part 2

Simulation of Cognitive Processes

The research reported in Part 2 is concerned with the construction of computer models of the information processes underlying human thought. While the major aim of the research reported in Part 1 is the programming of computers to perform intellectual tasks, the work reported here is concerned with the programming of computers to perform intellectual tasks in the same way that persons perform these tasks. An example may be helpful in clarifying the distinction between artificial intelligence research and simulation of cognitive processes research. An artificial intelligence researcher interested in programming a computer to play chess would be happy only if his program played good chess, preferably better chess than the best human player. However, the researcher interested in simulating the chess-playing behavior of a given individual would be unhappy if his program played chess better (or worse) than that individual, for this researcher wants his program to make the same moves as the human player, regardless of whether these moves are good, bad, or indifferent.

Why Program Computers To Do Tasks the Way People Do Them?

Researchers program computers to behave like people to further their understanding of, *i.e.*, their ability to predict, certain phenomena of human behavior. The computer program is a model which represents the researcher's hypotheses about the information proc-

esses underlying the behavior. The program is run on a computer to generate the predictions of the model. These predictions are compared with actual human behavior. There are usually some discrepancies between prediction and behavior, and the model is revised to reduce these discrepancies. Then the entire process is repeated. Eventually the researcher hopes to obtain a model which will be a good predictor of the relevant behavior. As he continues to test his model and to improve it, the researcher gains confidence in the belief that this model represents the processes underlying the behavioral phenomena he is studying.

This overview of the procedure of researchers using computer models to study human thought processes indicates that these workers use the same procedure that all scientists use. The only difference involves the representation of the model as a computer program and the use of the computer to determine the predictions of the model.

What Are the Advantages of Representing Models of Human Thought as Computer Programs?

The researcher may represent a model of human behavior in any of a number of different ways. Perhaps the most common representation is natural language, *i.e.*, the language in which we usually communicate, be it English, French, German, Russian, or Sanskrit. If we were to specify a model of a chess player in natural language, the description of the model might begin as follows:

> Before making a move, the player checks to see if his king or queen is in danger. . . .

A model of a chess player might also be represented in the language of mathematics.

A third possibility is the representation of the model as a program for a digital computer.

The selection of a medium for model construction depends on the characteristics of the medium, the characteristics of the model, and the resources of the model builder. Natural language is an extremely flexible medium, and perhaps any conceivable model can be represented in natural language. Moreover models represented in natural language are generally easy to communicate to others. However, these statements about ease of representation and communication must be qualified for extremely complex models. Perhaps the most serious failing of natural language as a medium for model construction is the difficulty of rigorously determining the predictions of a model expressed in this medium. This is true because, in expressing

ourselves in natural language, we use ambiguous words; *e.g.*, in the example of a statement in a natural language description of a model used on page 270, what does "checks" mean? What does "danger" mean?

Many of the models expressed in conventional mathematics can be analyzed in a rigorous fashion. If the mathematics is known to the model builder or can be discovered by him, he will be able to determine the implications of his model. If the mathematical techniques for solving certain equation systems are not known or available to the model builder, he is in no better a position than if he had only a natural language model. The effect of this last condition is to constrain the model builder to consider only that class of models for which he knows solutions are available. Unfortunately this constraint may have a spurious effect on the model builder; *e.g.*, he may oversimplify a complex situation. In general, many of the mathematical models of human behavior are elegant and simple. Sometimes, the constraints of the mathematical medium force unfortunate compromises upon the model and reduce its ability to predict.

The most recently developed medium for model construction is the digital computer program. The computer program has several important advantages as a medium for model construction. One of these is the wide range of models that can be represented as computer programs. Effective constraints on the size and complexity of the models which can be represented are so few and slight that they can be disregarded in most cases. The only real constraint on the model builder is that his statements be unambiguous and complete. For example, before the statement about the chess model given on page 270 could be programmed, the meaning of the words "checks" and "danger" would have to be specified. This constraint should be considered a blessing rather than a limitation, for it forces a refreshing rigor on builders of models of human thought processes. Despite the freedom given to the model builder in constructing computer models, he retains the ability to make a rigorous determination of the implications of the model. For the computer can execute the program and determine the behavior of the program in particular situations.

The construction of computer models has been facilitated by the development of computer languages that relieve the programmer from the burdens of the microprogramming associated with machine languages. The development of powerful algebraic languages like FORTRAN and the various dialects of ALGOL are one class of these higher-order computer languages. A second class of languages are the list processing languages—IPL, FLPL, LISP, and COMIT—which have been developed for work in artificial intelligence and

simulation of cognitive processes. These list processing languages provide dynamic storage allocation and many other features which simplify the programming task.

What Is the Information Processing Level of Explanation?

In constructing the models described in the reports in Part 2, the researchers had to make several choices. One is the behavior which they want to predict. All the reports are focused on human thought processes. A second choice is the medium for model construction. All the models described in this part are represented as computer programs. A third choice that must be made is the level of explanation of the model. In all the reports, the researchers chose an information processing level of explanation.

A particular phenomenon may be explained at any one of several different levels. In the present context, this statement means that human thought processes can be explained in terms of electrical and chemical processes which take place in the brain, in terms of the organization of the neurons of the brain, or in terms of information processes. At first glance, explanations at each of these levels may appear to be quite different, for we have not yet been able to develop an integrated theory of thinking which will simultaneously account for all that takes place in the way of electrical, chemical, organizational, and informational processes. Hopefully one day all these levels of explanation will be integrated, and the relationships between them will be established. Until that day, perhaps the wisest course is for investigators to continue to work in parallel, taking time out occasionally to compare notes, for there is not one best level of explanation. One can evaluate the models proposed at any level only by their ability to predict behavior. Models at any of these levels of explanation of human thought can be formulated as computer programs, and computers have been used to study neurons, neural organization, and information processes (see the Bibliography).

The reports in Part 2 explain human thinking processes at an information processing or symbol manipulation level. The basic premise of this approach is that complex thinking processes are built up of elementary symbol manipulation processes. A fundamental set of these elementary processes might be the following: read a symbol, write a symbol, copy a symbol, erase a symbol, and compare two symbols. Another basic facility that is required is the ability to take different courses of action depending on the outcome of the compare operation. If two symbols are compared and found to be identical, the information processing system will take one course of action, *i.e.*,

execute one set of elementary processes. If the symbols are different, the system will execute another set of processes.

The goal of the researcher is to find an ordered sequence of these basic processes which when provided with suitable information will produce behavior indistinguishable from the behavior produced by human beings when they are provided with comparable information. The "ordered sequence of these basic processes" is the "model" that we have referred to previously. Thus an information processing model of a chess player would consist of a sequence of these basic processes. We could provide this model with suitable initial information, *e.g.,* a chess problem, and with the aid of a computer determine the predictions of the model.

How are Information Processing Models of Human Thought Determined?

While there is no prescribed procedure for creating information processing models of thought, nevertheless the researchers whose work is reported in Part 2 all had to do certain things in the process of obtaining their models. We now list the major steps involved in the creation of such a model.

The researcher begins with an interest in a certain area of human behavior, *e.g.,* problem-solving, game playing, learning. He then usually focuses on behavior in a specific task, *e.g.,* solving logic problems, playing chess, learning nonsense syllables. The particular task may be selected because the researcher is interested in the task for its own sake or because he has some idea about how people behave in that particular task. In order to construct a preliminary model of behavior in the task, the researcher needs an idea or ideas about behavior in the task. The idea may come from observing behavior, from asking people what they are doing while performing the task, or from cogitating on what type of device would be required to perform the task. The next step is to construct a model, *i.e.,* write a program for a computer, embodying this idea. In the course of this activity, the researcher realizes the inadequacy of his original idea(s). He discovers that he needs more information. To get this information he may have to reanalyze old data or run new experiments. Thus the researcher encounters one of the advantages of computer models —the requirement for completeness and precision. After many changes and revisions of the initial program, the model is finally completed.

The researcher then provides the model with the same task given to human subjects, *i.e.,* a chess problem, a logic problem, a list of

nonsense syllables. The program is then run with this information. In effect, the model may be considered as an artificial subject participating in a replication of the experiment that was performed with human beings. The behavior of the program, *e.g.*, the chess moves, the steps in the solution of the logic problem, the responses in the learning task, is then compared with the behavior of the subject. Where possible, the processes which led to the overt behavior are also compared.

How Are Computer Models of Human Thought Processes Tested?

The actual comparison or test procedure depends on whether the model represents a particular individual or a generalized or idealized individual (see the article by Feigenbaum, pp. 299–300). The verbal learning model of Feigenbaum, the concept learning model of Hunt and Hovland, and the social behavior model of the Gullahorns are models of generalized or idealized individuals. The relevant comparisons here, and the comparisons that these authors make, are between the behavior of their models and the behavior of typical individuals.

The models of Newell and Simon, Clarkson, and Feldman are models of specific individuals. The relevant comparisons here are between the detailed choices of the individual subject and the detailed choices of the model. These comparisons involve many problems. None of these problems is new or unique to computer models, but they are highlighted by the types of comparisons made with these models of individual behavior. One of these problems is parameter estimation. Another problem involves the relationship between successive predictions of the model. The prediction of the model at time t depends on the behavior of the model at $t - 1$, $t - 2$, $t - 3$, and/or other decisions of the model. If the model makes a decision different from that of the subject at one point in time, the discrepancy may lead to further discrepancies or spurious agreements. Thus, if the model of a chess player makes a first move different from that of the player, we would hardly expect the second moves to agree. To eliminate this difficulty, some investigators (see the article by Feldman) have selected the strategy of "setting the model back on the track" after each decision. In the chess example, this would mean that after the model makes its first move, the move is compared with the first move of the human player. If the moves are the same, the opponent's move is made; and the model proceeds to its second move. If the first move of the model and the first move of the player differ, the discrepancy is noted; and the model's move is replaced by the

subject's move before the play continues. This setting-back-on-the-track strategy attempts to eliminate the dependencies involved and makes each decision of the model as independent as possible of the previous decisions of the model. Such a procedure allows a better evaluation of the performance of the model than a comparison procedure which permits errors to beget errors (or spurious correct decisions). Unfortunately it is not always possible to implement a set-back-on-the-track procedure, either because the intermediate information is not available (as it is in chess or the binary choice experiment) or because it is difficult to set the model back on the track, as, for example, in GPS.

What Are Some of the Unsolved Problems of Simulation of Cognitive Processes Research?

The problems of simulation of cognitive processes research are of two types. The substantive problems are those areas of human cognition about which we know very little. The problems listed above of learning heuristics, of inductive inference, and of using natural language are three good examples of areas in which our knowledge of human thought processes is pitifully small. These are areas which will attract much interest in the near future.

The procedural or methodological problems of the simulation of cognitive processes technique represent a second class of problems. Three of these problems are readily apparent:

1. Testing models and estimating parameters. Models are seldom entirely wrong, and researchers seldom reject models completely. The more common procedure is to try to identify the sources of error in the model and correct the model (Grant, 1962). Since computer models are so large and complex, techniques for identifying sources of error are extremely important. The efficient utilization of the available information on human thought processes is very important.

2. Experimentation. Much of the simulation of cognitive processes work has been identified with the use of a protocol obtained by asking the subject to think aloud. Work needs to be done on this and other ways for obtaining information about thought processes (Blum, 1961). Some work has been done and more needs to be done on the determination of the effects of having the subject think aloud (Colby, 1960; Gagne, 1962). Ingenious procedures must be developed to make explicit the thought processes of the subject. One such procedure is the use of artificial languages for problem statements (Wickelgren and Cohen, 1962). Much additional work is needed in all these areas.

3. Program organization and representation. We have repeatedly stated that the expression of models as computer programs allows considerable flexibility in statement. This assertion is true. However, as models and programs increase in size and complexity, the organization of these programs creates serious problems. Higher-order programming languages and list-processing languages in particular solve some of these problems, but many problems remain (Newell, 1962).

How Will the Computer Affect the Study of Cognitive Processes?

The effects of the modern digital computer on the study of human thought processes will be twofold. On the one hand the researchers in artificial intelligence want to understand how human beings perform all sorts of intelligent tasks. Thus computers will be contributing to the demand for more knowledge of human thought processes. On the other hand, the computer has become the basis of a powerful method for studying human thought processes, and so the computer will contribute to the supply of understanding of human cognitive processes. The supply of knowledge about human thought will never catch up with the demand. But the forecast for progress in research in human cognitive processes is most encouraging.

section 1

Problem-solving

The study of human problem-solving behavior has had a fascination for many persons since the time of the Greeks. With the advent of the modern digital computer and the ability to represent complex problem-solving models as computer programs and to study these models by executing the programs, the study of problem-solving behavior has received new impetus.

The authors of the following report and their colleague, J. C. Shaw, are three of the pioneers in the use of computer models of human problem-solving behavior. The psychological significance of their work on the General Problem Solving program (GPS) derives from their success in creating a model whose behavior in solving logic problems is strikingly similar to human behavior on these same problems. In the report reprinted here, the behavior of a variant of GPS on a problem is compared with the behavior of a human subject on the same problem. In other reports, the behavior of other variants of GPS has been compared with the behavior of other subjects, with equally good results.

While each person behaves somewhat differently, practically all the individuals that have been tested by Newell and Simon and their associates in the logic task display a common set of basic processes involving the use of means-ends type of analysis. Although other investigators (notably Duncker, 1945) have reported the use of means-ends analysis by subjects solving problems, Newell, Shaw, and Simon have rigorously specified means-ends analysis and implemented this problem-solving scheme on a computer. With their computer model,

they have been able to perform a wide variety of experimentation with the scheme.

In addition to the means-ends technique, some subjects make use of another powerful method, which has been called "planning." A subject using a planning method abstracts or simplifies a complex problem. He then solves the simpler problem and uses the information obtained in the solution of the simpler problem in the solution of the original complex problem. This technique has also been included in GPS and is more fully described in another report (Newell, Shaw, and Simon, 1959*a*).

GPS is more than a model of human behavior in logic problems. A deliberate effort has been made in GPS to partition the "general" part of the program—mostly concerned with mean-ends analysis—from the problem-specific part of the program—generally referred to as the "task environment." With the general part of GPS and appropriate task environments, computers can be programmed to solve trigonometric identities (Newell, Shaw, and Simon, 1959*a*), balance assembly lines (Tonge, 1960, 1961*a*), and compile computer programs (Simon, 1961*c*).

GPS, A PROGRAM THAT SIMULATES HUMAN THOUGHT

by Allen Newell & H. A. Simon

This article is concerned with the psychology of human thinking. It sets forth a theory to explain how some humans try to solve some simple formal problems. The research from which the theory emerged is intimately related to the field of information processing and the construction of intelligent automata, and the theory is expressed in the form of a computer program. The rapid technical advances in the art of programming digital computers to do sophisticated tasks have made such a theory feasible.

It is often argued that a careful line must be drawn between the attempt to *accomplish* with machines the same tasks that humans perform, and the attempt to *simulate* the processes humans actually use to accomplish these tasks. The program discussed in the report, GPS (General Problem Solver), maximally confuses the two approaches—with mutual benefit. GPS has previously been described as an attempt to build a problem-solving program (Newell, Shaw, and Simon, 1959*a*, 1960*a*), and in our own research it remains a major vehicle for exploring the area of artificial intelligence. Simultaneously, variants of GPS provide simulations of human behavior (Newell and Simon, 1961*a*). It is this latter aspect—the use of GPS as a theory of human problem-solving—that we want to focus on exclusively here, with special attention to the relation between the theory and the data.

As a context for the discussion that is to follow, let us make some brief comments on some history of psychology. At the beginning of this century the prevailing thesis in psychology was Associationism. It was an atomistic doctrine, which postulated a theory of hard little elements, either sensations or ideas, that became hooked or associated together without modifica-

tion. It was a mechanistic doctrine, with simple fixed laws of contiguity in time and space to account for the formation of new associations. Those were its assumptions. Behavior proceeded by the stream of associations: Each association produced its successors, and acquired new attachments with the sensations arriving from the environment.

In the first decade of the century a reaction developed to this doctrine through the work of the Wurzburg school. Rejecting the notion of a completely self-determining stream of associations, it introduced the task (*Aufgabe*) as a necessary factor in describing the process of thinking. The task gave direction to thought. A noteworthy innovation of the Wurzburg school was the use of systematic introspection to shed light on the thinking process and the contents of consciousness. The result was a blend of mechanics and phenomenalism, which gave rise in turn to two divergent antitheses, Behaviorism and the Gestalt movement.

The behavioristic reaction insisted that introspection was a highly unstable, subjective procedure, whose futility was amply demonstrated in the controversy on imageless thought. Behaviorism reformulated the task of psychology as one of explaining the response of organisms as a function of the stimuli impinging upon them and measuring both objectively. However, Behaviorism accepted, and indeed reinforced, the mechanistic assumption that the connections between stimulus and response were formed and maintained as simple, determinate functions of the environment.

The Gestalt reaction took an opposite turn. It rejected the mechanistic nature of the associationist doctrine but maintained the value of phenomenal observation. In many ways it continued the Wurzburg school's insistence that thinking was more than association—thinking has direction given to it by the task or by the set of the subject. Gestalt psychology elaborated this doctrine in genuinely new ways in terms of holistic principles of organization.

Today psychology lives in a state of relatively stable tension between the poles of Behaviorism and Gestalt psychology. All of us have internalized the major lessons of both: We treat skeptically the subjective elements in our experiments and agree that all notions must eventually be made operational by means of behavioral measures. We also recognize that a human being is a tremendously complex, organized system, and that the simple schemes of modern behavioristic psychology seem hardly to reflect this at all.

An Experimental Situation

In this context, then, consider the following situation. A human subject, a student in engineering in an American college, sits in front of a blackboard on which are written the following expressions:

$$(R \supset \sim P) \cdot (\sim R \supset Q) \mid \sim (\sim Q \cdot P)$$

This is a problem in elementary symbolic logic, but the student does not know it. He does know that he has twelve rules for manipulating expressions containing letters connected by "dots" ($\cdot$), "wedges" ($\vee$), "horseshoes" ($\supset$), and "tildes" ($\sim$), which stand respectively for "and," "or," "implies," and "not." These rules, given in Fig. 1, show that expressions of certain forms (at the tails of the arrows) can be transformed into expressions of somewhat different form (at the heads of the arrows). (Double arrows indicate transformations can take place in either direc-

Objects are formed by building up expressions from letters (P, Q, R, . . .) and connectives $\cdot$ (dot), $\vee$ (wedge), $\supset$ (horseshoe), and $\sim$ (tilde). Examples are P, $\sim Q$, $P \vee Q$, $\sim(R \supset S) \cdot \sim P$; $\sim\sim P$ is equivalent to P throughout.
Twelve rules exist for transforming expressions (where A, B, and C may be any expressions or subexpressions):

R 1. $A \cdot B \rightarrow B \cdot A$; $A \vee B \rightarrow B \vee A$

R 2. $A \supset B \rightarrow \sim B \supset \sim A$

R 3. $A \cdot A \longleftrightarrow A$; $A \vee A \longleftrightarrow A$

R 4. $A \cdot (B \cdot C) \longleftrightarrow (A \cdot B) \cdot C$; $A \vee (B \vee C) \longleftrightarrow (A \vee B) \vee C$

R 5. $A \vee B \longleftrightarrow \sim(\sim A \cdot \sim B)$

R 6. $A \supset B \longleftrightarrow \sim A \vee B$

R 7. $A \cdot (B \vee C) \longleftrightarrow (A \cdot B) \vee (A \cdot C)$; $A \vee (B \cdot C) \longleftrightarrow (A \vee B) \cdot (A \vee C)$

R 8. $A \cdot B \rightarrow A$; $A \cdot B \rightarrow B$ — Applies to main expression only.

R 9. $A \rightarrow A \vee X$ — Applies to main expression only.

R 10. $A, B \rightarrow A \cdot B$ — A and B are two main expressions.

R 11. $A, A \supset B \rightarrow B$ — A and $A \supset B$ are two main expressions.

R 12. $A \supset B, B \supset C \rightarrow A \supset C$ — $A \supset B$ and $B \supset C$ are two main expressions.

Example, showing subject's entire course of solution on problem:

1.	$(R \supset \sim P) \cdot (\sim R \supset Q)$	$\sim(\sim Q \cdot P)$
2.	$(\sim R \vee \sim P) \cdot (R \vee Q)$	Rule *6* applied to left and right of 1.
3.	$(\sim R \vee \sim P) \cdot (\sim R \supset Q)$	Rule *6* applied to left of 1.
4.	$R \supset \sim P$	Rule *8* applied to 1.
5.	$\sim R \vee \sim P$	Rule *6* applied to 4.
6.	$\sim R \supset Q$	Rule *8* applied to 1.
7.	$R \vee Q$	Rule *6* applied to 6.
8.	$(\sim R \vee \sim P) \cdot (R \vee Q)$	Rule *10* applied to 5. and 7.
9.	$P \supset \sim R$	Rule *2* applied to 4.
10.	$\sim Q \supset R$	Rule *2* applied to 6.
11.	$P \supset Q$	Rule *12* applied to 6. and 9.
12.	$\sim P \vee Q$	Rule *6* applied to 11.
13.	$\sim(P \cdot \sim Q)$	Rule *5* applied to 12.
14.	$\sim(\sim Q \cdot P)$	Rule *1* applied to 13. QED.

Figure 1. The task of symbolic logic.

Well, looking at the left hand side of the equation, first we want to eliminate one of the sides by using rule 8. It appears too complicated to work with first. Now – no, – no, I can't do that because I will be eliminating either the Q or the P in that total expression. I won't do that at first. Now I'm looking for a way to get rid of the horseshoe inside the two brackets that appear on the left and right sides of the equation. And I don't see it. Yeh, if you apply rule 6 to both sides of the equation, from there I'm going to see if I can apply rule 7.

Experimenter writes: 2. $(\sim R \vee \sim P) \cdot (R \vee Q)$

I can almost apply rule 7, but one R needs a tilde. So I'll have to look for another rule. I'm going to see if I can change that R to a tilde R. As a matter of fact, I should have used rule 6 on only the left hand side of the equation. So use rule 6, but only on the left hand side.

Experimenter writes: 3. $(\sim R \vee \sim P) \cdot (\sim R \supset Q)$

Now I'll apply rule 7 as it is expressed. Both – excuse me, excuse me, it can't be done because of the horseshoe. So – now I'm looking – scanning the rules here for a second, and seeing if I can change the R to a $\sim R$ in the second equation, but I don't see any way of doing it. (Sigh.) I'm just sort of lost for a second.

Figure 2. Subject's protocol on first part of problem.

tion.) The subject has practiced applying the rules, but he has previously done only one other problem like this. The experimenter has instructed him that his problem is to obtain the expression in the upper right corner from the expression in the upper left corner using the twelve rules. At any time the subject can request the experimenter to apply one of the rules to an expression that is already on the blackboard. If the transformation is legal, the experimenter writes down the new expression in the left-hand column, with the name of the rule in the right-hand column beside it. The subject's actual course of solution is shown beneath the rules in Fig. 1.

The subject was also asked to talk aloud as he worked; his comments were recorded and then transcribed into a "protocol,"—*i.e.*, a verbatim record of all that he or the experimenter said during the experiment. The initial section of this subject's protocol is reproduced in Fig. 2.

The Problem of Explanation

It is now proposed that the protocol of Fig. 2 constitutes data about human behavior that are to be explained by a psychological theory. But what are we to make of this? Are we back to the introspections of the Wurzburgers? And how are we to extract information from the behavior of a single subject when we have not defined the operational measures we wish to consider?

There is little difficulty in viewing this situation through behavioristic eyes. The verbal utterances of the subject are as much behavior as would

be his arm movements or galvanic skin responses. The subject was not introspecting; he was simply emitting a continuous stream of verbal behavior while solving the problem. Our task is to find a model of the human problem-solver that explains the salient features of this stream of behavior. This stream contains not only the subject's extemporaneous comments, but also his commands to the experimenter, which determine whether he solves the problem or not.

Although this way of viewing the behavior answers the questions stated above, it raises some of its own. How is one to deal with such variable behavior? Isn't language behavior considered among the most complex human behavior? How does one make reliable inferences from a single sample of data on a single subject?

The answers to these questions rest upon the recent, striking advances that have been made in computers, computer programming and artificial intelligence. We have learned that a computer is a general manipulator of symbols—not just a manipulator of numbers. Basically, a computer is a transformer of patterns. By suitable devices, most notably its addressing logic, these patterns can be given all the essential characteristics of linguistic symbols. They can be copied and formed into expressions. We have known this abstractly since Turing's work in the mid-thirties, but it is only recently that computers have become powerful enough to let us actually explore the capabilities of complex symbol manipulating systems.

For our purpose here, the most important branch of these explorations is the attempt to construct programs that solve tasks requiring intelligence. Considerable success has already been attained (Gelernter, 1959*b*; Kilburn et al., 1959; Minsky, 1961*a*; Newell, Shaw, and Simon, 1957*a*, 1958*b*; Samuel, 1959*a*; Tonge, 1960). These accomplishments form a body of ideas and techniques that allow a new approach to the building of psychological theories. (Much of the work on artificial intelligence, especially our own, has been partly motivated by concern for psychology; hence, the resulting rapprochement is not entirely coincidental.)

We may then conceive of an intelligent program that manipulates symbols in the same way that our subject does—by taking as inputs the symbolic logic expressions, and producing as ouputs a sequence of rule applications that coincides with the subject's. If we observed this program in operation, it would be considering various rules and evaluating various expressions, the same sorts of things we see expressed in the protocol of the subject. If the fit of such a program were close enough to the overt behavior of our human subject—*i.e.*, to the protocol—then it would constitute a good theory of the subject's problem-solving.

