of inferences that alert it to insults, and its hypersensitivity arises from these inferences. For example, if the doctor says, "You didn't answer my question," PARRY infers that the doctor thinks he is stupid; this statement can also be taken as evidence that the doctor thinks PARRY is not telling the truth—is dishonest. Whenever the doctor says anything, PARRY makes whatever inferences it can, and if the inferences support any of the four *humiliation* beliefs that we just mentioned, then PARRY increases its level of *shame*.

Thus, one of PARRY's processes is to search for evidence of humiliation in the doctor's communications. When this process finds evidence, another *affect* process increases PARRY's shame; if the level of shame crosses a threshold, PARRY launches into characteristic hostile paranoid behavior. This involves a third process dealing with intentions. PARRY has three emotions—fear, anger, and shame—each of which plays a role in PARRY's intentions. When anger is high, PARRY intends to attack the doctor; when fear is high, PARRY intends to alter the interview situation so that the outcome it fears—humiliation—is less likely. And when shame is high, PARRY does three things: It defends itself by throwing out the belief that led to humiliation and replacing it with another one, usually a revised and uncomplimentary belief about the doctor (e.g., *The doctor is crazy*); it reduces its shame, since the belief that caused shame has been exorcised; and it intends a strong action, usually based on its new belief about the doctor. It may intend to attack, lie, or withdraw.

We have discussed how PARRY integrates inferences with affects and intentions to produce paranoid behavior; we conclude with the following brief example of a dialogue with PARRY. Initially, the doctor asks a lot of innocuous questions, from which PARRY concludes that the doctor is friendly, wants to help, and can help. For example:

Doctor: `What do you do for a living?`

PARRY: `I work at Sears.`

But the doctor is sure to stumble onto a topic that PARRY finds sensitive, at which point PARRY will experience an increase in shame:

Doctor: How much do you get paid an hour?
PARRY: My salary is none of your business.

In this interaction, PARRY concluded that the doctor was implying that PARRY had no money—was worthless. Although shame was not elevated to the point that PARRY enters paranoid mode, it does provoke an angry attack. PARRY can be mollified if the doctor goes back to asking innocuous questions, or compliments PARRY, or apologizes.

At a later point in the interview, PARRY introduces its main concern—that bookies and the Mafia are out to get him. At this point, the doctor must tread very carefully. If the doctor expresses any skepticism about PARRY's delusions, shame will increase markedly, and PARRY will become uncooperative. As long as the doctor asks simple questions, PARRY will answer them:

Doctor: What about bookies?
PARRY: A bookie didn't pay me off once.
Doctor: And then what happened?
PARRY: I had an argument with the bookie who did not pay off. Later on I remembered he had friends in the underworld and I thought he would use them to get even with me.

Incidentally, PARRY has an elaborate story about the bookies, rather like Abelson's master-script. One of PARRY's goals for the interview is to tell the whole story. The last interchange was, therefore, quite successful. However, telling the whole story makes PARRY vulnerable and sensitive, so that when the doctor says:

Doctor: I find you interesting.

PARRY thinks it is an insult and responds angrily:

PARRY: Are you trying to say something?

The doctor then asks several questions that are inflammatory and that elevate PARRY's affects. The doctor finally says:

Doctor: You appear to be very sick.

This is intolerable to PARRY. It increases his fear, anger, and shame markedly. From this point on, the interview deteriorates. PARRY lies:

Doctor: How do you feel right now?
PARRY: I am perfectly fine.

And withdraws:

Doctor: Do you need to have psychiatric treatment?
PARRY: I don't want to talk about it.

PARRY substitutes uncomplimentary beliefs about the doctor for shameful beliefs about himself, concluding that all doctors are bad and that this one is abnormal. These beliefs allow PARRY to reduce shame and drop out of the paranoid mode into being merely angry, so that when the doctor concludes the interview and thanks PARRY for his cooperation, PARRY bluntly tells him not to come back.

Conclusion

The study of belief systems is challenging because, unlike "facts," beliefs are nonconsensual, have associated affects, and have associated confidences or credibilities. Even the basic problem of how confidences in beliefs are adjusted by evidence has no general solution, and the more difficult problems (e.g., the effects of emotion on cognition) are barely formulated, much less solved. Despite these difficulties, the researchers surveyed here are convinced of the importance of belief systems, since humans clearly do not reason entirely from facts with consistent inference rules, but instead, prejudices, biases, episodic memory, confidences, and emotional states are neatly integrated into "rational" reasoning.

References

Abelson (1973) provides a readable account of his research prior to his collaboration with Schank. Schank and Abelson (1977) is a very readable account of their collaborative work, although it is more concerned with natural-language understanding than with belief systems. For those who are interested in emotion, Mandler (1975) presents a complete cognitive theory, with historical information, and Bower (1981) is a survey of some surprisingly powerful effects of mood on memory and cognition.

Colby (1975) has written a short monograph that details the PARRY program; Faught, Colby, and Parkinson (1974) provide another good discussion, though it lacks the psychiatric background.

BIBLIOGRAPHY

Abelson, R. P. 1973. The structure of belief systems. In R. C. Schank and K. M. Colby (Eds.), *Computer models of thought and language*. San Francisco: Freeman.

Abelson, R. P. 1979. Differences between belief and knowledge systems. *Cognitive Science* 3:355–366.

Abelson, R. P., and Reich, C. M. 1969. Implicational molecules: A method for extracting meaning from input sentences. *IJCAI 1*, 647–748.

Amarel, S. 1968. On representations of problems of reasoning about actions. In D. Michie (Ed.), *Machine intelligence 3*. New York: American Elsevier, 131–171.