Conceptually the matter is perfectly straightforward. A program prescribes in abstract terms (expressed in some programming language) how

a set of symbols in a memory is to be transformed through time. It is completely analogous to a set of difference equations that prescribes the transformation of a set of numbers through time. Given enough information about an individual, a program could be written that would describe the symbolic behavior of that individual. Each individual would be described by a different program, and those aspects of human problem-solving that are not idiosyncratic would emerge as the common structure and content of the programs of many individuals.

But is it possible to write programs that do the kinds of manipulation that humans do? Given a specific protocol, such as the one of Fig. 2, is it possible to induct the program of the subject? How well does a program fit the data? The remainder of the report will be devoted to answering some of these questions by means of the single example already presented. We will consider only how GPS behaves on the first part of the problem, and we will compare it in detail with the subject's behavior as revealed in the protocol. This will shed considerable light on how far we can consider programs as theories of human problem-solving.

The GPS Program

We will only briefly recapitulate the GPS program, since our description will add little to what has already been published (Newell, Shaw, and Simon, 1959*a*, 1960*a*). GPS deals with a task environment consisting of *objects* which can be transformed by various *operators;* it detects *differences* between objects; and it organizes the information about the task environment into *goals*. Each goal is a collection of information that defines what constitutes goal attainment, makes available the various kinds of information relevant to attaining the goal, and relates the information to other goals. There are three types of goals:

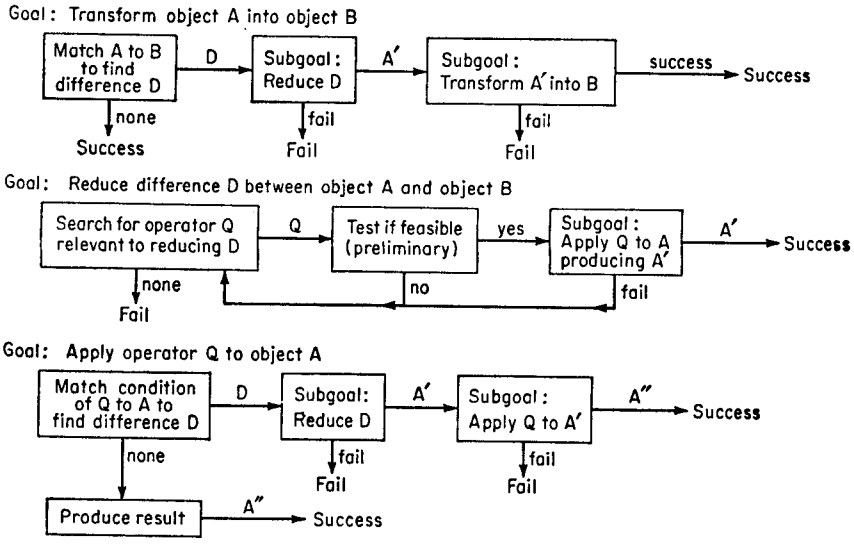
Transform object A into object B,
Reduce difference D between object A and object B,
Apply operator Q to object A.

For the task of symbolic logic, the objects are logic expressions; the operators are the twelve rules (actually the specific variants of them); and the differences are expressions like "change connective" or "add a term." Thus the objects and operators are given by the task; whereas the differences are something GPS brings to the problem. They represent the ways of relating operators to their respective effects upon objects.

Basically, the GPS program is a way of achieving a goal by setting up subgoals whose attainment leads to the attainment of the initial goal. GPS has various schemes, called methods, for doing this. Three crucial methods are presented in Fig 3, one method associated with each goal type.

Thus, to transform an object A into an object B, the objects are first matched—put into correspondence and compared element by element. If the match reveals a difference, D, between the two objects, then a subgoal is set up to reduce this difference. If this subgoal is attained, a new object, A′, is produced which (hopefully) no longer has the difference D when compared with object B. Then a new subgoal is created to transform A′ into B. If the transformation succeeds, the entire goal has been attained in two steps: from A to A′ and from A′ to B.

If the goal is to reduce the difference between two objects, the first step is to find an operator that is relevant to this difference. Relevance here



For the logic task of the text:

Feasibility test (preliminary):
Is the mean connective the same? (e.g., A·B → B fails against PvQ)
Is the operator too big? (e.g., (AvB)·(AvC) → Av(B·C) fails against P·Q)
Is the operator too easy? (e.g., A → A·A applies to anything)
Are the side conditions satisfied? (e.g., R8 applies only to main expressions)

Table of connections

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Add terms			×				×		×	×	×	×
Delete terms			×				×	×			×	×
Change connective					×	×	×					
Change sign					×							
Change lower sign		×			×	×						
Change grouping				×			×					
Change position	×	×										

× means some variant of the rule is relevant. GPS will pick the appropriate variant.

Figure 3. Methods for GPS.

means that the operator affects objects with respect to the difference. Operationally, relevance can be determined by applying the matching process already used to the input and output forms of the operators, due account being taken of variables. The results can be summarized in a table of connections, as shown in Fig. 3, which lists for each difference the operators that are relevant to it. This table also lists the differences that GPS recognizes. [This set is somewhat different from the one given in Newell, Shaw, and Simon (1959*a*); it corresponds to the program we will deal with in this report.] If a relevant operator, Q, is found, it is subjected to a preliminary test of feasibility, one version of which is given in Fig. 3. If the operator passes this test, a subgoal is set up to apply the operator to the object. If the operator is successfully applied, a new object, A′, is produced which is a modification of the original one in the direction of reducing the difference. (Of course, other modifications may also have occurred which nullify the usefulness of the new object.)

If the goal is to apply an operator, the first step is to see if the conditions of the operator are satisfied. The preliminary test above by no means guarantees this. If the conditions are satisfied, then the output, A, can be generated. If the conditions are not satisfied, then some difference, D, has been detected and a subgoal is created to reduce this difference, just as with the transform goal. Similarly, if a modified object, A′, is obtained, a new subgoal is formed to try to apply the operator to this new object.

These methods form a recursive system that generates a tree of subgoals in attempting to attain a given goal. For every new difficulty that is encountered a new subgoal is created to overcome this difficulty. GPS has a number of tests it applies to keep the expansion of this goal tree from proceeding in unprofitable directions. The most important of these is a test which is applied to new subgoals for reducing differences. GPS contains an ordering of the differences, so that some differences are considered easier than others. This ordering is given by the table of connections in Fig. 3, which lists the most difficult differences first. GPS will not try a subgoal if it is harder than one of its supergoals. It will also not try a goal if it follows an easier goal. That is, GPS insists on working on the hard differences first and expects to find easier ones as it goes along. The other tests that GPS applies involve external limits (*e.g.*, a limit on the total depth of a goal tree it will tolerate) and whether new objects or goals are identical to ones already generated.

GPS on the Problem

The description we have just given is adequate to verify the reasonableness, although not the detail, of a trace of GPS's behavior on a specific problem. (In particular we have not described how the two-line rules, R10

through R12, are handled, since they do not enter into the protocol we are examining.) In Fig. 4, we give the trace on the initial part of problem D1. Indentation is used to indicate the relation of a subgoal to a goal. Although the methods are not shown, they can clearly be inferred from the goals that occur.

The initial problem is to transform L1 into L0. Matching L1 to L0 reveals that there are R's in L1 and no R's in L0. This difference leads to the formulation of a reduce goal, which for readability has been given its functional name, *Delete*. The attempt to reach this goal leads to a search for rules which finds rule 8. Since there are two forms of rule 8, both of which are admissible, GPS chooses the first. (Variants of rules are not indicated, but can be inferred easily from the trace.) Since rule 8 is

```
L0  ~(~Q·P)
L1  (R⊃~P)·(~R⊃Q)

GOAL 1  TRANSFORM L1 INTO L0
     GOAL 2  DELETE R FROM L1
          GOAL 3  APPLY R8 TO L1
               PRODUCES L2 R⊃~P

     GOAL 4  TRANSFORM L2 INTO L0
          GOAL 5  ADD Q TO L2
               REJECT

     GOAL 2
          GOAL 6  APPLY R8 TO L1
               PRODUCES L3  ~R⊃Q

     GOAL 7  TRANSFORM L3 INTO L0
          GOAL 8  ADD P TO L3
               REJECT

     GOAL 2
          GOAL 9  APPLY R7 TO L1
               GOAL 10  CHANGE CONNECTIVE TO V IN LEFT L1
                    GOAL 11  APPLY R6 TO LEFT L1
                         PRODUCES L4  (~R∨~P)·(~R⊃Q)

               GOAL 12  APPLY R7 TO L4
                    GOAL 13  CHANGE CONNECTIVE TO ∨ IN RIGHT L4
                         GOAL 14  APPLY R6 TO RIGHT L4
                              PRODUCES L5  (~R∨~P)·(R∨Q)

                    GOAL 15  APPLY R7 TO L5
                         GOAL 16  CHANGE SIGN OF LEFT RIGHT L5
                              GOAL 17  APPLY R6 TO RIGHT L5
                                   PRODUCES L6  (~R∨~P)·(~R⊃Q)

                         GOAL 18  APPLY R7 TO L6
                              GOAL 19  CHANGE CONNECTIVE TO ∨
                                   IN RIGHT L6
                                   REJECT

                         GOAL 16
                              NOTHING MORE

                    GOAL 13
                         NOTHING MORE

               GOAL 10
                    NOTHING MORE
```

Figure 4. Trace of GPS on first part of problem.

applicable, a new object, L2, is produced. Following the method for transform goals, at the next step a new goal has been generated: to transform L2 into L0. This in turn leads to another reduce goal: to restore a Q to L2. But this goal is rejected by the evaluation, since adding a term is more difficult than deleting a term. GPS then returns to goal 2 and seeks another rule which will delete terms. This time it finds the other form of rule 8 and goes through a similar excursion, ending with the rejection of goal 8.

Returning again to goal 2 to find another rule for deleting terms, GPS obtains rule 7. It selects the variant $(A \vee B) \cdot (A \vee C) \rightarrow A \vee (B \cdot C)$, since only this one both decreases terms and has a dot as its main connective. Rule 7 is not immediately applicable; GPS first discovers that there is a difference of connective in the left subexpression, and then that there is one in the right subexpression. In both cases it finds and applies rule 6 to change the connective from horseshoe to wedge, obtaining successively L4 and L5. But the new expression reveals a difference in sign, which leads again to rule 6—that is, to the same rule as before, but perceived as accomplishing a different function. Rule 6 produces L6, which happens to be identical with L4 although GPS does not notice the identity here. This leads, in goal 19, to the difference in connective being redetected; whereupon the goal is finally rejected as representing no progress over goal 13. Further attempts to find alternative ways to change signs or connectives fail to yield anything. This ends the episode.

Comparison of the GPS Trace with the Protocol

We now have a highly detailed trace of what GPS did. What can we find in the subject's protocol that either confirms or refutes the assertion that this program is a detailed model of the symbol manipulations the subject is carrying out? What sort of correspondence can we expect? The program does not provide us with an English language output that can be put into one-one correspondence with the words of the subject. We have not even given GPS a goal to "do the task and talk at the same time," which would be a necessary reformulation if we were to attempt a correspondence in such detail. On the other hand, the trace, backed up by our knowledge of how it was generated, does provide a complete record of all the task content that was considered by GPS, and the order in which it was taken up. Hence, we should expect to find every feature of the protocol that concerns the task mirrored in an essential way in the program trace. The converse is not true, since many things concerning the task surely occurred without the subject's commenting on them (or even being aware of them). Thus, our test of correspondence is one-sided but exacting.

Let us start with the first sentence of the subject's protocol (Fig. 2):

> *Well, looking at the left-hand side of the equation, first we want to eliminate one of the sides by using rule 8.*

We see here a desire to decrease L1 or eliminate something from it, and the selection of rule 8 as the means to do this. This stands in direct correspondence with goals 1, 2, and 3 of the trace.

Let us skip to the third and fourth sentences:

> *Now—no,—no, I can't do that because I will be eliminating either the Q or the P in that total expression. I won't do that at first.*

We see here a direct expression of the covert application of rule 8, the subsequent comparison of the resulting expression with LO, and the rejection of this course of action because it deletes a letter that is required in the final expression. It would be hard to find a set of words that expressed these ideas more clearly. Conversely, if the mechanism of the program (or something essentially similar to it) were not operating, it would be hard to explain why the subject uttered the remarks that he did.

One discrepancy is quite clear. The subject handled both forms of rule 8 together, at least as far as his comment is concerned. GPS, on the other hand, took a separate cycle of consideration for each form. Possibly the subject followed the program covertly and simply reported the two results together. However, we would feel that the fit was better if GPS had proceeded something as follows:

```
GOAL 2 DELETE R FROM L1
     GOAL 3 APPLY R8 TO L1
          PRODUCES L2 R⊃~P OR ~R⊃Q

GOAL 4 TRANSFORM L2 INTO L0
     GOAL 5 ADD Q TO R⊃~P OR ADD P TO ~R⊃Q
          REJECT
```

We will consider further evidence on this point later.

Let us return to the second sentence, which we skipped over:

> *It appears too complicated to work with first.*

Nothing in the program is in simple correspondence with this statement, though it is easy to imagine some possible explanations. For example, this could merely be an expression of the matching—of the fact that L1 is such a big expression that the subject cannot absorb all its detail. There is not enough data locally to determine what part of the trace should correspond to this statement, so the sentence must stand as an unexplained element of the subject's behavior.

Now let us consider the next few sentences of the protocol:

> *Now I'm looking for a way to get rid of the horseshoe inside the two brackets that appear on the left and right side of the equation. And I don't see it. Yeh, if you apply rule 6 to both sides of the equation, from there I'm going to see if I can apply rule 7.*

This is in direct correspondence with goals 9 through 14 of the trace. The comment at the end makes it clear that applying rule 7 is the main concern and that changing connectives is required in order to accomplish this. Further, the protocol shows clearly that rule 6 was selected as the means. All three rule selections provide some confirmation that a preliminary test for feasibility was made by the subject—as by GPS—in the reduce goal method. If there was not selection on the main connective, why wasn't rule 5 selected instead of rule 6? Or why wasn't the $(A \cdot B) \vee (A \cdot C) \rightarrow A \cdot (B \vee C)$ form of rule 7 selected?

However, there is a discrepancy between trace and protocol, for the subject handles both applications of rule 6 simultaneously, (and apparently was also handling the two differences simultaneously); whereas GPS handles them sequentially. This is similar to the discrepancy noted earlier in handling rule 8. Since we now have two examples of parallel processing, it is likely that there is a real difference on this score. Again, we would feel better if GPS proceeded somewhat as follows:

```
GOAL 9 APPLY R7 TO L1
     GOAL 10 CHANGE CONNECTIVE TO V IN LEFT L1 AND RIGHT L1
          GOAL 11 APPLY R6 TO LEFT L1 AND RIGHT L1
               PRODUCES L5 (~R V ~P) • (R V Q)
```

A common feature of both these discrepancies is that forming the compound expressions does not complicate the methods in any essential way. Thus, in the case involving rule 8, the two results stem from the same input form, and require only the single match. In the case involving rule 7, a single search was made for a rule and the rule applied to both parts simultaneously, just as if only a single unit was involved.

There are two aspects in which the protocol provides information that the program is not equipped to explain. First, the subject handled the application of rule 8 covertly but commanded the experimenter to make the applications of rule 6 on the board. The version of GPS used here did not make any distinction between internal and external actions. To this extent it fails to be an adequate model. The overt-covert distinction has consequences that run throughout a problem, since expressions on the blackboard have very different memory characteristics from expressions generated only in the head. Second, this version of GPS does not simulate the search process sufficiently well to provide a correspondent to "And I don't see it. Yeh, . . .". This requires providing a facsimile of

the rule sheet, and distinguishing search on the sheet from searches in the memory.

The next few sentences read:

> *I can almost apply rule 7, but one R needs a tilde. So I'll have to look for another rule. I'm going to see if I can change that R to a tilde R.*

Again the trace and the protocol agree on the difference that is seen. They also agree that this difference was not attended to earlier, even though it was present. Some fine structure of the data also agrees with the trace. The right-hand R is taken as having the difference (R to $\sim$ R) rather than the left-hand one, although either is possible. This preference arises in the program (and presumably in the subject) from the language habit of working from left to right. It is not without consequences, however, since it determines whether the subject goes to work on the left side or the right side of the expression; hence, it can affect the entire course of events for quite a while. Similarly, in the rule 8 episode the subject apparently worked from left to right and from top to bottom in order to arrive at "Q or P" rather than "P or Q." This may seem like concern with excessively detailed features of the protocol, yet those details support the contention that what is going on inside the human system is quite akin to the symbol manipulations going on inside GPS.

The next portion of the protocol is:

> *As a matter of fact, I should have used rule 6 on only the left-hand side of the equation. So use 6, but only on the left-hand side.*

Here we have a strong departure from the GPS trace, although, curiously enough, the trace and the protocol end up at the same spot, $(\sim R \vee \sim P) \cdot (\sim R \supset Q)$. Both the subject and GPS found rule 6 as the appropriate one to change signs. At this point GPS simply applied the rule to the current expression; whereas the subject went back and corrected the previous application. Nothing exists in the program that corresponds to this. The most direct explanation is that the application of rule 6 in the inverse direction is perceived by the subject as undoing the previous application of rule 6. After following out this line of reasoning, he then takes the simpler (and less foolish-appearing) alternative, which is to correct the original action.

The final segment of the protocol reads:

> *Now I'll apply rule 7 as it is expressed. Both—excuse me, excuse me, it can't be done because of the horseshoe. So—now I'm looking—scanning the rules here for a second, and seeing if I can change the R to $\sim R$ in the second equation, but I don't see any way of doing it. (Sigh). I'm just sort of lost for a second.*

The trace and the protocol are again in good agreement. This is one of the few self-correcting errors we have encountered. The protocol records the futile search for additional operators to affect the differences of sign and connective, always with negative results. The final comment of mild despair can be interpreted as reflecting the impact of several successive failures.

Summary of the Fit of the Trace to the Protocol

Let us take stock of the agreements and disagreements between the trace and the protocol. The program provides a complete explanation of the subject's task behavior with five exceptions of varying degrees of seriousness.

There are two aspects in which GPS is unprepared to simulate the subject's behavior: in distinguishing between the internal and external worlds, and in an adequate representation of the spaces in which the search for rules takes place. Both of these are generalized deficiencies that can be remedied. It will remain to be seen how well GPS can then explain data about these aspects of behavior.

The subject handles certain sets of items in parallel by using compound expressions; whereas GPS handles all items one at a time. In the example examined here, no striking differences in problem solving occur as a result, but larger discrepancies could arise under other conditions. It is fairly clear how GPS could be extended to incorporate this feature.

There are two cases in which nothing corresponds in the program to some clear task-oriented behavior in the protocol. One of these, the early comment about "complication," seems to be mostly a case of insufficient information. The program is making numerous comparisons and evaluations which could give rise to comments of the type in question. Thus this error does not seem too serious. The other case, involving the "should have . . ." passage, does seem serious. It clearly implies a mechanism (maybe a whole set of them) that is not in GPS. Adding the mechanism required to handle this one passage could significantly increase the total capabilities of the program. For example, there might be no reasonable way to accomplish this except to provide GPS with a little continuous hindsight about its past actions.

An additional general caution must be suggested. The quantity of data is not large considering the size and complexity of the program. This implies that there are many degrees of freedom available to fit the program to the data. More important, we have no good way to assess how many relevant degrees of freedom a program possesses, and thus to know how easy it is to fit alternative programs. All we do know is that numerous minor modifications could certainly be made, but that no one has

proposed any major alternative theories that provide anything like a comparably detailed explanation of human problem-solving data.

It would help if we knew something of how idiosyncratic the program was. We have discussed it here only in relation to one sample of data for one subject. We know enough about subjects on logic problems to assert that the same mechanisms show up repeatedly, but we cannot discuss these data here in detail. In addition, several recent investigations more generally support the concept of information processing theories of human thinking (Bruner et al., 1956; Feigenbaum, 1961*a*; Feldman, 1961*a*; Hovland and Hunt, 1960; Miller et al., 1960).

Conclusion

We have been concerned in this report with showing that the techniques that have emerged for constructing sophisticated problem-solving programs also provide us with new, strong tools for constructing theories of human thinking. They allow us to merge the rigor and objectivity associated with Behaviorism with the wealth of data and complex behavior associated with the Gestalt movement. To this end their key feature is not that they provide a general framework for understanding problem-solving behavior (although they do that, too), but that they finally reveal with great clarity that the free behavior of a reasonably intelligent human can be understood as the product of a complex but finite and determinate set of laws. Although we know this only for small fragments of behavior, the depth of the explanation is striking.

section 2

Verbal Learning and Concept Learning

One of the principal topics in the study of human thinking has been learning—the process by which behavior is modified over time. Human learning has been studied in an experimental situation in which the subject is presented with a pair of items, one of which is called a *stimulus* and the other a *response*. After seeing a list of such pairs, the subject is presented with only the stimulus member of a pair and asked to reply with the response member of the pair. After his reply, the correct response is indicated. A large number of variations of this basic experiment have been explored in an effort to study the learning process.

In a popular verbal learning experiment, both the stimulus and the response are unique, three-letter nonsense syllables. To study learning in this experiment, investigators have varied the number of nonsense syllables to be learned, the degree of similarity between the syllables, the number and order of lists learned in a given experiment, the speed of presentation of the material, and other aspects of the experimental situation. The experimenters have been interested in the effect of these variations on the length of time and the number of repetitions required to learn a list, the number of errors made, and the ability to retain the list. In his model of verbal learning behavior, Feigenbaum offers a model of the information processing activity underlying human discrimination and association learning. His model accounts for many of the phenomena observed in these experiments. The basic processes of the model create a network of tests, or dis-

criminators, which distinguish items from each other. Other processes store associative cues which link stimuli with responses.

In a typical concept learning experiment, the subject is also presented with a series of stimuli and a series of associated responses. The stimuli may be geometric figures, nonsense syllables, numbers, etc. There are usually only two responses, *e.g.*, "yes" or "no," and these responses are associated with certain characteristics of the stimuli. While the principal task in the standard verbal learning experiment is to distinguish each stimulus so that the appropriate response may be associated with it, the principal task in the concept learning experiment is to find what each class of stimuli has in common with the other classes. To test his understanding of the concept, the subject may be asked to state the concept and/or produce the appropriate response to some additional stimuli. As in the verbal learning experiments, a large number of variations of this basic concept learning experiment have been conducted.

In the second article in this section, Hunt and Hovland present an information processing model which is consistent with many of the phenomena observed in these experiments.

Learning is of interest not only to psychologists studying human behavior but also to artificial intelligence researchers interested in improving the performance of computer programs. The interested reader is referred to the discussion of Minsky (pages 425 to 435), to Samuel's description of his work on learning with his checkers program (pages 71 to 105), and to the report of Newell, Shaw, and Simon on the Logic Theorist (pages 109 to 133). Feigenbaum and Simon (1961*b*) have discussed some of the implications for artificial intelligence of the verbal learning model described in this section. Hunt (1962) has constructed some other models of concept formation which are relevant to artificial intelligence.

Edward Feigenbaum is a member of the faculty of the School of Business Administration, University of California at Berkeley.

E. B. Hunt is a lecturer in psychology, University of Sydney, Sydney, Australia.

The late C. I. Hovland was Sterling Professor of Psychology at Yale University.

THE SIMULATION OF VERBAL LEARNING BEHAVIOR

by Edward A. Feigenbaum

The purpose of this report is to describe in detail an information processing model of elementary human symbolic learning processes. This model is realized by a computer program called the Elementary Perceiver and Memorizer (EPAM).

The EPAM program is the precise statement of an information processing theory of verbal learning that provides an alternative to other verbal learning theories which have been proposed.[1] It is the result of an attempt to state quite precisely a parsimonious and plausible mechanism sufficient to account for the rote learning of nonsense syllables. The critical evaluation of EPAM must ultimately depend not upon the interest which it may have as a learning machine, but upon its ability to explain and predict the phenomena of verbal learning.

I should like to preface my discussion of the simulation of verbal learning with some brief remarks about the class of information processing models of which EPAM is a member.

a. These are models of mental processes, not brain hardware. They are *psychological* models of mental function. No physiological or neurological assumptions are made, nor is any attempt made to explain information processes in terms of more elementary neural processes.

b. These models conceive of the brain as an *information processor* with sense organs as input channels, effector organs as output devices, and with internal programs for testing, comparing, analyzing, rearranging, and storing information.

[1] Examples of quantitative (or quasi-quantitative) theories of verbal learning are those of Hull et al. (1940), Gibson, (1940), and Atkinson (1954).

c. The central processing mechanism is assumed to be serial; *i.e.,* capable of doing only one (or a very few) things at a time.

d. These models use as a basic unit the *information symbol; i.e.,* a pattern of bits which is assumed to be the brain's internal representation of environmental data.

e. These models are essentially *deterministic,* not probabilistic. Random variables play no fundamental role in them.

The Basic Experiment

Early in the history of psychology, the psychologist invented an experiment to simplify the study of human verbal learning. This "simple" experiment is the rote memorization of nonsense syllables in associate pairs or serial lists.

The items to be memorized are generally three-letter words having consonant letters on each end and a vowel in the middle. Nonsense syllables are chosen in such a way that the three-letter combinations have no ordinary English meaning. For example, CAT is not a nonsense syllable, but XUM is.[2]

In one basic variation, the rote memory experiment is performed as follows:

a. A set of nonsense syllables is chosen and the syllables are paired, making, let us say, 12 pairs.

b. A subject is seated in front of a viewing apparatus and the syllables are shown to him, one pair at a time.

c. First, the left-hand member of the pair (*stimulus item*) is shown. The subject tries to say the second member of the pair (*response item*).

d. After a short interval, the response item is exposed so that both stimulus and response items are simultaneously in view.

e. After a few seconds, the cycle repeats itself with a new pair of syllables. This continues until all pairs have been presented (a *trial*).

f. Trials are repeated, usually until the subject is able to give the correct response to each stimulus. There is a relatively short time interval between trials.

g. For successive trials the syllables are reordered randomly. This style of carrying out the experiment is called *paired-associates presentation.*

The other basic variant of the experiment is called *serial-anticipation presentation.* The nonsense syllables (say, 10 or 12 items) are arranged

[2] People will defy an experimenter's most rigorous attempt to keep the nonsense syllables association-free. Lists of nonsense syllables have been prepared, ordering syllables on the basis of their so-called "association value," in order to permit the experimenter to control "meaningfulness."

in a serial list, the order of which is not changed on successive trials. When he is shown the nth syllable, the subject is to respond with the $(n + 1)$st syllable. A few seconds later, the $(n + 1)$st syllable is shown and the subject is to respond with the $(n + 2)$d syllable, and so on. The experiment terminates when the subject is able to correctly anticipate all of the syllables.

Numerous variations on this experimental theme have been performed.[3] The phenomena of rote learning are well studied, stable, and reproducible. For example, in the typical behavioral output of a subject, one finds:

a. Failures to respond to a stimulus are more numerous than overt errors.

b. Overt errors are generally attributable to confusion by the subject between similar stimuli or similar responses.

c. Associations which are given correctly over a number of trials sometimes are then forgotten, only to reappear and later disappear again. This phenomenon has been called oscillation.[4]

d. If a list x of syllables or syllable pairs is learned to the criterion; then a list y is similarly learned; and finally retention of list x is tested; the subject's ability to give the correct x responses is degraded by the interpolated learning. The degradation is called retroactive inhibition. The overt errors made in the retest trial are generally intrusions from the list y. The phenomenon disappears rapidly. Usually after the first retest trial, list x has been relearned back to criterion.

e. As one makes the stimulus syllables more and more similar, learning takes more trials.

The Information Processing Model

This section describes the processes and structures of EPAM.

EPAM is not a model for a particular subject. In this respect it is to be contrasted with the binary choice models of particular subjects which Feldman describes (1961*a*). The fact is that individual differences play only a small part in the results of the basic experiment described above.

It is asserted that there are certain elementary information processes which an individual must perform if he is to discriminate, memorize and associate verbal items, and that these information processes participate in all the cognitive activity of all individuals.[5]

It is clear that EPAM does not yet embody a complete set of such

[3] For an extended treatment of this subject, see Hovland, *Human Learning and Retention* in Stevens (1951).

[4] By Hull (1935). Actually he called it "oscillation at the threshold of recall," reflecting his theoretical point of view.

[5] Some information processing models are conceived as models of the mental

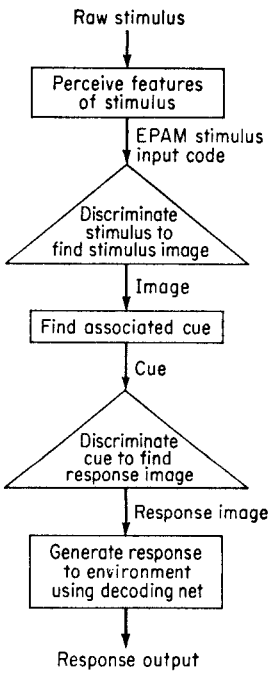


Figure 1. EPAM performance process for producing the response associated with a stimulus.

processes. It is equally clear that the processes EPAM has now are essential and basic.

OVERVIEW: PERFORMANCE AND LEARNING

Conceptually, EPAM can be broken down into two subsystems, a performance system and a learning system. In the performance mode, EPAM produces responses to stimulus items. In the learning mode, EPAM learns to discriminate and associate items.

The performance system is the simpler of the two. It is sketched in Fig. 1. When a stimulus is noticed, a *perceptual process* encodes it, producing an internal representation (an *input code*). A *discriminator* sorts the input code in a *discrimination net* (a tree of tests and branches) to find a stored *image* of the stimulus. A response *cue* associated with the image is found, and fed to the discriminator. The discriminator sorts the cue in the net and finds the response image, the stored form of the response. The response image is then decoded by a *response generator* letter by letter in another discrimination net into a form suitable for output. The response is then produced as output.

The processes of the learning system are more complex. The discrimination learning process builds discriminations by growing the net of tests and branches. The association process builds associations between images by storing response cues with stimulus images. These processes will be described fully in due course.

---

function of particular subjects; *e.g.*, Feldman's Binary Choice Model (1959). Others treat the general subject as EPAM does. Still others are mixed in conception, asserting that certain of the processes of the model are common for all subjects while other processes may vary from subject to subject; *e.g.*, the General Problem Solver of Newell, Shaw, and Simon (1959*a*). Alternatively, information processing models may also be categorized according to how much of the processing is "hard core" (*i.e.*, necessary and invariant) as opposed to "strategic" (*i.e.*, the result of strategy choice by control processes). I suggest the obvious: that models of strategies for information processing will tend to be models of particular subjects. As exemplars, Lindsay's Reading Machine (1960), a "hard-core" model, treats the general subject; Wickelgren's model of the conservative focusing strategy in concept attainment (Wickelgren, 1962; Bruner, Goodnow, and Austin, 1956), a pure strategy model, can predict only the behavior of particular subjects.

The succeeding sections on the information processing model give a detailed description of the processes and structures of both systems.

INPUT TO EPAM: INTERNAL REPRESENTATIONS OF EXTERNAL DATA

The following are the assumptions about the symbolic input process when a nonsense syllable is presented to the learner. A *perceptual system* receives the raw external information and codes it into *internal symbols.* These internal symbols contain descriptive information about features of external stimuli. For unfamiliar 3-letter nonsense symbols, it is assumed that the coding is done in terms of the individual letters, for these letters are familiar and are well-learned units for the adult subject.[6] The end result of the perception process is an internal representation of the nonsense syllable—a list of internal symbols (*i.e.,* a list of lists of bits) containing descriptive information about the letters of the nonsense syllable. Using Minsky's terminology (1961*a*), this is the "character" of the nonsense syllable.

I have not actually programmed this perception process. For purposes of this simulation, I have assigned coded representations for the various letters of the alphabet based on 15 different geometrical features of letters. For purposes of exploring and testing the model, at present all that is really needed of the input codes is:

a. that the dimensions of a letter code be related in some reasonable way to features of real letters.

b. that the letter codes be highly redundant, that is, include many more dimensions than is necessary to discriminate the letters of the alphabet.

To summarize, the internal representation of a nonsense syllable is a list of lists of bits, each sublist of bits being a highly redundant code for a letter of the syllable.

Given a sequence of such inputs, the essence of the learner's problem is twofold: first, to *discriminate* each code from the others already learned, so that differential response can be made; second, to *associate* information about a "response" syllable with the information about a "stimulus" syllable so that the response can be retrieved if the stimulus is presented.

DISCRIMINATING AND MEMORIZING: GROWING TREES OF IMAGES

I shall deal with structure first and reserve my discussion of process for a moment.

Discrimination Net. The primary information structure in EPAM is the

[6] The basic perception mechanism I have in mind is much the same as that of Selfridge (1955) and Dinneen (1955), whose computer program scanned letters and perceived simple topological features of these letters.

discrimination net. It embodies in its structure at any moment all of the discrimination learning that has taken place up to a given time. As an information structure it is no more than a familiar friend: a sorting tree or decoding network. Figure 2 shows a small net. At the terminals of the net are lists called *image lists,* in which symbolic information can be stored. At the nodes of the net are stored programs, called *tests,* which examine characteristics of an input code and signal branch left or branch right. On each image list will be found a list of symbols called the *image.* An image is a partial or total copy of an input code. I shall use these names in the following description of net processes.

Net Interpreter. The discrimination net is examined and altered by a number of processes, most important of which is the *net interpreter.* The net interpreter sorts an input code in the net and produces the image list associated with that input code. *This retrieval process is the essence of a purely associative memory: the stimulus information itself leads to the retrieval of the information associated with that stimulus.* The net interpreter is a very simple process. It finds the test in the topmost node of the tree and executes this program. The resulting signal tells it to branch left or branch right to find the succeeding test. It executes this, tests its branches again, and repeats the cycle until a terminal is found. The name of the image list is produced, and the process terminates. This is the discriminator of the performance system which sorts items in a static net.

Discrimination Learning. The discrimination learning process of the learning system grows the net. Initially we give the learning system no discrimination net but only a set of simple processes for growing nets and storing new images at the terminals.

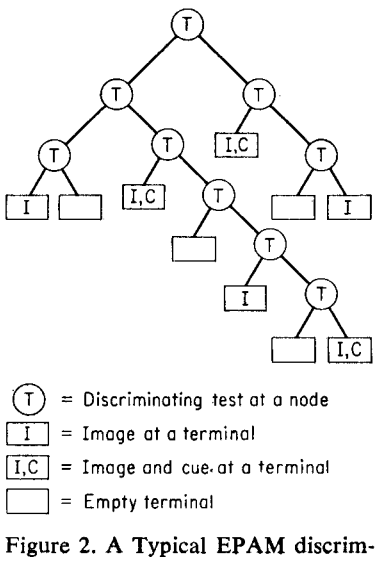


Figure 2. A Typical EPAM discrimination net.

To understand how the discrimination and memorization processes work, let us examine in detail a concrete example from the learning of nonsense syllables. Suppose that the first stimulus-response associate pair on a list has been learned. (Ignore for the moment the question of how the association link is actually formed.) Suppose that the first syllable pair was DAX-JIR. The discrimination net at this point has the simple two-branch structure shown in Fig. 3. Because the syllables differ in their first letter, Test 1 will probably be

a test of some characteristic on which the letters D and J differ. No more tests are necessary at this point.

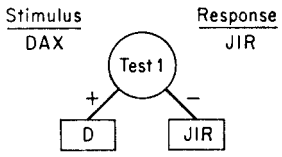


Figure 3. Discrimination net after the learning of the first two items. Cues are not shown. Condition: no redundant tests added. Test 1 is a first-letter test.

Notice that the image of JIR which is stored is a full image. Full response images must be stored—to provide the information for *producing* the response; but only partial stimulus images need be stored—to provide the information for *recognizing* the stimulus. How much stimulus image information is required the learning system determines for itself as it grows its discrimination net, and makes errors which it diagnoses as inadequate discrimination.

To pursue our simple example, suppose that the next syllable pair to be learned is PIB-JUK. There are no storage terminals in the net, as it stands, for the two new items. In other words, the net does not have the discriminative capability to contain more than two items. The input code for PIB is sorted by the net interpreter. Assume that Test 1 sorts it down the plus branch of Fig. 3. As there are differences between the incumbent image (with first letter D) and the new code (with first letter P) an attempt to store an image of PIB at this terminal would destroy the information previously stored there.

Clearly what is needed is the ability to discriminate further. A match for differences between the incumbent image and the challenging code is performed. When a difference is found, a new test is created to discriminate upon this difference. The new test is placed in the net at the point of failure to discriminate, an image of the new item is created, and both images—incumbent and new—are stored in terminals along their appropriate branches of the new test, and the conflict is resolved.[7] The net as it now stands is shown in Fig. 4. Test 2 is seen to discriminate on some difference between the letters P and D.

The input code for JUK is now sorted by the net interpreter. Since Test

[7] With the processes just described, the discrimination net would be grown each time a new item was to be added to the memory. But from an informational processing standpoint, the matching and net-growing processes are the most time-consuming in the system. In general, with little additional effort, more than one difference can be detected, and more than one discriminating test can be added to the net. Each redundant test placed in the net gives one "empty" image list. At some future time, if an item is sorted to this empty image list, an image can be stored without growing the net. There is a happy medium between small nets which must be grown all the time and large nets replete with redundant tests and a wasteful surplus of empty image lists. Experimentation with this "structural parameter" has been done and it has been found that for this study one or two redundant tests per growth represents the happy medium. However, I would not care to speak of the generality of this particular result.

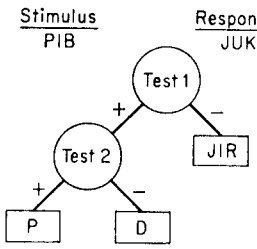


Figure 4. Discrimination net of Fig. 3 after the learning of stimulus item, PIB. Test 2 is a first-letter test.

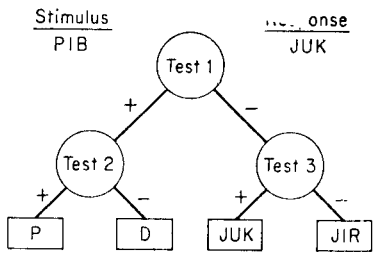


Figure 5. Discrimination net of Fig. 4 after the learning of the response item, JUK. Test 3 is a third-letter test.

1 cannot detect the difference between the input codes for JUK and JIR (under our previous assumption), JUK is sorted to the terminal containing the image of JIR. The match for differences takes place. Of course, there are no first-letter differences. But there are differences between the incumbent image and the new code in the second and third letters.

Noticing Order. In which letter should the matching process next scan for differences? In a serial machine like EPAM, this scanning must take place in some order. This order need not be arbitrarily determined and fixed. It can be made variable and adaptive. To this end EPAM has a *noticing order for letters of syllables,* which prescribes at any moment a letter-scanning sequence for the matching process. Because it is observed that subjects generally consider end letters before middle letters, the noticing order is initialized as follows: first letter, third letter, second letter. When a particular letter being scanned yields a difference, this letter is promoted up one position on the noticing order. Hence, letter positions relatively rich in differences quickly get priority in the scanning. In our example, because no first-letter differences were found between the image of JIR and code for JUK, the third letters are scanned and a difference is found (between R and K). A test is created to capitalize on this third-letter difference and the net is grown as before. The result is shown in Fig. 5. The noticing order is updated; third letter, promoted up one, is at the head.

Learning of subsequent items proceeds in the same way, and we shall not pursue the example further.

ASSOCIATING IMAGES: RETRIEVAL USING CUES

The discrimination net and its interpreter associate codes of external objects with internal image lists and images. But the basic rote learning experiment requires that stimulus information somehow lead to response information and a response. The discrimination net concept can be used for the association of internal images with each other (*i.e.*, response with stimulus) with very little addition to the basic mechanism.

An association between a stimulus image and a response image is accomplished by storing with the stimulus image some of the coded information about the response. This information is called *the cue.* A cue is of the same form as an input code, but generally contains far less information than an input code. A cue to an associated image can be stored in the discrimination net by the net interpreter to retrieve the associated image. If, for example, in the net of Fig. 3 we had stored with the stimulus image the letter J as a cue to the response JIR, then sorting this cue would have correctly retrieved the response image. *An EPAM internal association is built by storing with the stimulus image information sufficient to retrieve the response image from the net at the moment of association.*

The association process determines how much information is sufficient by trial and error. The noticing order for letters is consulted, and the first-priority letter is added to the cue. The cue is then sorted by the net interpreter and a response image is produced. It might be the wrong response image; for if a test seeks information which the cue does not contain, the interpreter branches left or right randomly (with equal probabilities) at this test.[8] During association, the selection of the wrong response is immediately detectable (by a matching process) because the response input code is available. The next-priority letter is added to the cue and the process repeats until the correct response image is retrieved. The association is then considered complete.

Note two important possibilities. First, by the process just described, a cue which is really not adequate to guarantee retrieval of the response image may by happenstance give the correct response image selection during association. This "luck" usually gives rise to response errors at a later time.

Second, suppose that the association building process does its job thoroughly. The cue which it builds is sufficient to retrieve the response image *at one particular time,* the time at which the two items were associated. If, at some future time, the net is grown to encompass new images being added to the memory, then a cue which previously was sufficient to correctly retrieve a response image may no longer be sufficient to retrieve that response image. In EPAM, association links are "dated," and ever vulnerable to interruption by further learning. Responses may be "unlearned" or "forgotten" temporarily, not because the response information has been destroyed in the memory, but because the information has been temporarily lost in a growing network. If an association failure of this type can be detected through feedback from the environmental or ex-

[8] This is the only use of a random variable in EPAM. We do not like it. We use it only because we have not yet discovered a plausible and satisfying adaptive mechanism for making the decision. The random mechanism does, however, give better results than the go-one-way-all-the-time mechanism which has also been used.

perimental situation, then the trouble is easily remedied by adding additional response information to the cue. If not, then the response may be more or less permanently lost in the net. The significance of this phenomenon will perhaps be more easily appreciated in the discussion of results of the EPAM simulation.

RESPONDING: INTERNAL AND EXTERNAL

A conceptual distinction is made between the process by which EPAM selects an internal response image and the process by which it converts this image into an output to the environment.

Response Retrieval. A stimulus item is presented. This stimulus input code is sorted in the discrimination net to retrieve the image list, in which the cue is found. The cue is sorted in the net to retrieve another image list containing the proposed response image. If there is no cue, or if on either sorting pass an empty image list is selected, no response is made.

Response Generation. For purposes of response generation, there is a fixed discrimination net (decoding net), assumed already learned, which transforms letter codes of internal images into output form. The response image is decoded letter by letter by the net interpreter in the decoding net for letters.

THE ORGANIZATION OF THE LEARNING TASK

The learning of nonsense symbols by the processes heretofore described takes time. EPAM is a serial machine. Therefore, the individual items must be dealt with in some sequence. This sequence is not arbitrarily prescribed. It is the result of higher order executive processes whose function is to control EPAM's focus of attention. These *macroprocesses,* as they are called, will not be described or discussed here. A full exposition of them is available in a paper by Feigenbaum and Simon (1962).

Stating the Model Precisely: Computer Program for EPAM

The EPAM model has been realized as a program in Information Processing Language V (Newell et al., 1961*e*) and is currently being run both on the Berkeley 7090 and the RAND 7090. Descriptive information on the computer realization, and also the complete IPL-V program and data structures for EPAM (as it stood in October, 1959) are given in an earlier work by the author (1959).

IPL-V, a list processing language, was well suited as a language for the EPAM model for these key reasons:

a. The IPL-V basic processes deal explicitly and directly with list *structures.* The various information structures in EPAM (e.g., discrimina-

tion net, image list) are handled most easily as list structures. Indeed, the discrimination is, virtually by definition, a list structure of a simple type.

b. It is useful in some places, and necessary in others, to store with some symbols information descriptive of these symbols. IPL-V's description list and description list processes are a good answer to this need.

c. The facility with which hierarchies of subroutine control can be written in IPL-V makes easy and uncomplicated the programming of the kind of complex control sequence which EPAM uses.

Empirical Explorations with EPAM

The procedure for exploring the behavior of EPAM is straightforward. We have written an "Experimenter" program and we give to this program the particular conditions of that experiment as input at the beginning of an experiment. The Experimenter routine then puts EPAM *qua* subject through its paces in that particular experiment. The complete record of stimuli presented and responses made is printed out, as is the final net. Any other information about the processing or the state of the EPAM memory can also be printed out.

A number of simulations of the basic paired-associate and serial-anticipation experiments have been run. Simulations of other classical experiments in the rote learning of nonsense syllables have also been run. The complete results of these simulation experiments and a comparison between EPAM's behavior and the reported behavior of human subjects will be the subject of a later report. However, some brief examples here will give an indication of results expected and met.

A. STIMULUS AND RESPONSE GENERALIZATION

These are psychological terms used to describe the following phenomenon. If X and X′ are similar stimuli, and Y is the correct response to the presentation of X; then if Y is given in response to the presentation of X′, this is called stimulus generalization. Likewise, if Y and Y′ are similar responses, and Y′ is given in response to the presentation of X, this is called response generalization. Generalization is common to the behavior of all subjects, and is found in the behavior of EPAM. It is a consequence of the responding process and the structure of the discrimination net. For those "stimuli" are similar in the EPAM memory whose input codes are sorted to the same terminal; and one "response" is similar to another if the one is stored in the same local area of the net as the other (and hence response error may occur when response cue information is insufficient).

B. OSCILLATION AND RETROACTIVE INHIBITION

We have described these phenomena in an earlier section.

Oscillation and retroactive inhibition appear in EPAM's behavior as

consequences of simple mechanisms for discrimination, discrimination learning, and association. They were in no sense "designed into" the behavior. The appearance of rather complex phenomena such as these gives one a little more confidence in the credibility of the basic assumptions of the model.

These two phenomena are discussed together here because in EPAM they have the same origin. As items are learned over time, the discrimination net grows to encompass the new alternatives. Growing the net means adding new tests, which in turn means that more information will be examined in all objects being sorted. An important class of sorted objects is the set of cues. Cue information sufficient at one moment for a firm association may be insufficient at a later moment. As described above, this may lead to response failure. The failure is caused entirely by the ordinary process of learning new items. In the case of oscillation, the new items are items within a single list being learned. In the case of retroactive inhibition, the new items are items of the second list being learned in the same discrimination net. In both cases the reason for the response failure is the same. According to this explanation, the phenomena are first cousins (a hypothesis which has not been widely considered by psychologists).

In the EPAM model, the term *interference* is no longer merely descriptive—it has a precise and operational meaning. The process by which later learning interferes with earlier learning is completely specified.

C. FORGETTING

The usual explanations of forgetting use in one way or another the simple and appealing idea that stored information is physically destroyed in the brain over time (*e.g.*, the decay of a "memory trace," or the overwriting of old information by new information, as in a computer memory). Such explanations have never dealt adequately with the commonplace observation that all of us can remember, under certain conditions, detailed and seemingly unimportant information after very long time periods have elapsed. An alternative explanation, not so easily visualized, is that forgetting occurs not because of information destruction but because learned material gets lost and inaccessible in a large and growing association network.

EPAM forgets seemingly well-learned responses. This forgetting occurs as a direct consequence of later learning by the learning processes. Furthermore, forgetting is only temporary: lost associations can be reconstructed by storing more cue information. EPAM provides a mechanism for explaining the forgetting phenomenon in the absence of any information loss. As far as we know, it is the first concrete demonstration of this type of forgetting in a learning machine.

Conclusion: A Look Ahead

Verification of an information processing theory is obtained by simulating many different experiments and by comparing in detail specific qualitative and quantitative features of real behavior with the behavior of the simulation. To date, H. A. Simon and I have run a number of simulated experiments. As we explore verbal learning further, more of these will be necessary.

We have been experimenting with a variety of "sense modes" for EPAM, corresponding to "visual" input and "written" output, "auditory" input and "oral" output, "muscular" inputs and outputs. To each mode corresponds a perceptual input coding scheme, and a discrimination net. Associations across nets, as well as the familiar associations within nets, are now possible. Internal transformations between representations in different modes are possible. Thus, EPAM can "sound" in the "mind's ear" what it "sees" in the "mind's eye," just as all of us do so easily. We have been teaching EPAM to read by association, much as one teaches a small child beginning reading. We have only begun to explore this new addition.

The EPAM model has pointed up a failure shared by all existing theories of rote learning (including the present EPAM). It is the problem of whether association takes place between symbols or between tokens of these symbols. For example, EPAM cannot learn a serial list in which the same item occurs twice. It cannot distinguish between the first and second occurrence of the item. To resolve the problem we have formulated (and are testing) processes for building, storing, and responding from chains of token associations (Feigenbaum and Simon, 1962).

PROGRAMMING A MODEL OF HUMAN CONCEPT FORMULATION

by Earl B. Hunt & Carl I. Hovland

What is a concept? Ordinarily usage is not precise. The English "the concept of force," "the concept of massive retaliation," and "concept of dogs" are all permissible. Church (1956) has offered a definition which has been accepted, implicitly, by psychologists who perform "concept learning" experiments. Church's argument is that any given symbol (or *name*) can be attached to the members of a set of objects. For any arbitrary object there exists a rule concerning the description of the object, a rule which can be used to decide whether or not the object is a member of the set of objects to which the name applies. The decision rule is the concept of the name, the set of objects is the denotation of the name.

In a typical concept learning experiment the subject is confronted with a series of stimuli which are given a particular name and another series of stimuli which are either given another particular name, or a series of different names. Thus the first set might be called "dogs" and the second either "not-dogs" or "cats, wolves, sheep, etc." Thus some routines are necessary to classify the instances to correspond to the names assigned by the experimenter. These are our ordering routines. Sometimes the various stimuli given one name have certain common characteristics, *e.g.*, all the positive instances may have three triangles. At other times there are no common relating elements, but there are common relationships, *e.g.* all the positive instances may have the same size of upper figure as lower figure, although the figures may be large, medium or small sized in each row. A machine routine may be required to describe relations between basic stimulus elements. So we must have description routines in a simulation. Finally, different types of stimulus sets may be organized differently in

terms of different types of logical connectives. Sometimes the concept involves the joint presence of two or more characteristics. Such concepts are referred to as conjunctive concepts (*e.g.*, large red figure). Other concepts involve the presence of different subsets of characteristics. These are disjunctive concepts, *e.g.*, red *or* large figures. Different ways of defining the form of an answer are provided by a set of solution routines.

The program must be capable of simulating a variety of conditions under which experiments have been performed. As illustrations of some of the variations, or manipulations, which must be simulated the following may be mentioned.

The number and complexity of the stimuli may vary from study to study. The speed of presentation of new stimuli can be altered. The instances may be left in view or the subject may be forced to rely on his memory. Different concept learning problems can be compared along such dimensions as: logical complexity of the correct answer, number of relevant vs. number of irrelevant dimensions, order of presentation of problems of different types, and presentation of information by positive or negative instances.