Anderson, J. R. 1976. *Language, memory, and thought*. Hillsdale, N.J.: Lawrence Erlbaum.

Anderson, J. R. 1980. On the merits of ACT and information-processing psychology: A response to Wexler's review. *Cognition* 8:73–88.

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

Baddeley, A. D. 1976. *The psychology of memory*. New York: Basic Books.

Bobrow, D. G., and Collins, A. 1975. *Representation and understanding*. New York: Academic Press.

Bower, G. H. 1981. Mood and memory. *American Psychologist* 36:129–148.

Brachman, R. J. 1978. On the epistemological status of semantic networks. BBN Rep. No. 3807, Bolt Beranek and Newman, Inc., Cambridge, Mass.

Cherry, C. 1970. *On human communication*. Cambridge, Mass.: MIT Press.

Colby, K. M. 1975. *Artificial paranoia*. New York: Pergamon.

Collins, A. M., and Quillian, M. R. 1969. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8:240–247.

Collins, A. M., and Quillian, M. R. 1972. How to make a language user. In E. Tulving and W. Donaldson (Eds.), *Organization and memory*. New York: Academic Press.

Conrad, C. 1972. Cognitive economy in semantic memory. *Journal of Experimental Psychology* 92:148–154.

Crowder, R. G. 1976. *Principles of learning and memory*. Hillsdale, N.J.: Lawrence Erlbaum.

Faught, W. S. 1975. Affect as motivation for cognitive and conative processes. *IJCAI 4*, 868–868.

Faught, W. S., Colby, K. M., and Parkison, R. 1974. The interaction of inferences, affects, and intentions, in a model of paranoia. Memo AIM 253, AI Laboratory, Stanford University.

Feigenbaum, E. A. 1963. The simulation of verbal learning behaviour. In E. A. Feigenbaum and J. Feldman (Eds.), *Computers and thought*. New York: McGraw-Hill, 297–309.

Feigenbaum, E. A., and Simon H. A. 1962. A theory of the serial position effect. *British Journal of Psychology* 53:307–320.

Hayes-Roth, B. 1980. Human planning processes. Rep. No. R-2670-ONR, Rand Corp., Santa Monica, Calif.

Hayes-Roth, B., and Hayes-Roth, F. 1978. Cognitive processes in planning. Rep. No. R-2366-ONR, Rand Corp., Santa Monica, Calif.

Hintsman, D. L. 1968. Explorations with a discrimination net model for paired-associate learning. *Journal of Mathematical Psychology* 5:123–162.

Klots, I. M. 1980. The N-ray affair. *Scientific American* 242:168–176.

Mandler, G. 1975. *Mind and emotion*. New York: Wiley.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some limits of our capacity for processing information. *Psychological Review* 63:81–97.

Miller, L. 1978. Has artificial intelligence contributed to an understanding of the human mind? A critique of arguments for and against. *Cognitive Science* 2:111–127.

Minsky, M. L. (Ed.). 1968. *Semantic information processing*. Cambridge, Mass.: MIT Press.

Neisser, U. 1976. *Cognition and reality*. San Francisco: Freeman.

Newell, A. 1970. Remarks on the relationship between artificial intelligence and cognitive psychology. In R. B. Banerji and M. D. Mesarovic (Eds.), *Theoretical approaches to non-numerical problem solving*. Berlin: Springer-Verlag.

Newell, A., and Simon, H. A. 1956. The logic theory machine. *IRE Transactions on Information Theory* 2:61–79.

Newell, A., and Simon, H. A. 1963. Computers in psychology. In R. D. Luce, R. R. Bush, and E. Galanter (Eds.), *Handbook of mathematical psychology* (Vol. 1). New York: Wiley, 361–428.

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

Nii, H. P., and Feigenbaum, E. A. 1978. Rule-based understanding of signals. In D. A. Waterman and F. Hayes-Roth (Eds.), *Pattern-directed inference systems*. New York: Academic Press, 483–501.

Norman, D. A., Rumelhart, D. E., and the LNR Research Group. 1975. *Explorations in cognition*. San Francisco: Freeman.

Quillian, M. R. 1968. Semantic memory. In M. Minsky (Ed.), *Semantic information processing*. Cambridge, Mass.: MIT Press, 216–270.

Rumelhart, D. E., Lindsay, P. H., and Norman, D. A. 1972. A process model for long-term memory. In E. Tulving and W. Donaldson (Eds.), *Organization and memory*. New York: Academic Press, 198–246.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, plans, goals, and understanding*. Hillsdale, N.J.: Lawrence Erlbaum.

Shannon, C. E., and Weaver, W. 1963. *The mathematical theory of communication*. Urbana: University of Illinois Press.

Simon, H. A. 1969. *Sciences of the artificial*. Cambridge, Mass.: MIT Press.

Simon, H. A. 1979. Artificial intelligence research strategies in the light of AI models of scientific discovery. *IJCAI* 6, 1008–1014.

Simon, H. A., and Feigenbaum, E. A. 1964. An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning. *Journal of Verbal Learning and Verbal Behavior* 3:385–396.

Skinner, B. F. 1976. *About behaviorism.* New York: Vintage Books.

Sternberg, S. 1969. Memory-scanning: Mental processes revealed by reaction time experiments. *American Scientist* 57:421–457.

Tversky, A., and Kahneman, D. 1974. Judgement under uncertainty: Heuristics and biases. *Science* 185:1124–1131.

Wexler, K. 1978. A review of John R. Anderson's language, memory, and thought. *Cognition* 6:327–351

NAME INDEX

Pages on which an author's work is discussed are italicised.

SUBJECT INDEX