The subject may make a variety of responses during the experiment. Subjects may describe, verbally, their intermediate and final hypotheses concerning the characteristics of the concept. These responses may give us clues as to the nature of the individual's information processing procedures. As such, they constitute accessory measures used in our simulation studies. The time taken to develop an answer under various experimental conditions is also a useful response measure. The errors that subjects make in subsequent identifications of stimulus names can be analyzed. The more objective records are to be preferred, and our major goal is to predict these by computer simulation.

In order to develop a theoretical explanation of concept learning we have accepted the "black box" analogy. We have attempted to write a computer program which, when given as input coded representations of the stimuli, will give as output coded responses that can be used to predict the responses of a human subject. Accurate prediction of the responses, not the development of a good hypothesis developer, nor, solely, the reproduction of previously obtained protocols, is our goal. We are not concerned with the processes specific to the task of categorizing geometric patterns. These are used as stimuli because they are convenient and because they represent stimuli which can be described in terms of previously discriminated stimuli. Hopefully we shall be able to make conclusions about concept learning processes irrespective of the particular form of the stimuli.

Reports of our psychological experimental work have been, and are being, made in separate publications. This report will be concerned with the programming details of our concept learning model. This model has

been completed, debugged, and used to simulate several experiments. After describing the model we shall indicate the result of some simulations and discuss the modifications of the model which have been indicated.

The concept learning program is a list processing language program written for the IBM 709-7090 data processing systems. The original programs were written in Information Processing Language V (IPL-V), the interpreter list processing language developed by Newell, Shaw, and Simon (Newell, 1961*e*). Partly for local administrative reasons, we are in the process of converting our programs to LISP, a list processing language developed by McCarthy (1960). We do not have sufficient experience with LISP to compare the two languages for our type of problem. As the basic logic of the LISP and IPL-V programs are the same, no distinction will be made between them.

Description of the Program

The program consists of two blocks of data, specified by the programmer at the beginning of each run, and five subsystems for data processing. At the beginning of a simulation the programmer specifies a sequence of problems, a set of parameters, and a set of lists. The last two represent the capabilities of the artificial subject. The problem data remains constant throughout the run, the specifications of the subject may be changed by the program.

Problems are presented by describing instances, the denotations of names (classes), and the conditions of presentation to be used. This takes the form of specification of memory requirements, number of stimuli presented at a single time, etc. All the conditions used to describe a problem are specified in the property list of the symbol naming the problem.

Each instance (*i.e.,* object to be categorized) is represented by a symbol whose property list specifies the symbol's class membership, and, by a list of pairs, the dimensions and values which constitute a formal description of the object. For instance, in our previous example, a large, red triangle would contain the following pairs on its description list: (class name-"alpha"), (size-large), (color-red), (shape-triangle). The formal description list constitutes the most molecular information about objects which is made available to the program. Higher-order, working descriptions based upon relations between elements of the formal description may be developed by the program.

Dimensions represent the manner in which objects are free to vary. We have utilized a "dimensional analysis" of objects which specifies a finite universe with a built-in structure to describe objects (cf. Hovland, 1952). Dimensions are also organized into "dimension sets," or groupings. These

groupings represent subsets of the set of all dimensions which will be considered together during recognition and answer development.

The "subject" specifications fall into two broad categories; numerical parameters and initial settings used to control the program. They will be discussed as they enter into the action of the model.

Figure 1 specifies the channels of communications between the various subsystems in the model. There are two major groups of subsystems. The first, as indicated in Fig. 1, is the recognition and memory system. Its task is to acquire information from the formal description of presented instances and to retain this information for later processing by the answer development and checking group.

By examining the property list of the problem, the program determines the conditions of presentation of stimuli. If these conditions would not require memorization by a human subject (*i.e.*, if instances are presented to the subject and left in view) the name of each instance, together with its entire formal description, is added to internal memory as the instance is presented. We do not maintain that subjects see all of a complex instance at the time it is presented. However, when the conditions of presentation are such that he can always reexamine the instance we see no reason for using a special recognition program.

If an instance is shown only once, and then removed, the subject can only store the information which he receives at the time the instance is presented. Here a special "recognition" program is needed. We have a rather primitive method for reading instances into memory in our present model. Included in the initial specification of the artificial subject is a list of dimension sets. Sets are read in the order in which they are placed on the list. During a particular problem our program reads, at every presentation of an instance, all dimension sets which have ever been read. If this provides sufficient information with which to discriminate the current in-

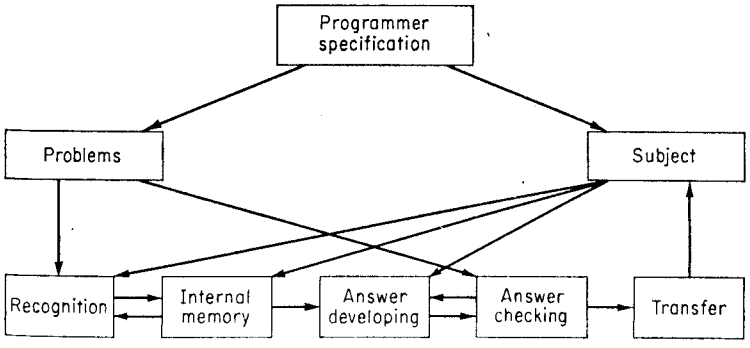


Figure 1. Program control chart.

stance from previous instances, the reading process terminates. If sufficient information is not presented, a new dimension set is read, and the discrimination test reapplied. New dimension sets are added until either all dimension sets have been used or discrimination from previously presented instances is possible. When the read program is terminated the appropriate description (some part of the formal description) is entered into internal memory.

For problems in which a requirement for memory exists, a limited occupancy model of human memory is employed (cf. Hunt, 1960*a*). The subject parameters specify a certain number of storage cells. These are set aside for representational memory of instances. Each new instance is stored, at random, in one of the cells. The previous content, if any, is lost. Thus, the probability that the artificial subject has a given instance available decreases as the number of intervening instances increases. We consider this model a crude approximation to human memory, although it has been shown to be useful in predicting the probability of utilization of information in certain cases.

Figure 1 represents the very indirect tie between recognition memory and answer developing checking units in the present system. It may be that this is not the most effective arrangement. Schemes for joining the subsystems may be considered in a later model.

The "heart" of the model is the answer development subsystem. Its internal procedure is depicted in Fig. 2. The answer developing section finds binary decision rules for distinguishing between the denotation of one name and its complement. In doing so it restricts its attention to one dimension set at a time. Dimension sets are selected in the order specified by the current description of the artificial subject. If an answer already exists which involves a particular dimension set, that set will be ignored in answer

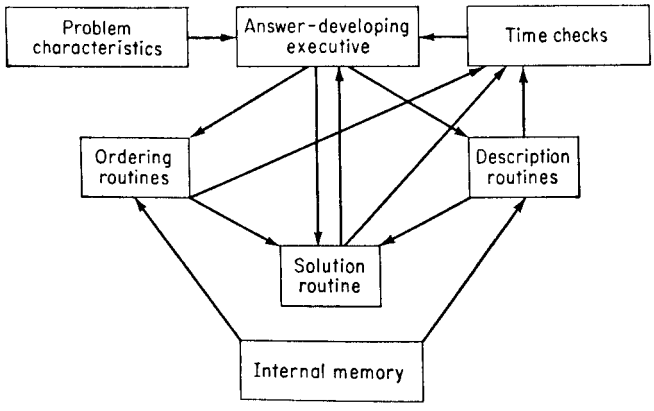


Figure 2. Answer-developing procedure.

development. The "executive routine" of the answer developing system is entered when a dimension set is found for which no answer is currently available. The plan followed by the executive routine is to prepare an *execution list* containing the names of three routines which will be executed in the order specified by the *execution list*. The contents of internal memory are used as output for the first and second of the three routines. They, in turn, provide the output for the third (last) routine of the execution list. Successful completion of the third routine results in a tentative concept definition.

The executive routine selects routines for the execution list from three reference lists. These are initially specified by the programmer as part of the subject's description list. They may be changed during execution of a simulation.

The first reference list contains the names of *ordering routines*. Each of these routines splits the instances on internal memory into two sets, working positive and working negative instances. The two categories are mutually exclusive and exhaustive of all instances in memory. In the simulations we have tried thus far three ordering routines are provided. One places in the "working positive" set all instances which are members of a class which has been indicated, by the programmer, as the class for which a concept is to be found. The second currently available routine reverses this procedure, placing the same instances in the working negative set. (If the programmer has indicated that there are several classes of equal importance the class name of the most recently presented instance is used by these two routines.) The third ordering routine defines as "working positives" all those instances which have the class name of the smallest set that is represented in internal memory, provided that there are at least two instances in the set.

Another reference list contains the name of routines which produce a working description of the instances in memory. These routines attach to each instance in internal memory a description based on a transformation of that part of the formal description included in the current dimension set. We have dealt with two description routines. One simply copies the necessary dimensions and values from the formal description to the working description. The other routine defines new dimensions based upon the relation between values of the dimensions of a formal description. The following rules are used to generate the working description:

1. A new dimension is defined for any pair of (source) dimensions whose values are numerical quantities on the same scale. For a particular instance the value of the new dimension is EQUAL, GREATER, or LESS, depending on the comparison of the values of the original pair of dimensions on that instance.

2. A new dimension is defined for any pair of source dimensions on the formal description list if, over the entire set of instances in memory, the two source dimensions share a common value. (The common value need not appear on both dimensions in the same instance.) The value of the new dimension is, for a particular instance, either SAME or DIFFERENT, depending on a comparison of the value of the original dimensions on the instance in question.

In actual programming, the ordering and description routines are applied serially. They are functionally parallel; the output of one does not affect the output of the other. They both provide output to the solution routine. This consists of all instances in internal memory, recategorized and redescribed. The solution routine attempts to define a method for discriminating between working positive and working negative instances. The discrimination is always stated as a definition of the working positive instances, even though these may be members of the complement of the class for which the program is trying to define a concept.

At present the model contains three solution routines. The first two are suited for handling conjunctive concept learning problems (problems in which the answer can be stated using only the logical connective *and*). The third is a conditional procedure which is slower, more complex, and of greater generality.

The two "conjunctive" routines both, as their first operation, list those dimensions which have only one value over the entire set of working positive instances. If this list does not exist no conjunctive definition of the working positive instances exists. If the list does exist, it is handled somewhat differently by the two routines. The first conjunctive routine searches through each of the dimensions to find if one of them *never* has the same value on the negative instances as it does on all positive instances. The second routine examines all negative instances to see whether any negative instance has the entire conjunction of dimension-value pairs which are common to all positive instances. The routine returns an answer if no such instance can be found. Thus either routine, when it succeeds, defines a conjunctive concept that can be used for the instances in internal storage.

The third solution routine, the conditional routine, is a recursive function which, if slightly modified, would give the artificial subject the capability of answering any concept learning problem. As it currently stands, it provides the capability of solving disjunctive concept learning problems of limited complexity.

The conditional routine first identifies the dimension-value pair which is most frequently found on positive instances. It then generates two sublists of working positives and working negatives, all of which contain this pair. The first conjunctive routine is applied to the two sublists. If it suc-

ceeds, it returns with an answer which can be applied to any future instance, which has the appropriate dimension-value pair. If it happens that the conditional routine generates only a sublist of positive instances, the answer is the value of the single dimension being considered. If the dimension-value pair does not occur on a future instance, the class membership of this instance is indeterminate.

If an answer is not generated in this manner, or if there remain unclassified instances, the conditional routine is repeated, omitting dimension-value pairs previously considered *and* any instances which have been classified. The result of the application of the conditional and conjunctive routines constitute a second "conditional" answer. This procedure is repeated until all instances in internal memory have been classified or until all dimensions have been considered. The result is a classification rule composed of a chain of statements about simple conjunctive answers and the rules under which they apply (*e.g.,* red triangle, green circle). The chain of statements may be of any length, but each statement must contain only two dimension-value pairs. We could have removed this restriction by applying the second conjunctive rule instead of the first. We could also have permitted an *n*th-level conditional rule by applying the conditional routine, recursively, to the sublists until all instances were classified. The resulting procedure would generate a rule for all concept learning problems. It would not necessarily be the most compact statement of the correct rule. It could degenerate into a description of particular instances.

When the executive routine selects an execution list it is, in effect, applying a template for an answer to a particular problem. If the problem has an answer which involves the relevant features abstracted by the ordering and description routines, operating on a particular dimension set, and if the answer is of a particular logical type, there exists an execution list which will find it.

The manner in which our first model changes its template is also indicated in Fig. 2. Initially the dimension set is selected. The first execution list is then selected from the reference lists contained in the subject description. The first execution list always uses the routines which are at the top of each reference list. If the execution list cannot obtain an answer, the description or solution routine (alternately) is replaced until the original execution list is reconstructed. When this happens a new ordering routine is chosen. The alternation of description and solution routines is repeated until, again, an execution list is repeated. At this point a new ordering routine is selected. When there are no more ordering routines the dimension set is replaced, using the next dimension set on the subject's list of order of noticing dimension sets. The process ends whenever either, an answer is developed, all dimension sets are examined, *or,* when the allotted time is exceeded. How this is instrumented will be described presently.

During a particular problem the order of dimension sets remains constant. However, during the time when an answer is being developed, the reference lists for description and solution routines may be temporarily altered. This is done by moving a symbol from first to last place on its reference list whenever it is removed from an execution list. One of the ways in which we can simulate individual differences is to change the initial order of routines on the reference lists.

As we have indicated, there is a "time-checking" mechanism which may interrupt the answer development process. Associated with each routine on a reference list is an index number. These numbers are specified by the programmer as part of the initial data. The programmer also specifies, as part of the problem data, a number which represents the time that the artificial subject has to develop an answer. Depending on the presentation conditions, this may represent the time he is permitted to spend on the entire problem or the time between stimulus presentations. Every time a routine on an execution list is applied, its index number is subtracted from a time signal which was, originally, set equal to the allowable time number. When the time signal reaches zero, answer developing is halted (possibly with the reference lists for description and solution rearranged) and control is returned from the executive solution routine to a higher level.

The index number associated with each routine can be thought of as an "effort" number, the cost of a particular information processing routine to the subject. Success in any problem depends on a complex interaction between the rules for rearrangement of order of routines on reference lists, the value of the index number, and the value of the allowable time number. One of our more fascinating research tasks is the unraveling of this relation.

The model, as presently programmed, has an independent check on time. Whenever a new instance is presented it is examined to see if its class membership agrees with that predicted for it by currently active answers. If the new instance does not agree, or (in the case of conditional answers) if no class membership is predicted for that instance, the answer development routine will be entered. If correct prediction occurs the answer development section is entered *only* if a "slow" rate of stimulus presentation is specified in the problem description.

Whenever an answer is developed the dimension set and execution list used are stored on its description list. When a problem is solved (*i.e.* after all instances have been presented), those dimension sets which have been associated with an answer, and those routines which have appeared on successful lists, are moved to the head of their respective reference lists. Thus, the characteristics of the subject which were originally specified by the programmer have been modified by the program.

The transfer procedure has an interesting psychological implication. Our artificial subject shows positive or negative transfer only when the

preceding problem is solved. Also, transfer is almost entirely dependent upon the form of the immediately preceding problem. We do not know whether or not this is true of human problem-solving.

Simulations and Evaluations

The model was not conceived *in vacuo*. Previous, unprogrammed models (Hovland and Hunt, 1960) had been considered for some time. In addition, we gathered protocols from Yale undergraduates who attempted to solve a "concept learning" problem which had three logically correct answers; a disjunction, a conjunction, and a relation. (This problem has been described previously (Hunt and Hovland, 1960) and some data on its difficulty was available.) All three conditions of presentation were given to each subject. The model we have just presented gave the best over-all "postdiction" of response of any model we could devise. In fitting it we altered the order and identity of symbols on reference lists, but otherwise kept the model constant. Since each subject solved three problems, we were able to make some tests of our transfer procedures and thus do not rely too heavily upon prespecified orders. The results of our match were generally encouraging. However, they cannot be taken as validating evidence since the protocols were used to develop the program.

Some more encouraging evidence came when the artificial subject attempted a series of problems used by Shepard, Hovland and Jenkins (1961). This was a completely separate study. Human subjects were asked to find categorizing rules for each of the six possible types of splits of eight instances, each describable by one of two values on three dimensions, into two sets of four each. Human subjects could solve, quite rapidly, a problem in which all relevant information could be derived from a single dimension. So could our artificial subject. Both human and artificial intelligence found a problem consisting of a "string" of two conditional statements (*e.g.* big and red, or small and white) easy. In a third case, humans and the artificial subject were unable to develop a workable rule for the authors' "Type VI" classification, in which the answer requires either description of each instance or a rather subtle rule about alternation of values. Humans did better than the artificial subject in one situation. When the correct answer could be stated as a simple rule with one exception, our program finds the problem difficult. Humans find it hard, but not nearly as hard as the "Type VI" problem. The results of this simulation, and particularly the discrepancy just mentioned, forced us to consider alternate recursions in the conditional solution routine.

A somewhat similar, unpublished, experiment was performed by Hunt and H. H. Wells. Here the five commonly used logical connectives between two elements provided the answer. A "Truth table" was constructed and

presented to subjects in geometric form. For example, the connective "p *and* q" might be represented by "red and star." The five problems were presented in five orders, each subject solving all five problems in one of the orders. Simulation and analysis of this experiment has not been completed at the time of this writing, however, we have some preliminary results. There is good general agreement between our simulation routines and *some* protocols. Both the computer model and the subjects are sensitive to the order in which problems are presented, but their reactions are not as similar as we would like. A new transfer procedure is needed. In an experiment which is not directly related to simulation, Wells is studying the manner in which human subjects learn methods of solution for disjunctive problems. We hope that his experiments will provide some clues about the nature of the transfer procedures we should include in our model.

We do not claim to have presented a complete explanation of concept learning! Certainly others will agree with us. In programming the model we made many decisions with little theoretical or empirical justification. Some of these are certain to be wrong. But which ones?

We shall probably have to change our routines for memory and recognition. Some of the known phenomena of memory cannot be reproduced by a simple occupancy model. For instance, the effect of stimulus similarity upon memory cannot be represented. Our model has an all-or-none aspect in its interference features. An intervening instance either completely eliminates the record of a previous instance or does not affect it at all. This does not seem to be the final answer to the problem of memory in concept learning.

Two alternative memory systems are being considered. One system retains and extends the limited occupancy model. Instead of storing one "code word" (actually, a list structure), representing all known information about an instance, on a single occupancy list, several code words would be stored in several occupancy lists. Each of these code words would represent a particular type of information about some part of the instance in question. Storage of each code word would be independent on each occupancy list. Code words referring to the same instance would reference each other's locations. When information from memory was required a picture of each instance would be reconstructed from the cross-referencing system. However, since intervening instances would be storing instances independently on each occupancy list, some of the code words might be replaced. The extent of this replacement would depend upon the similarity between the instance to be recalled and the stimuli which followed its presentation. This system would be sensitive to stimulus similarity effects.

Alternately, we could use an associationist memory system. Instead of trying to remember units of information directly we would build "associa-

tions" between names and stimulus features. This is the logic of the technique used by many learning theorists in psychology. Machinery to realize such a memory has been extensively investigated by Rosenblatt (1958, 1959). There is also some similarity between this approach and the classification mechanisms based upon Selfridge's "Pandemonium" scheme (Selfridge and Neisser, 1960). To adopt such a memory system would require changing the entire logic of our model. Association schemes generally contain, in themselves, a mechanism for concept learning. It also seems that they require some sort of gradient of generalization. Recent experiments (Shepard and Chang, 1961*a;* Shepard et al., 1961*b*) indicate that, in concept learning, the tendency to code stimuli symbolically plays a greater role than generalization based upon stimulus similarity. For these reasons we have, tentatively, rejected an associationist memory mechanism.

In the present model we subject the formal description of an instance to two transformations. When an instance is presented, the dimensions of the formal description are sampled to determine what information is to be placed in memory. At some later time, that part of the formal description which is in memory is retransformed to provide a working description. The two procedures could be combined if the description routine currently at the head of the description routine reference list were to be applied directly to an instance before it entered memory.

Such a procedure would have advantages in saving storage space. Instead of having to have two separate locations, for working and permanent description, in the internal memory, only one description need be stored. But we pay for saving this space by losing information. By definition, any working description can be derived from the formal description. All working descriptions cannot be derived from each other. For instance, if we know that an instance contained two figures of the same color, we do not know what that color is. As a result, our artificial subject's ability to utilize a particular description routine at time t would depend very much upon the description routines used previously.

The role of "set" at time of presentation as a determinant of later memory characteristics needs more extensive investigation. Some experiments (Lawrence and Coles, 1954; Lawrence and LaBerge, 1956), suggest that "set" is a function of how memory is searched rather than how items enter into memory. Also, there exists a rather contradictory literature on "latent learning," a term used to describe experiments in which animals, trained to respond to cue A in the presence of cue B, which is irrelevant to the animal's current need, learn more rapidly a later response to cue B. From present experimental results it is not obvious how stimulus recognition and answer development procedures should be connected in a concept learning simulation.

Procedures for representing transfer may not be represented adequately

in the present model. Transfer is defined as the effect of previous problem-solving experience upon solution of the problem with which the subject is faced at the time of the test. We decided to work first with a simple method of representing transfer, in which the subject tries whatever worked last time. A principal result of the simulation of the Hunt and Wells work on logical connectives has been a demonstration that a new transfer procedure is needed.

In the tradition of classical learning theory, we could attach a modifiable numerical index to each routine on a reference list. This index could be used to determine the probability that a routine would be selected. This method of representing learning is probably the most common. The principal objection to it is that it implies the existence of "counters in the head" and, essentially, makes our program a digital simulation of an analog system.

The alternative to association indices is a new method of ordinal rearrangement of routines on a reference list. The problem with ordinary rearrangements is that they did not permit us to specify a variable distance between routines on a list. Suppose we consider each concept learning problem as a contest between routines on the same reference list. The one that finds a place on a successful execution list is victorious. How many times must the routine in position n "win" before it gains the next highest position? Should it jump positions? As we have indicated, some research relative to this topic is being conducted.

Conceivably, we may have to change our entire method of transfer. At present our model records answers, with associated information about useful routines. We could attach to routines information about problems on which they had been useful. We would then have to develop some way for the artificial subject to extract, rapidly, key features of a problem while the answer is being developed. Routines would be examined to see what, in the light of past experience, was their probable usefulness on this sort of problem.

Closely related to the problem of transfer is the problem of response selection during learning. Our present model rearranges its order of response selection after a problem is solved. During a problem, response selection is controlled by time parameters which are independent of program control. No use is made of intermediate computations in selecting the next item to be placed on an execution list. In an alternate model this might be the controlling factor. The means-end analysis of the Logic Theorist (Newell and Shaw, 1957*b*) uses intermediate calculations heavily. Amarel (1960) has proposed a computer model for an area very similar to ours in which intermediate computations exert control on answer development.

Our simulation work, and analysis of experimental data, has convinced

us that some method of making the choice of one item on an execution list dependent upon the product of execution of previously selected routines is desirable. What is not clear is the optimum amount of dependency. Bartlett (1958) has presented an analog, in an entirely different context, which may clarify the problem. He compared problem-solving and thinking to motor skills responses, such as serving in tennis. There are certain points at which a chain of responses can be altered; in between these points a series of acts will be executed without interruption. Our problem, experimentally, is to identify the responses and choice points.

We feel that the principal use of our model, so far, has not been in the generating of an explanation of concept learning so much as it has been in indicating the type of new experimental data needed. We have had to be very specific in our thoughts as we programmed this model. As a result, we have reached some conclusions about the kind of experiments that need to be done. It may well be that the typical concept learning experiment confuses three processes; memory, recognition, and symbolic problem-solving. It is not clear whether or not these should be treated as part of a unitary "concept learning" act. They can be programmed separately. In addition we have become concerned with questions of transfer, the effect of the subject's current hypothesis upon his later retention of information, and the effect of time pressure upon information processing. A real awareness of these problems has been a major outcome of programming a concept learning model.

Comparisons with Related Work

Viewed formally, our problem is closely related to models of pattern recognition. Programming either a pattern recognizer or a concept learner involves the development of a mechanism which operates on a specified stimulus universe to map stimuli from predetermined subsets into particular responses. Because of this mathematical identity, at least one critic (Keller, 1961) has suggested that problems of this sort should be treated together, without "psychologizing" or "neurologizing." While this may be useful in developing theorems about a canonical form of categorization, it may not be an appropriate strategy for simulation studies. In particular, our approach is quite different from that of the pattern recognition studies with which we are familiar.

The most striking difference is in the manner in which we precode the stimuli. Pattern recognizers usually accept stimuli coded into projections on a grid. The result is a string of bits, each bit representing the presence or absence of illumination of some part of the grid. The same representation could be used for a temporal pattern. Each bit would stand for the presence or absence of some stimulus feature.

We presuppose the existence of a dimension and value coding (Hovland, 1952) and deal with perceptual aspects which are readily verbalizable. A pattern recognizer develops its own code. Any coding scheme developed by a pattern recognizer will be specific to the stimuli used (visual vs. auditory, etc.). Since we are interested in the manipulation of coded elements we avoid this problem by *fiat* in our programming and by explicit instructions to our subjects in our experimental work.

Our model is also different from most pattern recognizers in the processes it uses. Pattern recognizers, at least as developed by Selfridge and his co-workers (Selfridge and Neisser 1958, 1959), and by Rosenblatt (1960), are basically parallel processing devices which utilize a large number of redundant, error-prone tests. Our program is a serial processor which tries to develop a single, perhaps complex, error-free classification test. We do not see any incompatibility in the two approaches. Our program deals with the simulation of a symbolic process. That the two problems are formally similar does not mean that they are realized in the same way by problem-solvers.

In principle, there would be no objection to utilizing a pattern recognizer to provide the input to the concept learner. The combined system could develop its own dimensions and values and then operate upon them. In practice, such a scheme is undoubtedly premature. But it is a long-range goal.

The concept learning problem has been attacked directly in two previously mentioned studies, by Kochen (1961*a*) and Amarel (1960). Kochen restricted his program to solution of "concepts" based upon a conjunctive rule involving stimuli specified by strings of bits. His program consisted of executing algorithms upon the information about the universe of objects which was available *at any one time,* in memory. The program also contained features for making random guesses about the correct concept. These guesses could be weighed for "confidence," using an index which satisfied Polya's (1954) postulates for plausible reasoning. One of Kochen's (1954) findings, based on Monte Carlo runs of his system, was that changes in the value of the confidence index could be used to estimate the probability that an answer was correct before a proof of the answer was available.

Amarel (1960) proposed a machine that could generate routines to map arguments to values in symbolic logic. The key feature of his proposal, one we might well adopt, is his use of intermediate results to "monitor" future answer development.

Neither Kochen nor Amarel were directly concerned with simulation of human performance. This difference in goals, and features of programming, are the major differences between our work and theirs.

Superficially, our program is similar to the list processing programs

written by the Carnegie Institute of Technology–RAND Corporation group headed by Newell, Shaw, and Simon, and McCarthy (McCarthy, 1960) and his associates at MIT. In particular, the work of Feigenbaum (1959), at RAND, is related to ours. He developed a program to simulate paired-associates learning. As part of his program he included a routine for selective recognition of stimulus features. As more experience with the stimulus universe was provided, more features were read into the system to enable it to make finer discriminations. The logic of Feigenbaum's recognizing system, and in particular its capability for dropping stimulus features which are not useful in discrimination, could be incorporated into our program.

Our present program, although running now, is in no sense complete. Almost every new simulation has indicated ways in which it could be improved. We intend to continue to investigate concept learning by use of an information processing model. But we do wish to add a word of caution. Neither our model, nor any other, has generated a large number of new experiments. This is a traditional test of the utility of a scientific model, and it is going to have to be met by us and by others interested in this field. We do not feel that the utility of computer programming models in psychology has been proven or disproven. The jury is still out. We, of course, hope that a favorable verdict will be returned.

section 3

Decision-making under Uncertainty

A large number of decisions are made under conditions of uncertainty, *i.e.*, where the decision-maker does not know the consequences of his alternatives. The commander sending his troops into battle is faced with such a decision problem. The book publisher deciding how many copies of a book to print is in a similar situation. Economists, mathematicians, and statisticians have studied how people "should" behave in these situations, while behavioral scientists have studied how people do behave in these situations. The two articles in this section are in the latter category. In both of them, the computer is used in the construction of information processing models of the behavior of individual decision-makers in uncertain situations.

Feldman reports a study of behavior in the binary choice experiment. In this experiment the subject is asked to predict which of two events will occur on each of a series of trials. With the aid of data obtained from students "thinking aloud" in the binary choice experiment, Feldman has been able to construct models of the cognitive processes underlying binary choice behavior. These models represent the hypothesis-testing behavior that subjects exhibit in these experiments. One of these models and the associated protocol are presented in the following article.

Clarkson has chosen to study human behavior in portfolio selection. He has studied how an investment officer in a bank selects a portfolio of stocks for a trust fund, given the legal constraints involved, the goals of the trust, and the conditions of the market.

Clarkson's analysis of the behavior of this decision-maker has enabled him to construct a computer program which, in test cases, has been able to make very accurate predictions of the trust officer's behavior. In four test cases, the model selects 29 stocks—the same number as the trust officer—and correctly predicts the number of different stocks in each portfolio. The model does very well on the number of shares of each stock, too. Of the 29 selections, there are only five cases in which the model selected a stock different from the stock selected by the trust officer. These results certainly indicate that Clarkson's model is an excellent predictor of the trust officer's behavior.

The implications of both these models extend beyond the particular decision situations in which they were developed. Feldman's work indicates that what appears to be a simple problem is treated by the subject as a very complex situation. The hypothesis-testing procedure which the subject uses is quite similar to the behavior of the researcher studying the subject. The subject is looking for patterns in the event series, and the experimenter is looking for patterns in the subject's behavior. Both are trying to induce the regularity which they both believe does exist. Clarkson's model furnishes important support for a problem-solving model of human decision-making that depicts a decision-maker of limited rationality using a limited memory and a rather small set of rules of thumb. Neither the subject in the binary choice experiment nor the trust investment officer are following the accepted strategies for "rational" behavior in their environments. Detailed analyses such as Feldman and Clarkson have done provide useful information on how people behave in particular decision situations and suggest explanations of behavior in other situations.

Julian Feldman is a member of the faculty of the School of Business Administration, University of California, Berkeley.

Geoffrey Clarkson is a member of the faculty of the School of Industrial Management, Massachusetts Institute of Technology.

SIMULATION OF BEHAVIOR IN THE BINARY CHOICE EXPERIMENT

by Julian Feldman

Introduction

Modern, high-speed digital computers have been used to simulate large, complex systems in order to facilitate the study of these systems. One of these systems that has been studied with the aid of computer simulation is man. The present report describes another addition to the growing list of efforts to study human thinking processes by simulating these processes on a computer. The research summarized here has been concerned with simulating the behavior of individual subjects in the binary choice experiment (Feldman, 1959). The first section contains a description of the experiment. An overview of the model is given in the second section. The model for a particular subject is described in some detail in the third section.

The Binary Choice Experiment

In the binary choice experiment, the subject is asked to predict which of two events, E_1 or E_2, will occur on each of a series of trials. After the subject makes a prediction, he is told which event actually occurred. The sequence of events is usually determined by some random mechanism, e.g., a table of random numbers. One and only one event occurs on each trial. The events may be flashes of light or symbols on a deck of cards. The subject is usually asked to make as many correct predictions as he can.

In the research reported here, the experiment described in the preceding paragraph was modified by asking the subject to "think aloud"—to give his reasons for making a prediction as well as the prediction itself. The

subject's remarks were recorded. The subject was instructed to "think aloud" in order to obtain more information on the processing that the subject was doing. This technique has been used in some of the classical investigations of problem-solving behavior (Duncker, 1945; Heidbreder, 1924) and in other computer simulation studies of thinking (Clarkson and Simon, 1960; Newell and Simon, 1959*d*). A comparison of the behavior of subjects in the binary choice experiment who did "think aloud" with the behavior of subjects who did not "think aloud" did not reveal any major differences (Feldman, 1959). The events in the present experiment were the symbols "plus" and "check." "Check" occurred on 142 of 200 trials and "plus" on the remaining 58 trials. The symbols were recorded on a numbered deck of 3-inch $\times$ 5-inch cards. After the subject made his prediction for trial t, he was shown card t which contained a "plus" or "check." While the subject was predicting the event of trial t, he could only see the event of trial t-1. A transcription of the tape recording of the remarks of subject DH and the experimenter, the author, in an hour-long binary choice experiment is presented in the Appendix. In the Appendix and the rest of this report, the symbols "plus" and "check" are represented by "P" and "C" respectively. The transcription will be referred to as a protocol.

The Basic Model

To simulate the behavior of an individual subject in the binary choice experiment, a model of the subject's behavior must be formulated as a computer program. If the program is then allowed to predict the same event series as the subject has predicted, the behavior of the program—the predictions and the reasons—can be compared to the behavior of the subject. If the program's behavior is a reasonable facsimile of the subject's behavior, the program is at least a sufficient explanation of the subject's behavior. The level of explanation is really determined by the subject's statements. No attempt is made to go beyond these to more basic processes, *e.g.,* neurological or chemical, of human behavior. Thus, the model is an attempt to specify the relationship between the reasons or hypotheses that the subject offers for his predictions and the preceding hypotheses, predictions, and events. The subject is depicted as actively proposing hypotheses about the structure of the event series. These hypotheses are tested by using them to predict events. If the prediction of the event is correct, the hypothesis is usually retained. If the prediction of the event is wrong, a new hypothesis is generally proposed.

The Model for DH

The model for each subject is based on a detailed examination of the protocol and some conjectures about human behavior. Perhaps the best

thing to do at this point is to describe in some detail a model for the subject, DH, whose protocol appears in the Appendix.

The Hypotheses

This model proposes two types of hypotheses about the event series. The first type of hypothesis is a pattern of events. The model has a repertoire of nine patterns:

progression of C's
progression of P's
single alternation
2 C's and 1 P
1 C and 2 P's
2 P's and 2 C's
3 P's and 3 C's
4 P's and 4 C's
4 P's and 3 C's

The model can propose that the event series is behaving according to one of these patterns and use the pattern hypothesis to predict the event of a given trial, t. The predictions of the first two patterns—progression of C's and progression of P's—for trial t are independent of the events preceding trial t. The predictions of the other patterns (the alternation patterns) are dependent on these preceding events. Thus, if the subject proposes the pattern "single alternation" for trial t and the event of trial t − 1 was a C, the prediction for trial t is a P. In order to facilitate the determination of the prediction of an alternation pattern for trial t, the patterns are coded as sorting nets. For example, the pattern "2 C's and 1 P" is represented in the following fashion:

Is event t − 1 a C?
  No—Predict C for trial t.
  Yes—Is event t − 2 a C?
    No—Predict C for trial t.
    Yes—Predict P for trial t.

The second type of hypothesis that the model can propose is an anti-pattern or guess-opposite hypothesis. For example, the model can propose that the event of trial t will be the opposite of that predicted by a given pattern. This type of hypothesis is the model's representation of the notion of "gambler's fallacy"—the reason people predict "tails" after a coin falls "heads" seven times in a row.

The most general form of hypothesis has two components: a pattern component and a guess-opposite component. The prediction of the hypothesis is obtained by finding the prediction of the pattern component. If the hypothesis has a guess-opposite component, then the prediction of the

hypothesis is the opposite of the pattern prediction. If the hypothesis does not have a guess-opposite component, then the prediction of the hypothesis is the prediction of the pattern component. Thus, while the prediction of the pattern hypothesis "progression of C's" is always a C, the prediction of the hypothesis "guess-opposite-progression-of-C's" is always a P.

The Basic Cycle

The basic cycle of the model is as follows: The model uses an hypothesis to predict the event of trial t. The event is then presented. The model in Phase One "explains" the event of trial t with an explanation hypothesis. In Phase Two a prediction hypothesis for trial $t + 1$ is formed. The model uses this prediction hypothesis to predict trial $t + 1$. The event of trial $t + 1$ is presented, and the cycle continues.

Phase One

The basic motivation for this phase of the model is that the model must "explain" each event. An acceptable explanation is an hypothesis that could have predicted the event. The processing of Phase One is represented in the flow chart of Fig. 1.

The processing to determine the explanation hypothesis for trial t begins by testing whether the pattern component of the prediction hypothesis for trial t could have predicted the event of trial t correctly. If the pattern component could have predicted correctly, the pattern component is the explanation hypothesis. If the pattern component could not have predicted correctly, the pattern-change mechanism is evoked. Thus if the prediction hypothesis for trial t contained only a pattern component and the hypothesis predicted correctly, the explanation hypothesis for trial t is the prediction hypothesis for trial t. If the prediction hypothesis for trial t contained a guess-opposite component and the hypothesis predicted correctly, the pattern-change mechanism is evoked because the pattern component could not have predicted the event correctly by itself. If the

A. COULD THE PATTERN COMPONENT OF THE PREDICTION-HYPOTHESIS FOR TRIAL T HAVE PREDICTED THE EVENT OF TRIAL T CORRECTLY?
  B. YES—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THE PATTERN COMPONENT OF THE PREDICTION-HYPOTHESIS FOR TRIAL T.
  C. NO—EVOKE PATTERNS THAT COULD HAVE PREDICTED THE EVENTS OF TRIALS T AND T-1 CORRECTLY. THE PATTERN OF THE PREDICTION-HYPOTHESIS FOR TRIAL T IS EVOKED IF IT COULD HAVE PREDICTED CORRECTLY THE EVENTS OF TRIAL T-1, T-2, AND T-3.
  D. SELECT FROM THE SET OF EVOKED PATTERNS THAT PATTERN THAT HAS BEEN SELECTED MOST OFTEN ON PRECEDING TRIALS.
  E. IS THE SELECTED PATTERN THE PATTERN COMPONENT OF THE PREDICTION-HYPOTHESIS FOR TRIAL T?
    F. YES—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THROW ME OFF THE SELECTED PATTERN.
    G. NO—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THE SELECTED PATTERN.

Figure 1. Phase One of binary choice model for DH.

prediction hypothesis for trial t was a guess-opposite hypothesis and it predicted incorrectly, the pattern component of the prediction-hypothesis becomes the explanation hypothesis for trial t. The motivation here is really quite simple although the explanation may sound involved. If, in this binary situation, the hypothesis that a pattern will change leads to an incorrect prediction, the pattern must have persisted; and the pattern is an acceptable explanation of the event. If the hypothesis that a pattern will change leads to a correct prediction, the pattern obviously did not persist; and the possibility of a new pattern is considered.

The pattern-change mechanism is evoked on trial t if the pattern component of the prediction hypothesis for trial t is unable to predict the event of trial t. The pattern-change mechanism consists of two parts. The first part evokes a subset of the nine patterns listed above. The second part of the pattern-change mechanism selects a single pattern out of the evoked set. A pattern is evoked, *i.e.,* considered as a possible explanation of the event of trial t, if the pattern can predict the events of trials t and t — 1. The pattern of the prediction hypothesis for trial t, *i.e.,* the pattern that cannot predict event t, is included in the evoked set if it can predict events t — 1, t — 2, and t — 3. Of the patterns that are evoked, the pattern that has been selected most often on prior trials is selected as the pattern component of the explanation hypothesis. If the pattern component of the prediction hypothesis for trial t is selected, then the explanation hypothesis is an antipattern hypothesis which is the model's interpretation of the subject's hypothesis "you have thrown me off the pattern" (cf. trial 9 of the protocol in the Appendix). The model interprets event t as an attempt to "throw me off" when the following three conditions are met: (1) the pattern is unable to predict the event of trial t; (2) the pattern is able to predict at least the three consecutive events of trials t — 1, t — 2, and t — 3; and (3) the pattern is also the most frequently selected of those patterns that are evoked.

Phase Two

While Phase One is concerned mainly with the processing of the pattern component of the hypothesis, Phase Two is concerned with the processing of the guess-opposite component. Phase Two is represented in the flow chart of Fig. 2.

If the prediction hypothesis for trial t contained a guess-opposite component, the guess-opposite component is processed in a fashion quite analogous to the processing of the pattern component in Phase One. If the antipattern prediction hypothesis for trial t predicted the event of trial t correctly, the guess-opposite component is retained, and the prediction hypothesis for trial t + 1 is guess-opposite-the-pattern-of-the-explanation-hypothesis. If the antipattern prediction hypothesis for trial t predicted

H. DID THE PREDICTION-HYPOTHESIS FOR TRIAL T CONTAIN A GUESS-OPPOSITE COMPONENT?
  I. YES—DID THE PREDICTION-HYPOTHESIS FOR TRIAL T PREDICT THE EVENT OF TRIAL T CORRECTLY?
    J. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-OPPOSITE THE PATTERN COMPONENT OF THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
    K. NO—DID THE PREDICTION HYPOTHESIS FOR TRIALS T-1 AND T-2 CONTAIN GUESS-OPPOSITE COMPONENTS AND WERE THE PREDICTIONS OF THE EVENTS OF THESE TRIALS CORRECT?
      L. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-OPPOSITE THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
      M. NO—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
  N. WILL THE EXPLANATION-HYPOTHESIS FOR TRIAL T CONTINUE? (SEE TEXT FOR AN EXPLANATION OF THIS TEST.)
    O. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
    P. NO—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-OPPOSITE THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
Q. PREDICT EVENT FOR TRIAL T+1.

Figure 2. Phase Two of binary choice model for DH.

the event of trial t incorrectly, the guess-opposite component is considered for retention in a fashion analogous to the "throw-me-off" consideration for patterns. If the prediction hypotheses for trial $t-1$ and $t-2$ had guess-opposite components and these hypotheses predicted correctly, then the guess-opposite component is retained for the prediction hypothesis of trial $t+1$. If these conditions are not fulfilled, the guess-opposite component is dropped; and the prediction hypothesis for trial $t+1$ is the explanation hypothesis for trial t.

If the prediction hypothesis for trial t did not contain a guess-opposite component, the model considers whether or not the guess-opposite component should be introduced on trial $t+1$. The model makes this decision on the basis of its past experience. It determines the number of consecutive events including and preceding the event of trial t that can be predicted by the explanation hypothesis for trial t. This number will be called N_1. Then the model searches its memory backward from the last trial included in N_1 to find a trial for which the explanation hypothesis was the same as the explanation hypothesis for trial t. Then the model determines the number of contiguous events including, preceding, and following this prior occurrence of the explanation hypothesis of trial t that can be predicted by this hypothesis. This number will be called N_2. If $N_2 = N_1$, the model decides that the explanation hypothesis for trial t will not be the prediction-hypothesis for trial $t+1$. The prediction hypothesis for trial $t+1$ becomes guess-opposite-the-explanation-hypothesis for trial t. If $N_2 > N_1$, the model decides that the explanation hypothesis for trial t will be the prediction hypothesis for trial $t+1$. If $N_1 > N_2$, the model decides that this prior occurrence of the explanation hypothesis for trial t is really not pertinent and continues to search its memory for an occurrence of the explanation hypothesis where $N_2 \geqq N_1$. If no such occurrence can be

found, the prediction hypothesis for trial t + 1 is the explanation hypothesis for trial t.

Predicting with the Models

Models of individual behavior like the one described for DH can be used to predict the same series of binary events that the subject was asked to predict. The predictions and hypotheses of the model—the model's protocol—can then be compared to the subject's protocol. The model does not speak idiomatic English, and so the comparison is made between the machine's protocol and a suitably coded version of the subject's protocol.

```
A. COULD THE PATTERN COMPONENT OF THE PREDICTION-HYPOTHESIS
   FOR TRIAL T HAVE PREDICTED THE EVENT OF TRIAL T CORRECTLY?
   B. YES—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THE PATTERN
      COMPONENT OF THE PREDICTION-HYPOTHESIS FOR TRIAL T.
            * 1. DID SUBJECT'S EXPLANATION-HYPOTHESIS
            *    FOR TRIAL T CONTAIN PATTERN COMPONENT
            *    OF THE PREDICTION-HYPOTHESIS FOR TRIAL T?     120
            *    YES—GO TO 6.                                  117
            *    NO—ERROR—FAILURE TO EVOKE PATTERN-
            *    CHANGE MECHANISM. GO TO C.                      3
   C. NO—EVOKE PATTERNS THAT COULD HAVE PREDICTED THE EVENTS
      OF TRIALS T AND T-1 CORRECTLY. THE PATTERN OF THE
      PREDICTION-HYPOTHESIS FOR TRIAL T IS EVOKED IF IT COULD
      HAVE PREDICTED CORRECTLY THE EVENTS OF TRIALS T-1, T-2,
      AND T-3.
            * 2. WAS THE PATTERN OF THE SUBJECT'S EXPLANATION-
            *    HYPOTHESIS FOR TRIAL T EVOKED?                 78
            *    YES—GO TO D.                                   61
            *    NO—ERROR—FAILURE TO EVOKE PATTERN.
            *    ADD SUBJECT'S PATTERN TO EVOKED SET
                 AND CONTINUE.                                  17
   D. SELECT FROM THE SET OF EVOKED PATTERNS THAT PATTERN THAT
      HAS BEEN SELECTED MOST OFTEN ON PRECEDING TRIALS.
            * 3. WAS THE PATTERN OF THE SUBJECT'S EXPLA-
            *    NATION-HYPOTHESIS FOR TRIAL T SELECTED?        78
            *    YES—GO TO E.                                   64
            *    NO—ERROR—FAILURE TO SELECT PATTERN.
            *    REPLACE INCORRECT PATTERN WITH SUBJECT'S
            *    PATTERN AND CONTINUE.                          14
   E. IS THE SELECTED PATTERN THE PATTERN COMPONENT OF THE
      PREDICTION-HYPOTHESIS FOR TRIAL T?
      F. YES—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THROW ME OFF
         THE SELECTED PATTERN.
            * 4. DID SUBJECT'S EXPLANATION-HYPOTHESIS
            *    FOR TRIAL T CONTAIN THROW-ME-OFF?              27
            *    YES— GO TO H.                                  26
            *    NO—ERROR—INCORRECT EVOCATION OF
            *    THROW-ME-OFF. DELETE THROW-ME-OFF
            *    AND GO TO H.                                    1
      G. NO—EXPLANATION-HYPOTHESIS FOR TRIAL T IS THE
         SELECTED PATTERN.
            * 5. DID SUBJECT'S EXPLANATION-HYPOTHESIS
            *    FOR TRIAL T CONTAIN THROW-ME-OFF?              51
            *    YES—ERROR—FAILURE TO EVOKE THROW-ME-
            *    OFF. INSERT THROW-ME-OFF AND GO TO H.           3
            *    NO—GO TO H.                                    48
            * 6. DID SUBJECT'S EXPLANATION-HYPOTHESIS
            *    FOR TRIAL T CONTAIN THROW-ME-OFF?             117
            *    YES—ERROR—FAILURE TO EVOKE THROW-ME-
            *    OFF. INSERT THROW-ME-OFF AND GO TO H.           3
            *    NO—GO TO H.                                   114
```

Figure 3. Summary of behavior of Phase One of binary choice model for DH adapted for conditional prediction.

```
H. DID THE PREDICTION-HYPOTHESIS FOR TRIAL T CONTAIN A GUESS-
   OPPOSITE COMPONENT?
      I. YES—DID THE PREDICTION-HYPOTHESIS FOR TRIAL T PREDICT
         THE EVENT OF TRIAL T CORRECTLY?
         J. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-
            OPPOSITE THE PATTERN COMPONENT OF THE EXPLANATION-
            HYPOTHESIS FOR TRIAL T.
                     * 7. DID SUBJECT'S PREDICTION-HYPOTHESIS
                     *    FOR TRIAL T+1 CONTAIN GUESS-OPPOSITE
                     *    COMPONENT?                              6
                     *    YES—GO TO 12.                           5
                     *    NO—ERROR—INCORRECT RETENTION OF
                     *    GUESS-OPPOSITE COMPONENT. DELETE
                     *    GUESS-OPPOSITE AND GO TO 12.            1
         K. NO—DID THE PREDICTION HYPOTHESIS FOR TRIALS T-1 AND
            T-2 CONTAIN GUESS-OPPOSITE COMPONENTS AND WERE THE
            PREDICTIONS OF THE EVENTS OF THESE TRIALS CORRECT?
            L. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-
               OPPOSITE THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
                     * 8. DID SUBJECT'S PREDICTION-HYPOTHESIS
                     *    FOR TRIAL T+1 CONTAIN GUESS-OPPOSITE
                     *    COMPONENT?                              2
                     *    YES—GO TO 12.                           2
                     *    NO—ERROR—INCORRECT RETENTION OF
                     *    GUESS-OPPOSITE COMPONENT. DELETE
                     *    GUESS-OPPOSITE AND GO TO 12.            0
            M. NO—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS THE
               EXPLANATION-HYPOTHESIS FOR TRIAL T.
                     * 9. DID SUBJECT'S PREDICTION-HYPOTHESIS
                     *    FOR TRIAL T+1 CONTAIN GUESS-OPPOSITE
                     *    COMPONENT?                             12
                     *    YES—ERROR—FAILURE TO KEEP GUESS-
                     *    OPPOSITE COMPONENT. INSERT GUESS-
                     *    OPPOSITE AND GO TO 12.                  1
                     *    NO—GO TO 12.                           11
      N. WILL THE EXPLANATION-HYPOTHESIS FOR TRIAL T CONTINUE?
         (SEE TEXT FOR AN EXPLANATION OF THIS TEST.)
         O. YES—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS THE
            EXPLANATION-HYPOTHESIS FOR TRIAL T.
                     *10. DID SUBJECT'S PREDICTION-HYPOTHESIS
                     *    FOR TRIAL T+1 CONTAIN GUESS-OPPOSITE
                     *    COMPONENT?                            136
                     *    YES—ERROR—FAILURE TO EVOKE GUESS-
                     *    OPPOSITE COMPONENT. INSERT GUESS-
                     *    OPPOSITE AND GO TO 12.                 10
                     *    NO—GO TO 12.                          126
         P. NO—PREDICTION-HYPOTHESIS FOR TRIAL T+1 IS GUESS-
            OPPOSITE THE EXPLANATION-HYPOTHESIS FOR TRIAL T.
                     *11. DID SUBJECT'S PREDICTION-HYPOTHESIS
                     *    FOR TRIAL T+1 CONTAIN GUESS-OPPOSITE
                     *    COMPONENT?                             39
                     *    YES—GO TO 12.                           2
                     *    NO—ERROR—INCORRECT SELECTION OF
                     *    GUESS-OPPOSITE. DELETE GUESS-OPPOSITE
                     *    AND CONTINUE.                          37
                     *12. WAS PATTERN OF SUBJECT'S EXPLANATION-
                     *    HYPOTHESIS FOR TRIAL T THE SAME AS
                     *    THE PATTERN OF THE SUBJECT'S PREDICTION-
                     *    HYPOTHESIS FOR TRIAL T+1?              195
                     *    YES—GO TO Q.                           192
                     *    NO—ERROR—FAILURE TO CHANGE PATTERN.
                     *    INSERT SUBJECT'S PATTERN IN PREDICTION-
                     *    HYPOTHESIS FOR TRIAL T+1 AND CONTINUE.   3
Q. PREDICT EVENT FOR TRIAL T+1.
                     *13. DID SUBJECT PREDICT SAME EVENT?        195
                     *    YES—GO TO A.                           193
                     *    NO—ERROR—INCORRECT PREDICTION.
                     *    CORRECT AND GO TO A.                     2
```

Figure 4. Summary of behavior of Phase Two of binary choice model for DH adapted for conditional prediction.

The model's protocol can be generated by presenting the model with the events in the same way the subject was presented with the events in the binary choice experiment; or the computer can take the experimenter's role, too, if suitable precautions are taken to prevent the model from peeking. However, this straightforward method of simulating the subject's behavior raises difficulties. These difficulties are identical to those of getting a chess or checker program to play a book game (Newell, Shaw and Simon, 1959*c*; Samuel 1959*a*). Because the decision of the chess or checker program at move m depends on its decisions at the preceding moves, m — 1, m — 2, . . . , such a program, when it is playing a book game, must be "set back on the track" if its move deviates from the book move. The program and the book must have the same history if the program is to have a fair chance to make the same decision as that made in the book game. This "setting back on the track" may involve resetting a large number of parameters as well as changing the move itself. Elsewhere, I have called this "setting-back-on-the-track" technique *conditional prediction*. The prediction of the model is conditional on the preceding decisions of the model being the same as those of the subject it is trying to predict (Feldman, 1962).

The application of the conditional prediction technique to binary choice models such as the one described above for the subject DH involves (1) comparing the program's behavior and the subject's behavior at every possible point, (2) recording the differences between the behaviors, and (3) imposing the subject's decision on the model where necessary. A type of monitor system is imposed on the program to perform these functions. The model for DH with the conditional prediction system controls is represented in Figs. 3 and 4. An example will help clarify these figures. In Fig. 3, after each decision by the model to keep the pattern of the prediction hypothesis for trial t for the explanation hypothesis for trial t (B), this decision is compared to the subject's decision (1). If the model's decision was different from that of the subject, control is transferred to the pattern-change mechanism (3 trials). If the model's decision was the same as that of the subject, control is transferred to another part of the monitor (117 trials). Figures 3 and 4 only contain the results for 195 trials because the model began at trial 6.

Conclusions

Deficiencies of the Models

The model for DH and the similar models that have been constructed to simulate the behavior of two other subjects in the binary choice experiment (Feldman, 1959) are deficient in several respects. First of all, the comparison of the behavior of the model to that behavior of the subject from

which the model was developed is, of course, not a very good test of the model. This type of comparison only yields some indication of the adequacy of the model and its components. Comparison of the behavior of the model to sequences of behavior of the subject not used in constructing the model awaits correction of some of the deficiencies mentioned below.

The segment of the model which has the highest number of errors relative to the number of times it is used is the guess-opposite segment (see Fig. 4). The subject certainly exhibits this type of behavior, but the model does not very often predict "guess opposite" when the subject does.

The pattern-change segment has a better error record, but it raises another issue. This segment is actually a selection device. A pattern is selected from the list of patterns that the subject uses. A more elegant pattern-change mechanism would generate a pattern out of the preceding sequence of events and some basic concepts. One of these concepts might be that patterns with equal numbers of P's and C's are preferred to alternation patterns with unequal numbers of P's and C's, all other things being equal.

The models have no mechanisms for making perceptual errors—"seeing" one symbol when another has occurred. Examination of the protocol of DH (Appendix) indicates that he does sometimes think that a C is a P (*e.g.*, trial 196).

The models do not have a sufficiently rich repertoire of hypotheses. Subjects entertain more types of hypotheses about the event series than the two types, pattern and antipattern used in the model for DH. Some subjects entertain more sophisticated hypotheses. For example, one subject was able to detect the fact that a series of events was randomized in blocks of ten trials, *i.e.*, the series had 7 P's and 3 C's in each block of ten trials.

Some evidence also exists that when suitably motivated by money, some subjects in a binary choice experiment will predict the most frequent event on each trial. Models for these subjects require statements of the conditions under which subjects abandon testing other hypotheses or at least abandon testing hypotheses by using them to predict events. Hypotheses could still be considered and tested without using them to predict events.

Contributions of the Models

The consequences of computer simulation for the study of human behavior have been discussed at some length in several places, and I have made a limited statement of my views on this matter in another place (Feldman, 1962). It will suffice then to discuss some of the implications of the work reported here for our understanding of behavior. The computer models of binary choice behavior are relatively simple computer programs; however, they are relatively complex psychological models. A widely accepted view

of binary choice behavior has been the idea of verbal conditioning embodied in the stochastic learning model. In its simplest form, this model says the subject's probability of predicting E_1 or E_2 in the binary choice experiment is an exponentially weighted moving average over preceding events. The verbal conditioning model is hardly consistent with the hypothesis-testing behavior exhibited by DH and a dozen other subjects for whom I have protocols. Protocols of group behavior in the binary choice experiment made available to me by David G. Hays are also consistent with the general idea of hypothesis-testing. Other inadequacies of the verbal conditioning model and evidence for hypothesis-testing models have been discussed elsewhere (Feldman, 1959).

The computer has provided the exponents of hypothesis-testing models of behavior with the means for studying and testing these complex models. Oversimplified explanations of human behavior can no longer be justified on the grounds that the means for studying complex models do not exist. Hopefully, the use of computers to simulate human behavior can extend man's intellect by helping him study his own behavior.

Appendix: Protocol of Subject DH[1]

(All right, now I'll read the instructions to you. I'm going to show you a series of symbols. They will either be a P symbol or a C symbol. Before each word I'll give the signal NOW. When you hear the signal NOW, tell me what symbol you expect will occur on the next trial and why you selected that symbol. That's the purpose of the tape recorder. Take your time. After you have given me your guess, I will show you the correct symbol. Your goal is to anticipate each word as accurately as you can. Please . . . Well, do you have any questions?) Primarily, I just guess whether it'll be a P or a C. (That's it.) But this explaining why I think so. It can be little more than—I think it'll be this, I guess, I have a feeling. How more involved can it be than that? (Well, whatever reasons you have. If those are the only reasons that occur to you as you go thru this, those will be the only reasons. Maybe they won't. OK, we'll try a few and then if you have any questions . . .)

(Now what do you expect the first symbol will be?) P. (OK, the 1st symbol is a C.)

(OK, now what do you expect the 2d symbol will be?) It'll be a P. (Why?) It's pictured in my mind. (OK, the 2d symbol is a C.)

I'll say a C. (Why?) Primarily this time because I'm trying to outguess you. (OK, the 3d symbol is a C.)

(What do you say for the 4th symbol?) I'll say C again. (Why?) This

[1] The statements in parentheses are those of the experimenter.

time I feel it'll be a C. (The 4th symbol is a C. When you give your answer, if you say, "I think the 5th one will be something," it'll be easier to check the tape against the answer sheet.)

(What do you think the 5th one will be?) The 5th one will be a P. (Why is that?) I feel it'll be a P, that's all. (The 5th one is a C.)

(What do you think the 6th one will be?) The 6th one will be a C because you've been giving me C's all along, and I don't think this progression will end. (The 6th one is a C.)

(What do you think the 7th one will be?) The 7th one will be a C because I don't think the progression will be broken. (OK, the 7th one was a C.)

The 8th one will be a C for the same reason. You won't break the progression. (OK, the 8th one is a P.)

(What do you think the 9th one will be?) The 9th one will be a C. (Why is that?) I think that you just gave me the P to throw me off and you'll continue the progression. (The 9th one is a C. Oh, one thing, can you see these cards?) Yes. (Can you see me writing?) No, I can't. (OK.) I'm not looking. (Well, you can look at these cards. I want you to see I'm not picking these out of my head. This set has been predetermined.)

All right. This one will be a P. The 10th one will be a P. (Why is that?) I feel that the progression will start to mix up now. (The 10th one is a C.)

(What do you think the 11th one will be?) The 11th one will be a C. You're continuing the progression. (The 11th one is a C.)

(What do you think the 12th one will be?) The 12th one will be a C because you're continuing the progression. (The 12th one is a P.)

The 13th one will be a C. The 12th one was a P. You were trying to throw me off. The progression will continue. (The 13th one is a P.)

The 14th one will be a P. You're beginning a new progression with P's. (The 14th one is a P.)

The 15th one will be a P. You're still continuing the progression. (The 15th one is a P.)

(What about the 16th one?) The 16th one will be a C. . . . to throw me off now. (The 16th one is a C.)

The 17th one will be a C. You're going to see if I'll revert to the progression of P's. (The 17th one is a C.)

The 18th one will be a P. You're going to break this progression of C's. (The 18th one is a C.)

The 19th one will be a P. You're going to get off this progression of C's. (The 19th one is a P.)

The 20th one will be a P. You're going to try to throw me off trying to make me think that all—think you're going back to the other progres-

sion which I'm confused about now. I don't remember what the last one was—C, I believe. (The 20th one is a P.)

The 21st one will be a C. You won't continue with the progression of P's. (The 21st one is a P.)

The 22d one is a C. You're doing this so that I might think the P progression will continue. (The 22d one is a C.)

The 23d one will be a C. You're trying to make me think that the next one will be a P—going back to the old progression. (The 23d one is a C.)

The 24th one will be a C. You're going to continue the progression of C's. (The 24th one was a C.)

The 25th one is a C. You're still going to continue the progression of C's. (The 25th one is a C.)

The 26th one is still a C. You'll continue the progression. (The 26th one is a C.)

The 27th one is a P. You'll break the progression now. (The 27th one is a C.)

The 28th one will be a P. You're going to break the progression now. (The 28th one is a C.)

The 29th one is a C. You're continuing the progression. (The 29th one is a C.)

The 30th is a C. You'll still continue the progression. (The 30th is a C.)

The 31st is a C. You'll continue the progression. (The 31st is a C.)

The 32d is a C. You'll still continue the progression. (The 32d is a P.)

The 33d is a C. You gave me a P last time to throw me off. (The 33d is a C.)

The 34th is a C. You'll continue the progression. (The 34th is a C.)

The 35th is a P. You're going to throw me off the progression. (The 35th is a C.)

The 36th is a C. You'll continue the progression. (The 36th is a C.)

The 37th is a C. You'll continue the progression. (The 37th is a C.)

The 38th is a C. You'll continue the progression. (The 38th is a C.)

The 39th is a C. You'll continue the progression. (The 39th is a C.)

The 40th is a C. You'll continue the progression. (The 40th is a C.)

The 41st is a C. You'll continue the progression. (The 41st is a C.)

The 42d is a C. You'll continue the progression. (The 42d is a C.)

The 43d is a C. You'll still continue the progression. (The 43d is a C.)

The 44th is a C. You'll still continue the progression. (The 44th is a C.)

The 45th is a C. You'll still continue the progression. (The 45th is a C.)

The 46th is a C. You'll still continue the progression. (The 46th is a C.)

The 47th will be a P. You'll now break the progression. (The 47th is a C.)

The 48th will be a C. You'll go back to the old progression. (The 48th is a C.)
The 49th is a C. You'll continue the progression. (The 49th is a C.)
The 50th is a C. You'll continue the progression. (The 50th is a P.)
The 51st will be a C. You gave me the P to throw me off. (The 51st is a P.)
The 52d is a P. You've begun a progression of P's. (The 52d is a C.)
The 53d is a P. You gave me a C to throw me off. (The 53d is a C.)
The 54th is a C. You'll continue the progression of C's. (The 54th is a C.)
The 55th is a C. You'll still continue the progression. (The 55th is a C.)
The 56th is a C. You'll continue the progression. (The 56th is a P.)
57 is a P. The P will throw me off the progression thinking you had tried to throw me off the C progression with your last P. (57 you said was a P?) P. (57 was a C.)
58 is a C. You began a progression of C's. (58 is a P.)
59 is a C. You're still trying to throw me off with the C's. (59 is a P.)
60 will be a P. You're beginning a progression of P's. (60 is a C.)
61 is a P. You're zigzagging between P's and C's. (61 is a P.)
62 is a C. You'll continue the oscillation. (62 is a C.)
63 is a C—rather 63 is a P because of the oscillation pattern. (63 is a P.)
64 is a C because of the oscillation pattern. (64 is a C.)
65 is a P because of the oscillation pattern. (65 is a C.)
66 is a C. You've begun a progression of C's. (66 is a P.)
67 will be a C. You're oscillating again. (67 is a C.)
68 is a C. You're having a different type of oscillation—2 C's between a P. (68 is a P.)
69 is a C. You're oscillating with C's and P's. (69 is a C.)
70 will be a P. It's the alternate symbol. (70 is a P.)
71 will be a C because of the oscillation sequence. (71 is a C.)
72 will be a P because of the oscillation sequence. (72 is a C.)
73 will be a C. You've begun a new progression of C's. (73 is a C.)
74 is a C. You're continuing the progression. (74 is a C.)
75 is a C. You're still continuing with the progression. (75 is a C.)
76 is still a C. You're continuing with the progression. (76 is a C.)
77 is a C. You're still continuing with the progression. (77 is a C.)
78 is a C. The progression is continuing. (78 is a P.)
79 is a C. The P is to throw me off. The progression continues. (79 is a C.)
80 is a C. The progression will continue. (80 is a C.)
81 is a C. The progression continues. (81 is a P.)
82 will be a C. You're alternating now with C's and P's. (82 is a P.)
83 is a P. You've begun a progression of P's. (83 is a C.)

84 will be a C. The P's were given to throw me off. (84 is a P.)
85 will be a P. You've begun a new alternating sequence. (85 is a P.)
86 will be a C. You're following with a C and 2 P's. Another C will come. (86 is a C.)
87 will be a P. You'll follow the same sequence. (87 is a C.)
88 will be a P. You've begun a sequence of 2 C's and a P. (88 is a C.)
89 is a C. You've begun a new progression of C's. (89 is a C.)
90 is a C. You'll continue the progression. (90 is a C.)
91 is a C. The progression continues. (91 is a C.)
92 is a C. The progression continues. (92 is a P.)
93 is a P. The P's given to me previously to make me think that the progression was being broken and that you would revert to it after the P. The next one will be a P. (93 is a C.)
94 will be a C. You've gone back to the C progression. (94 you say now is a C.) 94 is a C. (OK, 94 is a C.)
95 is a C. You've begun a progression of C's. (95 is a P.)
96 will be a C. You're alternating now with C's and P's. (96 is a P.)
97 is a C. You've begun a progression of a C and 2 P's. (97 is a P.)
98 is a P. You've begun a progression of P's. (98 is a C.)
99 is a C. You've begun a progression of 3 P's and 3 C's. You've already had the 3 P's. 98 (sic) will be a C. (That was . . . 99 is going to be a C. You said. 99 is a C.)
(What's 100?) 100 will be a C. It follows the progression. (100 is a C.)
101 will still be a C. Continue the progression of 3 P's and 3 C's. (101 is a C.)
102 will be a C. You've begun a progression of C's. (102 is a C.)
103 is a C. You'll continue the progression of C's. (103 is a C.)
104 is a C. You'll continue with the progression. (104 is a C.)
105 will be a C. You'll continue the progression. (105 is a C.)
106 will be a P. You'll break the progression now. (106 was a C.)
107 will be a C. You'll continue the progression. (107 was a P.)
108 will be a C. You gave me the P to throw me off. The progression will continue. (108 is a C.)
109 will be a C. You'll continue the progression. (109 was a P.)
110 will be a C. You're alternating with C's and P's. (110 is a C.)
111 will be a P. You'll continue the alternation. (111 was a P.)
112 will be a C. You'll continue the alternation. (112 was a P.)
113 will be a C. You've begun a progression of a C and 2 P's. (113 is a P.)
114 will be a P. You've begun a progression of P's. (114 is a P.)
115 will be a P. You'll continue the progression. (115 is a C.)
116 will be a P. The C was given to throw me off. (116 is a C.)

117 is a C. You've begun a progression of 4 P's and 4 C's. (117 is a P.)
118 will be a P. The progression has changed from 4 P's and 4 C's to 4 P's and 3 C's. (118 is a C.)
119 will be a P. You're alternating with C's and P's. (119 is a C.)
120 will be a C. You're continuing the progression. (120 is a P.)
121 will be a P. You have a progression of 2 C's and 2 P's. (121 is a P.)
122 will be a C. You'll continue this progression of 2 and 2. (122 is a C.)
123 will be a C. You're continuing the progression. (Of what?) Of 2 C's and 2 P's. (123 is a C.)
124 will be a C. You've begun a progression of C's. (124 is a C.)
124 (sic) will be a C. You're continuing the progression. (125 is a C.)
126 will be a C. You're continuing the progression. (126 is a P.)
127 will be a C. You gave me the P to throw me off. (127 is a P.)
128 will be a P. You've begun a progression of P's. (128 is a C.)
129 will be a C. You've begun a progression of 2 P's and 2 C's. (129 was a C.)
130 will be a C. You've begun a progression of C's. (130 is a P.)
131 will be a P. You're continuing the progression of 2 P's and 2 C's. (131 is a C.)
132 will be a P. You're alternating the signs now. (132 is a C.)
133 will be a C. You've begun a sequence of C's. (133 is a C.)
134 will be a C. You're continuing the sequence. (134 is a C.)
135 is a C. You're continuing with the progression. (135 is a P.)
136 will be a P. You've begun . . . you're trying to throw me off now with a 2d P. Think there would be only one P. (136 is a C.)
137 is a C. You're going to continue with the progression of C's. (137 is a C.)
138 is a C. You'll continue the progression. (138 is a C.)
139 is a C. You'll continue the progression. (139 is a P.)
140 is a C. The P was given to throw me off. (140 is a P.)
141 is a C. You gave me the 2 C's (sic) for the same reason as the previous time you had given me the 2 C's 'er 2 P's . . . (141 is a C.)
142 is a C. You'll continue with the progression. (142 is a C.)
143 is a C. You'll continue with the progression. (143 is a C.)
144 is a C. You'll continue with the progression. (144 is a C.)
145 is a P. You'll break the progression. (145 is a C.)
146 is a C. You'll continue the progression. (146 is a C.)
147 is a C. You'll continue the progression. (147 is a C.)
148 is a C. You'll continue the progression. (148 is a C.)
149 is a C. You'll continue the progression. (149 is a C.)
150 is a C. You'll still continue the progression. (150 is a C.)
151 is a C. You'll still continue the progression. (151 is a C.)
152 will be a P. You'll break the progression. (152 is a C.)

153 is a C. You'll continue the progression. (153 is a P.)
154 is a C. You've broken the progression and you'll revert to it now. (154 is a C.)
155 is a C. You'll continue the progression. (155 is a P.)
156 is a C. You're alternating with P's and C's. (156 is a C.)
157 is a C. The alternation of P's and C's was to throw me off the progression of C's. The C progression will continue. (157 is a P.)
158 is a C. You're still going back to C sequence. (158 is a C.)
159 is a C. You're still going to continue this sequence. (159 is a P.)
160 is a C. You have an alternating sequence of P's and C's. (160 is a C.)
161 will be a P. You'll continue to alternate. (161 is a P.)
162 will be a C. You'll continue this oscillation. (162 is a P.)
163 will be a C. You'll continue the alternation. (163 is a C.)
164 will be a P. You'll continue the alternation. (164 is a C.)
165 will be a P. You'll go back to the alternation. (165 is a C.)
166 will be a C. You've begun a sequence of C's. (166 is a C.)
167 will be a C. You've begun a sequence of C's. (167 is a P.)
168 will be a P. You've begun a sequence of 2 C's and 2 P's. (168 is a C.)
169 is a C. The previous P's were given to throw me off. You'll continue the sequence of C's. (169 is a C.)
170 will be a C. You'll continue the sequence. (170 is a P.)
171 will be a P. You'll begin a sequence of P's. (171 is a P.)
172 will be a C. You'll revert to the C's. (172 is a C.)
173 will be a C. You're alternating with 2 P's and 2 C's. (173 is a P.)
174 will be a C. The alternation is a C and a P. (174 is a C.)
175 will be a P. You'll continue this alternation. (175 is a C.)
176 will be a C. You've begun a sequence of C's. (176 is a P.)
177 will be a C. You'll continue with the progression of C's. (177 is a P.)
178 will be a C. You've begun a progression of 2 P's and 2 C's. (What did you say 178 was?) A C. (178 is a C.)
179 will be a C. You'll continue with another C to complete the sequence of 2 P's and 2 C's. (179 is a C.)
180 will be a P. You'll continue this sequence. (180 is a C.)
181 is a C. You've begun a sequence of C's. (181 is a C.)
182 is a C. You'll continue the sequence. (182 is a C.)
183 is a C. You'll continue the sequence. (183 is a P.)
184 will be a C. The P was given to throw me off. (184 is a C.)
185 is a C. You'll continue the sequence of C's. (185 is a C.)
186 will be a C. You'll continue the sequence. (186 is a C.)
187 will be a C. You'll continue the sequence. (187 is a C.)
188 is a C. You'll continue the sequence. (188 is a C.)
189 is a C. You'll continue the sequence. (189 is a P.)
190 will be a C. The P was given to throw me off. (190 is a C.)

191 will be a C. The double P (sic) was given to throw me off a little more. (191 is a C.)
192 is a C. You've . . . been giving me a sequence of 2 P's and 2 C's. (192 is a C.)
192 (sic) is a P. You're continuing the sequence of 2 P's and 2 C's. (193 is a C.)
194 is a C. You've begun a sequence of C's. (194 is a C.)
195 is a C. You'll continue the sequence. (195 is a P.)
196 will be a P. You have a sequence here of inserting 2 P's. (196 is a C.)
197 is a C. The P was given to throw me off. (197 is a C.)
198 will be a C. You'll continue the sequence. (198 is a C.)
199 is a C. You'll continue the sequence. (199 is a C.)
200 will be a C. You'll continue the sequence. (200 is a C.)

A MODEL OF THE TRUST INVESTMENT PROCESS

by Geoffrey P. E. Clarkson

The object of this study is the investment of trust funds held by banks in the United States—funds that currently amount to nearly $60 billions. The purpose of our model is to simulate the process employed in the investment of trust funds in common stocks. When making a decision a trust officer in a bank is confronted with a large assortment of information. Information abounds on the operation of firms and the market valuation of their stocks, and published reports make predictions about the future state of the general economy and stock market. When an investor acts in an agency or fiduciary capacity, legal restrictions and the desires of his client must also be considered. These factors, when evaluated and combined into an investment program, ultimately result in a decision to buy specific quantities of particular stocks and bonds. Thus, an investor choosing a portfolio is processing information: he sorts the useful from the irrelevant, and decides which parts of the total information flow are most important.

The investment process is a problem in decision-making under uncertainty. Our model, written as a computer program, simulates the procedures used in choosing investment policies for particular accounts, in evaluating the alternatives presented by the market, and in selecting the required portfolios. The analysis is based on the operations at a medium-sized national bank[1] and the decision-maker of our model is the trust investment officer.[2] We require our simulation model to select portfolios

[1] The trust assets of this bank are approximately equal to the average for all national banks.

[2] It should be noted that our model reflects the behavior of one investor and hence

using the same information that is available to the trust officer at the time his decisions are made.

Postulates and Data for the Model

Since our model is a theory of individual decision-making behavior, the method of analysis is based on the theory of human problem-solving (Newell, Shaw, and Simon, 1958*a*). In keeping with the postulates of this theory, the main postulates for the analysis of the investment decision process are that there exist:

1. A *memory* that contains lists of industries each of which has a list of companies associated with it. The memory also contains information associated with the general economy, industries, and individual companies.[3]
2. *Search* and *selection* procedures that perform the task of searching the lists of information stored in memory, selecting those bits that have the required attributes, regrouping the selected items of information into new lists, and performing algebraic operations when necessary. These procedures function in a manner similar to a clerk who prepares lists of stocks suitable for current investment by scanning a master list.
3. A *set of rules* or criteria that guide the decision-making process by stipulating when and how each decision process is to be used. The set of rules constitutes the structure of the decision processes for an individual investor. It might be compared to the "rules of thumb" of the traditional "expert," but there is an important difference—namely, the set of rules must be defined unambiguously.

In common with other problem-solving programs, the processes are used iteratively and recursively. Lists of industries and companies are searched for particular attributes; sublists are created, searched and divided again. For example, to obtain a high growth portfolio, the list of companies stored in memory is searched to obtain securities with the desired characteristics. Additional criteria are employed to narrow (or expand) this list. Further search and testing against desired criteria yields the specific selection of stocks to buy.

Like the investor it simulates, the program stores the final result (list)

---

may not describe the general case. The implications of this study for more general theories of investment are discussed in Clarkson (1962), chap. 8.

[3] Investors categorize companies by industry. Not all investors may associate identical companies with a given industry, but the process of classification by industry remains invariant as the primary basis for listing companies in the memory. The information associated with each company also varies among investors, but each may be represented as having a list of attributes with their values stored in memory, *e.g.* growth rate, dividend rate, price earnings ratio, expected earnings, expected yield, etc.

for future use. If the same problem reoccurs, the entire process need not be repeated. The list may be judged by present criteria, accepted, adapted to meet new conditions, or completely rejected. In the latter event the program would renew search and selection activity until a new list had been formed.

To define a model of trust investment behavior within this general framework we require the basic rules (operations) used in making a decision to purchase particular securities. To obtain these data, trust departments of several local banks are studied by interviewing departmental officers and by observing behavior at committee meetings called to review past and future decisions. Attention was then focused on an investment officer who was chiefly responsible for all decisions relevant to the choice of portfolios within a particular bank. The history of several accounts were examined and naïve behavioral models were constructed to help uncover these decision processes that appeared to be invariant among accounts.

In an attempt to confirm or refute these hypotheses, the trust officer was asked to permit "protocols" to be made of his decision processes.[4] These protocols recorded the trust officer's decision processes for accounts that arose in the course of his work. The decisions made during those problem sessions determined the particular securities that were purchased for those accounts.

Close inspection of the protocols revealed that many of the decisions pertaining to the formulation of expectations, and the evaluation of industries, companies and stocks were made before the selection of a particular portfolio began. In an attempt to discover how these prior decisions were made a new approach was taken. The trust officer was asked to read articles from financial journals and analysts' reports, to which he subscribed, and comment on the ideas, forecasts, facts, etc., presented in the articles. Protocols of these thought processes were more successful in that they revealed many of the decision processes subsumed in the earlier transcripts

On the basis of these data and analytic techniques, a model was constructed. The model considers the problem of investing the funds of new accounts in common stocks. It does not directly consider the problem of allocating the funds among bonds, preferreds, and common stocks. The trust investment model is stated in terms of a computer model and is presented in the next section.[5]

[4] A "protocol" is a transcript of the verbalized thought and actions of a subject when the subject has been instructed to think or problem-solve aloud. Thus, the transcript is a record of the subject's thought processes while engaged in making a decision. Since a protocol is a detailed description of what a person does it avoids some of the problems inherent in interview and questionnaire techniques that ask the subject to state his reasons for behaving as he does. For further discussion see Newell, Shaw, Simon (1958*a*).

[5] The program is written in Information Processing Language V (Newell, 1961*e*).

The trust investment process can be divided into three parts: (*a*) the analysis and selection of a list of stocks suitable for current investment—the "A" List, (*b*) the formulation of an investment policy, and (*c*) the selection of a portfolio. Each of these sections can be also broken down into a number of subsections (see Fig. 1).

The process of selecting a current list of stocks [step (*a*)] entails an analysis of individual companies as well as an appreciation of the factors affecting their respective industries and the economy as a whole. The problem of formulating an investment policy [step (*b*)] involves a process that translates the information on the beneficiary or client into an investment goal that will yield the desired combination of income and/or appreciation. This process requires the trust investor to consider such things as the effect of taxes on the stream of income generated by the portfolio as well as the stability of that stream. The actual selection of a portfolio [step (*c*)] follows directly from steps (*a*) and (*b*). While the selection

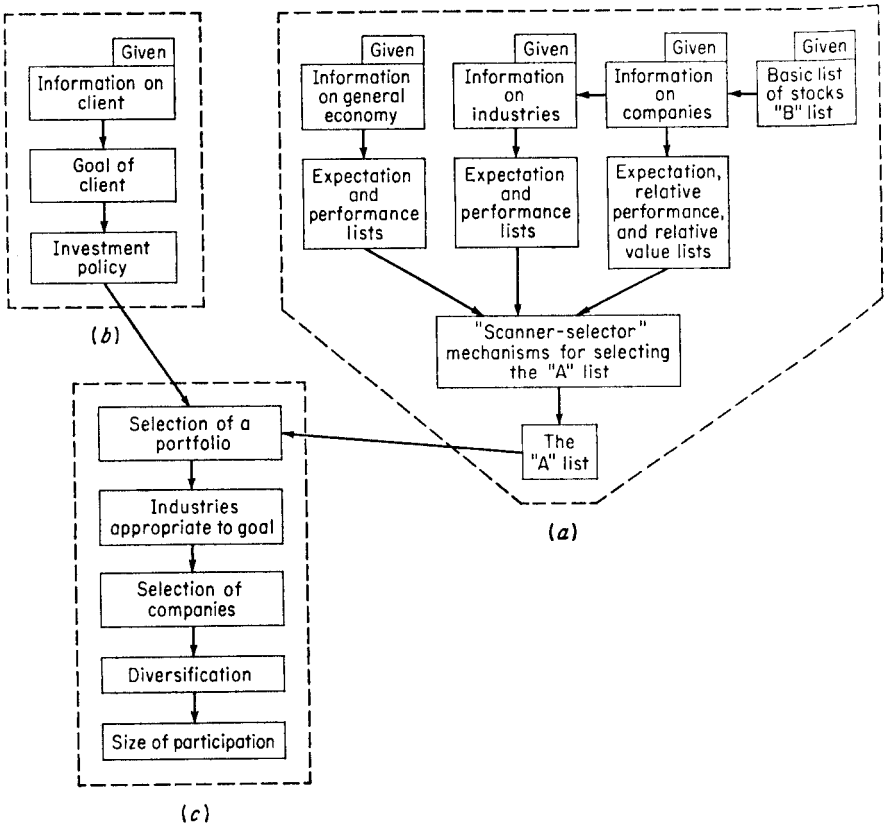


Figure 1. Structure of decision process.

procedure contains rules on diversification and on how to determine the size of participations, the essence of the process lies in carrying the prior analysis to its logical conclusion.

In presenting this model of trust investment behavior, we shall follow the outline of the process given in Fig. 1 so that each subsection as well as the interrelations can stand by themselves for critical appraisal.

Having outlined the investment process and the method of analysis used in constructing the model, the only question that needs to be examined before proceeding with a description of the model is the effect of the organization and the fiduciary relation on the trust investment process.[6]

Since banks are responsible for all investments made in their name, elaborate procedures are set up to review and approve all investment decisions.[7] Also, the necessity of being able to justify their investment decisions in a court of law has led trust investors to create a set of criteria with which to judge the quality of any given portfolio or investment. For all practical purposes these criteria can be reduced essentially to one maxim: A security is of investment quality *if and only if* it is being bought or is being held by other leading trust institutions.[8] Clearly, this maxim is circular in nature and if strictly true would preclude change. However, the smaller the bank the truer the maxim, which implies that innovations must come from the larger banks acting by themselves or in small groups. If innovations do not occur very frequently, the maxim then asserts that the general list of stocks that are considered suitable for trust investment will remain fairly stable over time. The addition of a further observation, namely, that trust investors eschew taking losses, *i.e.*, selling stocks whose prices have fallen below the purchase price, allows an even stronger prediction to be made. The basic list of stocks—the "B" List—that are considered to be suitable for trust investment by a particular bank will remain fairly stable over time, any changes being in the form of additions. Thus, for any given trust investor, the basic list of stocks from which he can choose is given to him by the historical record. At any particular point in time an investor selects stocks from a subset of his basic list. This subset

[6] As we are principally concerned with the investment of trust funds for individual accounts, the important constraints are those that are imposed on the investor by the banking institution and the fiduciary relation with the client.

[7] "All investments of trust funds shall be made, retained or disposed of only with the approval of the Trust Committee. . . . The Trust Committee shall, at least once during each period of twelve months, review all the assets held in or for each fiduciary account to determine their safety and current value and the advisability of retaining or disposing of them." Excerpt from the *Trust Manual* of a National Bank.

It is interesting to note that this Trust Committee is appointed by the Board of Directors and is composed of the President, the Vice-President in charge of investments, the Vice-President in charge of trusts, and other officials.

[8] By a simple substitution of words this maxim can roughly be applied to the composition of portfolios, *i.e.* the ratio of common stock to bonds and preferred stocks.

is a proper subset of the "B" List and is defined by a concept of relative valuation. As expectations, prices, yields, and other metrics change with time, so does the content of this subset which is called the "A" List.

Hence, institutional constraints reduce the problem of determining the list of stocks from which, at a given point in time, an investor actually chooses—the "A" List—to one of "stocks" and "flows." Since the "stocks" change slowly with time the model assumes them to be given and takes as part of its goal the analysis and prediction of the "flows."

1. Selection of the Current List of Stocks—The "A" List

In this section we shall present the data and the mechanisms that the model uses to evaluate and select the stocks for the "A" List. Unlike the model's processes for steps (*b*) and (*c*), the mechanisms described in this section are not intended to be a reproduction of the analytic procedures used by the trust officer each time he selects a new portfolio. To reproduce

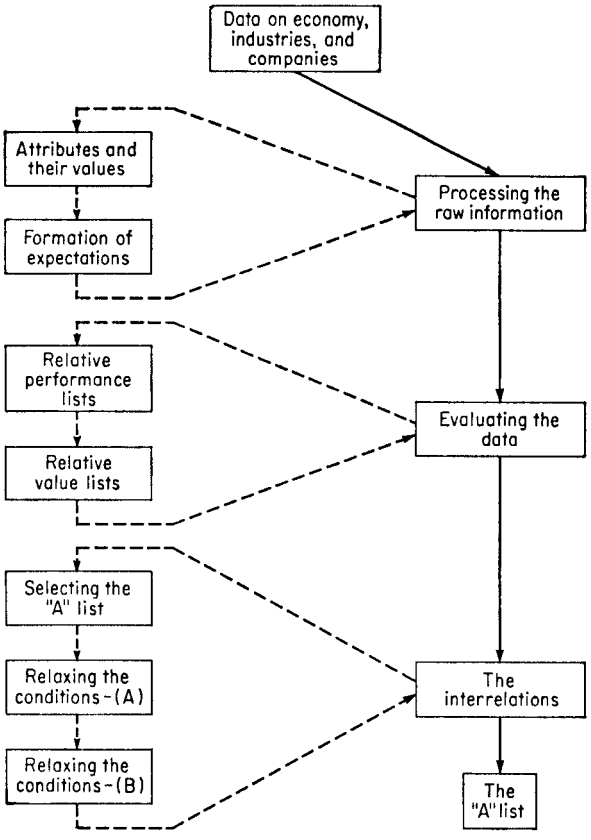


Figure 2. Selecting the "A" List.

only those procedures would require us to ignore all the data on each company that he has collected and processed in preceding years. To take the historical data into account, the model must employ a set of mechanisms that generate the same sorts of measures and comparative data that the trust officer actually employs when he is selecting a portfolio. Clearly, the trust investor (unlike the model) does not evaluate all companies at one time. But, our object is to use that set of mechanisms that yield the right kind of data and measures of performance. Thus, the processes described in this section should not be viewed as a complete simulation of what the trust officer does prior to each portfolio selection, but rather as an approximation of the processes he has used over the years in order to build up a set of measures by which the performance of a company can be judged. Our success in this respect will be tested later on.

In order to describe the processes that are involved in the selection of the "A" List it is necessary, at first, to treat some of the mechanisms as though they were independent of each other. While this is not in fact the case, the ways in which they are interrelated will be discussed after the data processing mechanisms have been described. To facilitate this explication a flow chart of the selection procedure is presented in Fig. 2.

1A. PROCESSING THE RAW INFORMATION

Although the information used to derive the current list of stocks is classified into three main categories, e.g., general economy, industry, and company, the processes by which the information is handled are roughly the same. Differences occur in the content of the information processed and the manner in which interrelations are formed, but the basic structure of the sorting and evaluating processes remains the same.

For each category there is a set of attributes that correspond to the important variables in that category. For example, for all companies the set of attributes consists of sales, earnings, cash flow per share, profit margin, working capital, price earnings ratio, dividend payout ratio, dividends per share, dividend yield, and prices. The values of these attributes are their numerical values, and these are determined by the information which is fed into the model. Since the values will reflect the changes that occur in economy, industry and company variables those that change frequently are readily distinguished from those that do not. Those that change infrequently with time reflect the general trend of the economy, industry, or company, while the others indicate those attributes that are more sensitive to short-run fluctuations. The mechanisms that derive these values are the same in all cases, and it is to these processes that attention is now directed.

1A1. Determination of Attributes and Their Values. All information, except that dealing with economy or industry forecasts, is fed into the model in numerical form. These data consist of the historical values of each

attribute in the system for the last ten years.[9] The data are entered in the form of lists, and from these basic lists the model generates, for each attribute, three additional lists. The first of these lists contains the mean of the ten historical values. The second contains a set of nine values which record the rate of increase (or decrease) of each value in the historical record over the value immediately proceeding it. The third of these lists contains the average rate of change of the values for the entire ten year period. For each attribute, then, the model contains the four following lists of information:

(i) *Current Value.* This list contains the last ten annual values of each attribute arranged chronologically so that the most recent is at the head of the list.

(ii) *Ten-year Average.* Each time a new value is added to (i) a new average of the ten values is placed on this list. Thus, this list contains a ten year moving average of the values in (i).

(iii) *Recent Changes.* This list contains the rate of increase (or decrease) of each value in the Current Value List over the value immediately below it. Thus, if the values of the Current Value List are called x_i, where $i = 1, 2, \ldots, 10$, then the Recent Change List will have nine entries whose values will equal:

$$\frac{x_i - x_{i+1}}{x_{i+1}} \qquad \text{where } i = 1, 2, \ldots, 9$$

(iv) *Average Rate of Change.* This list contains the average rate of change of the values on list (i) for the entire ten year period. Like list (ii), this is revised every time there is a new entry on the Current Value List.

Hence, the basic information which is given to the model is processed so that it is expressed in terms of rates of change and/or ratios which are directly comparable throughout the system.

1A2. The Formation of Expectations. Information on forecasts is fed into the model in two different forms. Forecasts for economy and industry attributes are converted for input into a three-valued scale "above," "below," or "equal to." The entry is based on the published predictions that the value for a given attribute is going to rise, fall, or stay the same over the next interval of time. Numerical data is not used in an attempt to avoid the chaos of averaging the array of forecasts found in financial literature.

For the analysis of company performance, however, numerical forecasts

[9] The attributes themselves are taken as given. They were derived by an analysis of trust investors' decision processes and by observing which variables are considered important by investment services.

The data for economy and industry attributes was taken from *Moody's Industrials, Review of Current Business,* and *Statistical Abstracts,* while data for company attributes was taken from the *Value Line Investment Survey.*

are needed, and in a further effort to avoid conflicting opinions all forecasts for company attributes are taken from the *Value Line Investment Survey*.

All forecast attributes have the current forecasts as their only value. Previous forecasts are not kept and the model takes each forecast at face value without making any attempt to judge its "goodness" or "record of success." This procedure may not be too realistic as it ignores the effects of personal preferences on perception. But, the model is not equipped to handle "second guessing" and other judgmental modifications and the information is assumed to be reliable. Before discussing the role of expectations in our model, it is necessary to mention some further behavioral characteristics of trust investors.

By and large, trust investment is long-term investment. As previously noted, trust investors do not engage in trading stocks for their clients, but look to the long-term growth of the economy and the market to justify their investments. This is not to say that they remain aloof from daily, monthly, or yearly fluctuations, but rather that their emphasis is on the analysis of industries and their respective companies. Their basic belief is that the market will eventually recognize a company's "true value." Hence, in general, trust investors analyze companies and not the market.

Clearly changes in the market do affect investor behavior, but the effects are more in keeping with a feedback mechanism than one where the investor acts on the basis of his own market forecasts. Thus, attributes containing forecasted information are included in this model, but they receive different amounts of attention depending on whether the attributes belong to the economy, industries, or specific companies. Since the content of the Expectation Lists varies as well as the form, these lists are described in turn.

(i) *Economy and Industry Expectation Lists.* For each attribute in both of these categories the Expectation Lists contain two entries. The first is the forecasted value for that attribute converted into the input form of "above," "below," or "equal to." The second is the first value on the Recent Change List—namely, the rate of change of that attribute for last year—converted into the same three-valued scale.[10] Hence Economy and Industry Expectation Lists contain pairs of "aboves," "belows," or "equals to" which under two possible sets of conditions will form a pattern of only "aboves" or "belows."

(ii) *Company Expectation Lists.* Expectation Lists exist for five of the ten company attributes.[11] These Expectation Lists contain one or two entries all of which are in numerical form. These entries are derived from

[10] In this case the three-valued scale is recording whether the rate of change for this attribute last year was positive, negative or zero.

[11] The five attributes which have forecasted values are: sales, earnings per share, cash flow per share, profit margin, and dividends per share.

the twelve-month and three- to five-year forecasts recorded in the model for these attributes. The first entry is on all Expectation Lists and is obtained by converting the twelve-month forecast into an expected rate of change. The second entry exists only for sales and earnings per share Expectation Lists and is obtained by converting the three- to five-year forecast into an expected average rate of change.

1B. EVALUATING THE DATA

Logically this section should contain all the procedures of evaluation used in this model. However, in order to simplify the problem of describing the actual mechanisms, the processes of evaluation have been divided into two parts. Those that pertain to the information within each major category, *i.e.*, economy, industry, and company, are examined here; those that involve the interrelations between these sections are discussed in Sec. 1C.

The model evaluates the data by creating two main lists: the Relative Performance List, and the Relative Value List. As these processes are described in some detail it is worth pausing for a moment to make a list of the information already gathered for each attribute of each company:

(i) A list of the last ten values of the attribute
(ii) The mean of these ten values
(iii) A list of the rates of change of these values
(iv) The mean of these rates of change
(v) For relevant attributes an Expectation List that contains the expected rate of change for the coming year and, in the case of the sales and the earnings per share attributes, the expected average rate of change for the next three to five years

Attention has been drawn to this information as the processes of evaluation use these data as inputs.

1B1. The Relative Performance List. In order to determine the relative performance of each company within its given industry a list is made for *each* of the basic lists for *each attribute* of the mean for that attribute for *each company*. Hence, for each attribute there is now a list of means each of which belongs to a particular company within a given industry. The average of this list of means is taken so that we now have a distribution of means for a given attribute, plus the mean of that distribution. The deviation of each mean from the distribution mean is calculated as a percentage deviation and is then converted into the three-valued scale "above," "below," or "equal to." These per cent deviations from the distribution mean are recorded on the Relative Performance List of each attribute.

To classify this process further let a_{ij} represent the class of all company attribute means where:

$i = 1, 2, \ldots, n$ represents the number of attributes for each company
$j = 1, 2, \ldots, m$ represents the number of companies for each industry

Then the matrix $A_{n,m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$

is the row-by-row array of means for each attribute, for all companies within a given industry. The mean of the distribution of means for attribute i is given by:

$$\bar{a}_i = \frac{1}{m} \sum_{j=1}^{m} a_{ij}$$

The list of all such means forms the vector

$$\bar{A}_n = \begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{a}_n \end{bmatrix}$$

To determine the deviations of each a_{ij} from its respective mean $\bar{a}_i$ the model takes the difference $(a_{ij} - \bar{a}_i)$ as a per cent of $\bar{a}_i$. Hence, the deviations for each attribute for all companies are given by the row-by-row array:

$$\begin{bmatrix} \frac{a_{11} - \bar{a}_1}{\bar{a}_1}, & \frac{a_{12} - \bar{a}_1}{\bar{a}_1}, & \ldots, & \frac{a_{1m} - \bar{a}_1}{\bar{a}_1} \\ \frac{a_{21} - \bar{a}_2}{\bar{a}_2}, & \frac{a_{22} - \bar{a}_2}{\bar{a}_2}, & \ldots, & \frac{a_{2m} - \bar{a}_2}{\bar{a}_2} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \frac{a_{n1} - \bar{a}_n}{\bar{a}_n}, & \frac{a_{n2} - \bar{a}_n}{\bar{a}_n}, & \ldots, & \frac{a_{nm} - \bar{a}_n}{\bar{a}_n} \end{bmatrix}$$

These percentage deviations are then converted into the three-valued scale "above," "below," or "equal to" where the base for the comparison is given by a five per cent boundary level either side of the distribution mean $\bar{a}_i$. Thus for the relevant[12] attribute there is a Relative Performance List on which is recorded:

[12] All Relative Performance Lists contain items (i) and (ii). Lists for attributes cash flow per share and profit margin contain items (i), (ii) and (iii). While lists for sales, earnings per share, and dividends per share attributes contain all four items.

(*i*) The mean value over the last ten years as well as whether this mean is "above," "below," or "equal to" the mean for this attribute for the other companies in this industry.

(*ii*) The mean rate of growth over the last ten years as well as whether this mean is "above," "below," or "equal to" the population mean for this attribute.

(*iii*) The expected rate of growth over the coming twelve months as well as whether this expected rate of growth is "above," "below," or "equal to" the mean of the population of expected rates of growth.

(*iv*) The mean, expected rate of growth over the next three to five years as well as whether this mean rate of growth is "above," "below," or "equal to" the mean of the population of mean, expected rates of growth.

1B2. The Relative Value List. Having described the procedures that determine the Relative Performance of each company within its industry, we will now examine the set of processes that determine the Relative Value of each company's stock.

As noted above each company has an attribute that records a three- to five-year forecast of its earnings per share. Although this is only an estimate, the figure is assumed to be reliable and is used, for each company to form a price earnings ratio of the forecasted earnings. As the model already contains the values for the current price earnings ratio and the historical mean of the prices earnings multiple for that company, the entries for the Relative Value List are as follows: The first consists of the difference between the mean price earnings ratio and the price earnings ratio of the forecasted earnings. This difference is taken as a per cent of the mean and is recorded as "above," "below," or "equal to" the historical mean. The second entry consists of the difference between the historical mean and the current price earnings ratio. As before, this difference is taken as a per cent of the historical mean and is recorded as "above," "below," or "equal to." To clarify this process, let:

P = current market price
E = expected earnings per share for the current year
E^* = forecasted earnings per share three to five years from now
$\bar{P}/E^*$ = ten year average of price earnings ratio

Then for each company the calculations are as follows, the results of each being recorded as "above," "below," or "equal to."

$$\text{(i)} \quad \frac{(\bar{P}/E) - (P/E^*)}{\bar{P}/E}$$

$$\text{(ii)} \quad \frac{(\bar{P}/E) - (P/E)}{\bar{P}/E}$$

The Relative Value List contains these results, plus their value on the three point scale in the order that they are produced. Thus, for each company the Relative Value List is a pair of "aboves," "belows," or "equals to" which under two possible conditions will form a pattern of only A "aboves" or "belows."

1C. THE INTERRELATIONS

Up to now we have described the mechanisms which process the data as though they were independent of each other. While this is true to a certain extent, these mechanisms are related by the processes that select the stocks suitable for current investment. In order to present these interrelations in as orderly fashion as possible, we will first examine the processes that select the "A" List under simplified conditions. By relaxing these conditions we will be able to examine the complexities as they occur.

To facilitate the exposition it is necessary to assign names to the two values which appear on the Relative Value List. Hence, if we let:

$$x = \frac{\bar{P}}{E} - \frac{P}{E^*} \qquad \text{and} \qquad y = \frac{\bar{P}}{E} - \frac{P}{E}$$

we can, in the future, refer to the values of x and y of the Relative Value List.

1C1. Selecting the "A" List. For simplicity, we shall first assume that all Economy and Industry Expectation Lists have both of their values reading "above." For such a condition to hold, the economy would have to be in the middle of a roaring boom. But ignoring this implication for a moment, we can now examine the basic operations of the selection mechanism which is composed of two parts:

(i) *The Scanner.* This mechanism examines each Economy and Industry Expectation List in turn and notes the values of adjacent pairs. In this case all adjacent pairs have the same value, *i.e.*, "above." Hence, having completed its search and finding such perfect accord the Scanner halts and the Selector takes over.

(ii) *The Selector.* Under such ideal conditions the selection process consists of searching through the Relative Value Lists of all companies and placing on the "A" List those companies whose Relative Values are recorded as:

(x) = "above," or "equal to"
(y) = "above," "equal to," or "below"

1C2. Relaxing the Conditions—A. Throughout this discussion it must be remembered that information is fed into the various categories, i.e., economy, industry, and company, at different intervals of time. Although these intervals may be chosen to suit any particular set of requirements,

we have assumed the following time lags: Information on economy and industry attributes is fed in quarterly while company attributes are adjusted monthly.

Given these time differentials we will now examine the effects of adding new information, to the respective categories, in the order in which they are assumed to occur.

(i) After a change in prices or earnings per share the information is processed as per Secs. 1A and 1B above, and new values are placed on the Relative Value List. The Scanner then proceeds to check the Economy and Industry Expectation Lists and finding them unchanged initiates the selection procedure. The Selector examines the "A" List first and removes from it any companies whose entries on their Relative Lists have changed to:

$$(x) = \text{"below"}$$

The Selector then proceeds to the remaining list of companies and places on the "A" List all companies whose entries on their Relative Value List now record:

$$(x) = \text{"above" or "equal to"}$$

(ii) At the end of each quarter, new information is entered into the model on economy and industry attributes and, when relevant, on company attributes as well. Whenever new information is fed in it is processed immediately, as per Secs. 1A and 1B, and the attention of the Scanner is directed toward that category which received the new information. When more than one category receives new information, the Scanner always goes to the most general category first, *e.g.*, economy or industry, and then proceeds down through the categories noticing and recording changes as it goes. At this point changes in the Economy and Industry Expectation Lists are translated into one of two values, "hold" or "delete hold." These values are placed on the Relative Value List. Companies which were previously on the "A" List are not taken off the list. They are left there until the new information on the companies themselves decides the issue of whether they should stay on the list or not.

1C3. Relaxing the Conditions—B. In order to examine all the operations of the Scanner and the Selector, changes in the forecasted values of the Economy and Industry Expectation Lists will be divided into three categories:

(i) *Forecasted Value Falls below Recent Change Value.* As noted earlier, the function of the Scanner is to examine the Economy and Industry Expectation Lists of all the attributes that have received new information. In this case let us assume that information has been entered

into the model which forecasts a leveling off in capital spending, while at the same time the most recent change in this index is still rising. Given this change the Scanner will first proceed to the capital spending Expectation List. Noticing that the other economic Expectation Lists are unchanged the Scanner will descend a level and create a list of the capital intensive industries. The Scanner then examines the changes that have occurred in the Industry Expectation Lists. Since the forecasts for some of these industries will also have fallen or leveled off, the list of affected industries is reduced to that set whose forecasts have been lowered.[13]

The Selector then takes over and scans the list created by the Scanner and searches the "A" List for companies belonging to those industries. All such companies are subjected to the following test:

(α) Mark all companies "hold" which have entry (x) on the Relative Value List recorded as "equal to."

If the forecasts for the other economic attributes fall, the Scanner searches all industry Expectation Lists for corresponding changes, makes a list of those industries whose forecasted values have fallen and presents this list to the Selector which applies the same set of tests as before.

(ii) *Recent Change Value Falls Below Forecasted Value.* In this case the functions of the Scanner and Selector are essentially the same as in (i) except that the Selector applies one extra test.

If economic indices have turned down the performance of some industries and companies will also have turned down. This means that basic changes in company evaluations may be taking place at the same time. However, since these changes are completed first the function of the Scanner is still to create a list of the affected industries, and of the Selector to apply the following tests to those companies on the "A" List which belong to the affected industries.

(α) Mark all companies "hold" which have entry (x) on the Relative Value List recorded as "equal to."

(β) Mark all companies "hold" which have entry (y) on the Relative Value List recorded as "below."

(iii) *Forecasted Values and Recent Change Values Both Turn Down.* Under these conditions, although the Scanner performs in the same manner, a change occurs in the tests applied by the Selector. Instead of testing the companies presented to it by the Scanner on the basis of the tests given above, the Selector makes the following more rigorous tests:

[13] The assumption here is that the forecasts for total capital spending cannot change without a corresponding change in one or all of the capital intensive industries. The only exception to this rule is the Construction Industry which is also included on the list of industries to be examined if there is a fall in the expected level of capital spending.

(γ) Remove all companies from the "A" List which have entry (x) on the Relative Value List recorded as "equal to."

(β) Mark all companies "hold" which have entry (y) on the Relative Value List recorded as "below."

Clearly, the three categories of forecast changes are not mutually exclusive and at any given point in time one would not expect to find the model in one particular category but rather in some combination of the three. This situation in no way changes the functions of the Scanner and Selector; it merely requires them to take each category in turn and perform the required operations sequentially.

When forecast and recent change values are moving up, instead of down as described above, the testing procedures of the Selector are reversed. Instead of marking companies with "hold" and removing them from the "A" List, a "hold" is replaced by a "delete hold" and companies are restored to the "A" List.

The Formulation of an Investment Policy

By and large, trust investors formulate investment policies for two types of funds: (1) large trust funds, *e.g.,* Common Trust Funds (excluding Pension and similar types of funds, and (2) individual trust accounts.

As we are primarily interested in the investment decisions pertaining to the latter set of accounts, the model does not consider the problem of investing the funds of Common Trust Funds. The decision on whether to invest an account in a Fund or not, however, is relevant to the decision process. Although the rules governing this process are not explicitly included in the model—that is, the model is only concerned with investing the funds of individual accounts—a brief discussion of these rules is included here.

2A. COMMON TRUST FUNDS

As the cost of management per dollar invested is much lower in Common Trust Funds than in individual accounts, banks prefer to invest small accounts in their funds. In order to persuade clients to participate in these funds, banks are forced to make the funds' goals explicit. In practice these funds have goals which range from an emphasis on capital appreciation to stability of principal with emphasis on current income.

As the legal restriction governing the investment of Common Trust Funds have ben discussed elsewhere (Clarkson, 1962), the rules outlined here pertain only to the decision on whether to invest the assets of individual accounts in one of these funds.

(*a*) All "legal"[14] trusts are eligible for investment in a Common Trust Fund. Accounts which are not legal trusts and/or whose beneficiaries have waived legal requirements are not so invested.

(*b*) All legal trusts that have assets of less than K dollars are automatically placed in a Common Trust Fund.[15]

(*c*) Legal trusts greater than K dollars may or may not be placed in a Common Trust Fund. As noted before, no account may participate for more than $100,000. Thus, in the range between K dollars and $100,000 the decision will be determined by the degree of correspondence between the goals of the account and the expected results of the Common Trust Fund.

2B. INDIVIDUAL TRUST ACCOUNTS

In order to determine a client's goal, the investment officer has two main sources of information: an administrative officer's interview with the client, and the written record. The former provides the investor with some subjective impressions of the client and the latter with a copy of the legal instrument (often a will) setting up the trust. In most cases this document contains information about the beneficiary, the investment powers of the bank, what is to be done with the principal, the desired amount of income, etc. The instrument also contains information about the beneficiary's age, marital status, number and age of dependents, place of legal residence, income-tax bracket, and status and age of future beneficiaries if any.

Armed with this data, the investment officer must now decide on an investment policy for the account. This policy must lie somewhere along the continuum between the extremes of growth and income and the process that determines it is as follows:

[14] " 'Legal investment' statutes fall into two general categories: (1) those that restrict all or part of the investments to specific investments or specific classes of investments, and (2) those that limit investment in non-legal securities to a given percentage of the account or fund. The statutory limitations on investment in non-legal securities range from 30 percent to 50 percent of the market value (in one state, inventory value) of the fund." Survey of Common Trust Funds, 1959, *Federal Reserve Bulletin,* May, 1960, p. 480.

Pennsylvania belongs in the first category and "legal" stocks are defined by law (Act No. 340, 1951) as those securities which, if preferred stocks have paid dividends for sixteen years and which, if common stocks have had positive earnings and have paid dividends in twelve out of the last sixteen years. A list of securities meeting these requirements is prepared by the Pennsylvania Bankers Association. (*Corporate Securities Considered Legal Investments for Trust Funds in the State of Pennsylvania,* Trust Division, Pennsylvania Bankers Association, October, 1960).

Many people when setting up the trust relation specifically waive these investment restrictions. Thus, "legal" refers to situations in which the investment officer must comply with these investment restrictions.

[15] To protect this Bank's anonymity, the precise dollar figures are not revealed. Nationally, the average Common Trust Fund participation is approximately $23,000. *Federal Reserve Bulletin,* May, 1960, p. 481.

(1) *The Scanner.* Information on the client is fed into the model in the form of a list which contains the following attributes: (i) The desired amount of growth, (ii) The desired amount of income, (iii) Whether current income is sufficient for the client's needs, (iv) The desired amount of stability of income and principal, (v) Income-tax bracket, (vi) Client's profession, (vii) Client's place of legal residence, (viii) Whether trust is revocable or not, and (ix) Whether trust is legal or not. The function of the Scanner is to proceed through the first six of these attributes testing for the value of each in turn.[16] The tests consist of classifying the values of attributes (i), (ii), (iv), (v), and (vi) on the basis of whether they are below a median value or not. The criteria for these tests are given to the model in advance and the Scanner converts the values of the attribute into a two-valued scale—"Low," or "Not Low"—which correspond to being below or not below the particular criterion. Attributes (iii) and (vii) are scaled on a "Yes," "No" basis.

The results of these tests are placed on a list so that for each client there is a particular pattern of test answers. Thus for a client in the legal profession, who is a resident of Pennsylvania and has a large current income, a high tax bracket, and desires to build an estate to provide for his retirement, the pattern generated by the Scanner would read: (i) "$\sim$ Low," (ii) "Low," (iii) "Yes," (iv) "Low," (v) "$\sim$ Low," (vi) "$\sim$ Low," (vii) "Yes."

(2) *The Selector.* The function of the Selector is to take the list generated by the Scanner and convert it into the appropriate investment policy. Clearly, the number of possible combinations of growth and income is large. But, in practice they can be characterized in the following manner:[17]

(i) *Growth Account.* In these accounts assets are expected to appreciate at an average rate of 10% per year over a ten-year period. Income is not stressed and fluctuations in principal are tolerated.

(ii) *Growth and Income Account.* Here assets are expected to appreciate at 5–6% per year, while dividend yield should approach 2–3% per year.

(iii) *Income and Growth.* In this type of account assets are only expected to appreciate at 3–4% per year. The desired dividend yield is 3–4% per year and the stability of the income stream is stressed.

(iv) *Income Account.* Here the size and stability of the income streams are stressed with the expected dividend yield being

[16] Attributes (viii) and (ix) are used by the portfolio-selection process.

[17] It should be noted that the figures used here are in no way fixed and will in fact vary with changing market conditions.

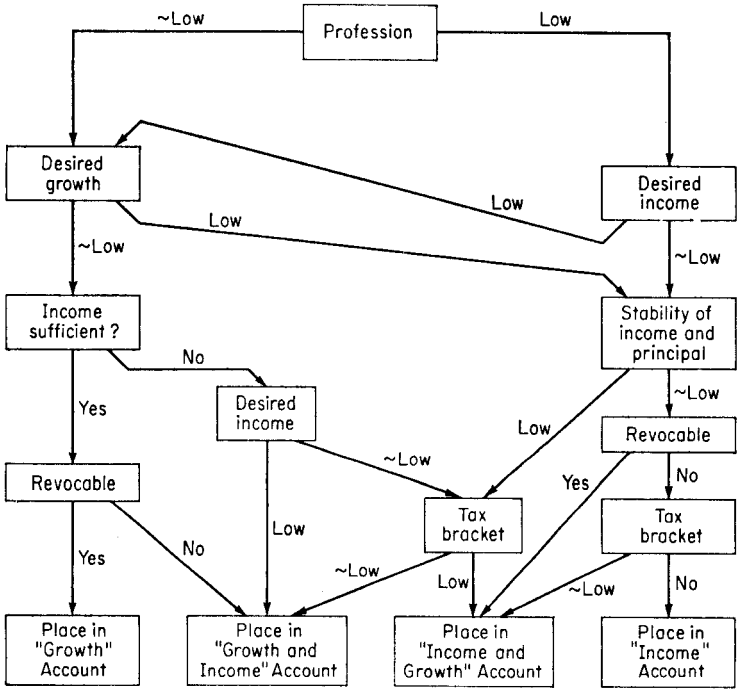


Figure 3. Selector for investment policy.

4–6% per year. In this type of account growth (capital appreciation) is not stressed.

The Selector chooses an investment policy for a particular client by applying a set of tests to the pattern of answers given to the Selector by the Scanner. The flow chart for this procedure is given in Fig. 3, and essentially consists of applying different sets of tests depending on the type of pattern derived by the Scanner. Thus, the Selector chooses the appropriate investment policy by correctly identifying the pattern of answers that is presented to it.

The Selection of a Portfolio

To facilitate the explanation of the selection procedures it is worthwhile interrupting the discussion for a moment to outline the information that is on hand prior to choosing a set of stocks for a particular portfolio.

(*a*) A list of stocks, the "A" List, which contains those stocks that are judged to be suitable for current investment. These stocks are categorized by industry.

(*b*) For each company on the "A" List there is a Relative Value List, a set of Relative Performance Lists, as well as historical, current, and forecasted information on sales, earnings, dividend yield, and other attributes.

(*c*) A list of information on the client for whom this portfolio is to be selected. This list includes the information discussed in the second section as well as an attribute that records the sum of money which is to be invested in common stocks.

(*d*) An investment policy that was chosen for this client as outlined above.

Given this information, the selection of a portfolio is essentially a process of mapping the set of industries and companies in (*a*) onto the investment policy in (*d*). This process yields a subset of industries and their respective companies that is reduced to a particular set of stocks for a portfolio by the addition of the information in (*b*) and (*c*), and the application of a set of tests based on this information. The actual processes governing this selection procedure are as follows:

3A. SELECTION OF INDUSTRIES APPROPRIATE TO THE INVESTMENT POLICY

Despite the large overlap between the characteristics of various industries, the investment officer associates a set of industries with each goal. As this association depends on the characteristics of the goal as well as the general characteristics of the companies within each industry, the particular set of industries associated with a given goal may include some of the industries which are associated with other goals. For example, some industries contain companies which vary only slightly in their individual characteristics, *e.g.*, banks, or utilities, while others, like oils, are more heterogeneous and appear on several lists. As the investment officer's classification of an industry's characteristics change very slowly with time, no attempt was made to determine how these attitudes and associations were developed. Instead, these lists were derived by direct questioning and examination of the investment officer's behavior. The model, then, takes these lists as given and by searching through the "A" List derives, for each goal, a list of those industries and companies that are on the "A" List. Thus, for each goal there is now a list of industries whose companies are both currently acceptable as well as suited to the investment performance desired from the portfolio.

3B. SELECTION OF COMPANIES

Once the list of industries has been generated, the companies on this list are selected for participation by the application of still another Scanner-Selector mechanism.

In this case the Scanner and the Selector have two separate functions. The first is to check the list of information on the client and see if the trust is a legal trust and/or whether the client is a resident of Pennsylvania [attributes (ix) and (vii)]. If either or both are the case the Selector applies one or both of the following two tests:

(i) If the trust is a true fiduciary relation all the companies on the given list that do not have legal status in Pennsylvania are rejected.

(ii) If the client is a resident of Pennsylvania, all the companies that are subject to property tax in Pennsylvania are rejected.

Having eliminated all companies that do not meet the only two absolute criteria the model then takes the remaining list of companies and applies to it the set of tests that are associated with each investment policy.

The Scanner performs the task of ordering the companies in each industry on the basis of the dominant attribute of the investment policy. For example, if an Income Portfolio was being selected the Scanner would rank order the companies in each industry on the basis of yield. The Selector takes the first company from the industry that is at the head of this list and applies a set of tests to it.

The tests consist of a series of binary decisions on the performance and expectations of important attributes. As the importance of particular attributes depends on the investment policy that is being applied, the series of tests varies with each investment goal.

The set of tests is qualitative in nature and is applied, in turn, to the companies within each industry. Unless the value of some attribute is very much out of line with what it should be, the Selector will accept the first company that is processed. If for some reason the first company does not pass the tests, the Selector moves on to the second company and repeats the process. If no company from that industry is able to pass through the set of tests, the Selector moves on to the next industry. If after processing all the industries funds remain to be invested, the Selector returns to the first industry from which no selection was made and recommences processing. This time processing begins at that test that immediately proceeds the spot where the Selector stopped on the first run through. As soon as a company is selected the Scanner and Selector move on to the next industry.

To further clarify this process, consider the set of tests which the Selector applies in order to choose growth portfolios (see Fig. 4). As can be seen from the flow chart, the tests are grouped in hierarchies. Thus, if Company A passes Test 3 it will go directly to Test 5. But, if it does not pass Test 5, it must pass Tests 6, 7, and 8, before it can be accepted back into the mainstream of tests. If no company from a particular industry succeeds in being accepted, and the Selector returns to it in order

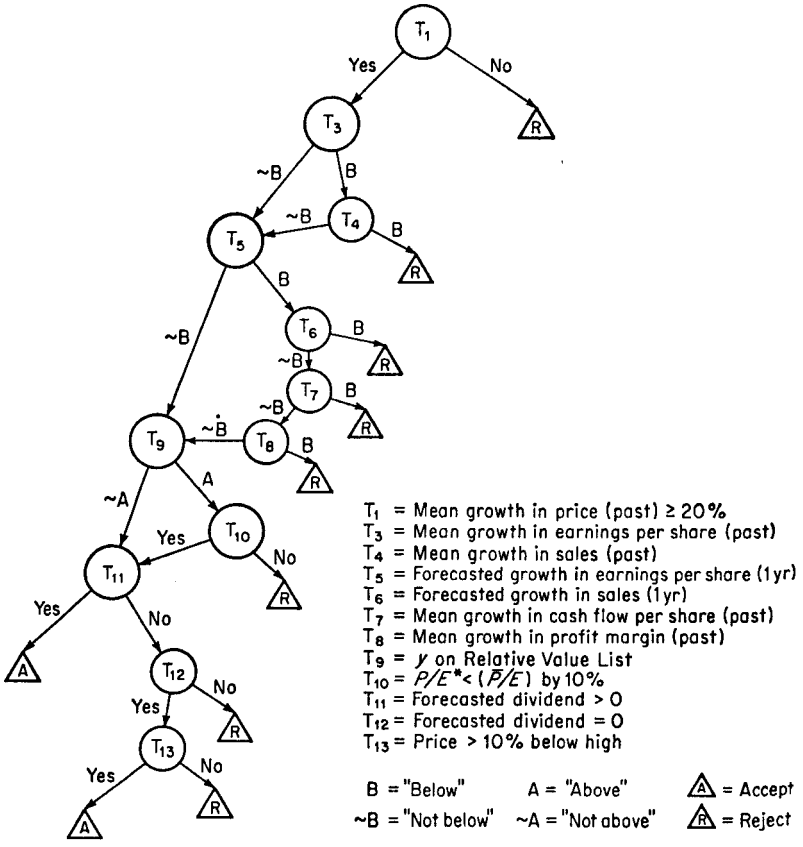


Figure 4. Growth-portfolio discrimination net.

to recommence testing, then this testing would occur in the following way. If company A was first rejected at Test 6, the Selector would now begin testing at Test 7. In this particular case, testing might continue until a company was selected. However, as each Discrimination Net has a test that all participations must meet, it is entirely possible for the model to reject all companies within a given industry.

3C. DIVERSIFICATION

Diversification is achieved by insisting that all accounts participate in at least *five* industries, and that participation in stocks be limited, in general, to *one* per industry. When the portfolio includes bonds and preferred stocks, each $10,000 invested in bonds or preferreds is taken to be equivalent to a participation in one industry. Hence, for an account of $50,000

with $20,000 invested in government bonds, the model would require that the remaining funds be invested in at least three industries.[18]

3D. SIZE OF PARTICIPATION

The number of shares to be purchased of each company that is selected for participation is determined by the "Share Selector." The essence of this process is given by the following rules:

(1) The total funds to be invested in common stocks are divided by the number of participations desired.[19] This produces the average number of dollars to be invested in each company.

(2) To determine the number of shares to be purchased, the average number of dollars to be invested in each company is divided by the price of the particular company's stock. This figure is always rounded to the nearest multiple of five, and whenever the funds available for each participation permit it, round lots, *e.g.*, 100 shares, are purchased.

Clearly, this selection process can only continue as long as there are funds remaining for investment. When the funds have been used up, the selection process stops, and the stocks that have been chosen become the required portfolio.

Testing the Model

In order to test the model's ability to reproduce the behavior of the trust investor—*i.e.*, to simulate the trust investment process—the model was required to select portfolios for a particular set of actual trust accounts. In particular, stock-exchange and other data were fed into the computer to cover the first and third quarters of 1960. The running program was then presented with data on four of the bank's new clients, for whom the trust investor had selected portfolios during the same two quarters, and the program was required to generate its portfolios for these accounts. The portfolios are presented in Figs. 5 and 6, along with the selections made by the trust officer for the same accounts. The generated portfolios were then compared with other portfolios generated by various random and naïve models. The results of these tests indicate that the trust

[18] As can be seen from the above, the investment officer's "rule of thumb" seeks to spread risk by diversification. But as Markowitz has shown (H. Markowitz, *Portfolio Selection*, p. 109, New York, 1959) when the returns on securities are correlated, this may not be accomplished if the amount invested for the client is relatively small.

[19] For accounts of $50,000 or less the usual number of participations is five, each $10,000 of bonds and preferreds counting as one. For accounts greater than $50,000 the minimum number is usually five as approximately $10,000 is invested in each participation.

Simulation of Account 1, 1/8/60
Growth Account
Funds available for investment: $22,000

The *program* selected:	The *Trust Officer* selected:
60 General American Transportation	30 Corning Glass
50 Dow Chemical	50 Dow Chemical
10 I.B.M.	10 I.B.M.
60 Merck and Company	50 Merck and Company
45 Owens Corning Fiberglass	50 Owens Corning Fiberglass

Simulation of Account 2, 6/10/60
Income and Growth Account
Funds available for investment: $37,500

The *program* selected:	The *Trust Officer* selected:
100 American Can Co.	100 American Can Co.
100 Continental Insurance	100 Continental Insurance
100 Equitable Gas Co.	100 Equitable Gas Co.
100 Duquesne Light Co.	100 General Public Utilities
100 Libbey Owens Ford	100 Libbey Owens Ford
100 International Harvester	50 National Lead
100 Philadelphia Electric	100 Philadelphia Electric
100 Phillips Petroleum	100 Phillips Petroleum
100 Socony Mobil	100 Socony Mobil

Figure 5. Comparison of portfolios selected by the model and by a trust officer: Accounts 1 and 2.

Simulation of Account 3, 7/8/60
Income and Growth Account
Funds available for investment: $31,000

The *program* selected:	The *Trust Officer* selected:
100 American Can Co.	100 American Can Co.
100 Continental Insurance	100 Continental Insurance
100 Duquesne Light	100 Duquesne Light
100 Equitable Gas	100 Equitable Gas
100 Pennsylvania Power and Light	100 General Public Utilities
100 International Harvester	100 International Harvester
100 Libbey Owens Ford	100 Libbey Owens Ford
100 Socony Mobil Oil	100 Socony Mobil Oil

Simulation of Account 4, 8/26/60
Income Account
Funds available for investment: $28,000

The *program* selected:	The *Trust Officer* selected:
100 American Can Co.	100 American Can Co.
100 Continental Insurance	100 Continental Insurance
100 Duquesne Light	100 Duquesne Light
100 Equitable Gas	100 Equitable Gas
100 Pennsylvania Power and Light	100 General Public Utilities
100 International Harvester	100 International Harvester
100 Phillips Petroleum	100 Phillips Petroleum

Figure 6. Comparison of portfolios selected by the model and by a trust officer: Accounts 3 and 4.

investment program selected a greater proportion of correct securities than did any one of the alternative models.

To obtain additional confirmation, the testing process was carried one step further—that is, the processes by which the portfolios are generated were submitted to empirical test. The test consisted of comparing the stream of output of the trust investment model to the recorded decision behavior of the trust investor. This test was applied to several of the mechanisms incorporated in the model. While it is not possible to state that all the processes were unequivocally confirmed, the evidence strongly supports the hypothesis that the model's mechanisms capture a considerable portion of the trust investment process.

section 4

Social Behavior

In recent years much empirical work has been done in the field of social behavior, particularly in the area of small group experiments. This work has provided us with much information about the effects of a wide range of variables on group behavior.

More recently, work has been done on models of group behavior. This work has been in part based on the empirical studies, but it has also contributed to the direction of the empirical work. The models have been formulated in ordinary verbal terms, and more recently in mathematical forms—differential equations and Markov processes.

The empirical work and the mathematical models have been very interesting and stimulating. However, there have been many complaints about the simplicity and paucity of the mathematical models. One reply to this sort of criticism has been the creation of computer models.

The charge to students of social behavior to look to the computer model as a useful technique was given by Bales in his paper on "Small Group Theory and Research" (1959). Bales took as the sociologist's goal the prediction of behavior in natural settings as opposed to the prediction in highly controlled laboratory situations. To accomplish this goal, Bales believes that what is required is a synthesis of "large numbers of variables in highly complex conditional relationships to one another." And he goes on to say that the computer is the tool that can aid in the attainment of such a model.

This charge has been taken up by Bales and his associates, who are pursuing a research program directed toward developing com-

puter models of group behavior (Bales, 1959); by McPhee, who has developed a computer model of voting behavior (1961); and by John and Jeanne Gullahorn, who have developed a computer model of social interaction. The following article by the Gullahorns is a report of their work prepared especially for this collection.

John Gullahorn is a member of the faculty of the Department of Sociology and Anthropology at Michigan State University. Jeanne Gullahorn is a graduate student at the same institution. They are also consultants to the Artificial Intelligence Section, System Development Corporation.

A COMPUTER MODEL OF ELEMENTARY SOCIAL BEHAVIOR

by John T. Gullahorn & Jeanne E. Gullahorn

Ten years ago the social psychologist Solomon Asch observed, "To act in the social field requires a knowledge of social facts—of persons and groups. To take our place with others we must perceive each other's existence and reach a measure of comprehension of one another's needs, emotions and thoughts" (1952, p. 139). In recent years the traditions of psychoanalysis, field theory, and symbolic interaction have generated many insightful explorations of how individuals perceive and cognize the human environment. A radical departure from these relatively intuitive approaches has recently appeared in George Homans' *Social Behavior,* which incorporates principles from two self-consciously rigorous disciplines—classical economics and behavioral psychology (1961). In our opinion the work represents one of the most provocative explanations of human response in interpersonal situations yet published, and we have selected Homans' treatise as a model for research concerning individual reactions in relatively simple social interaction. At present our efforts are directed primarily toward building and refining a statement of the model of elementary social behavior in the form of a computer program written in Information Processing Language (Newell, 1961*e*). In addition to enhancing the clarity and precision of the model, we hope such a representation will ultimately contribute to the goal of naturalistic prediction of behavior in small groups.

Before proceeding to a discussion of the program itself let us consider briefly Homans' treatment of elementary social behavior, that is, of "face-to-face contact between individuals, in which the reward each gets from the behavior of the others is relatively direct and immediate" (1961, p. 7). His model envisages human behavior as a function of its payoff; in amount

and kind, an individual's responses depend on the amount and quality of reward and punishment his actions elicit.

To illustrate the application of the propositions he advances to explain social exchange, Homans uses Blau's description of interpersonal behavior in a bureaucracy (1955). Sixteen agents holding the same title were employed in this federal office. The men varied in competence, and as expected the more skilled received more requests for assistance from their co-workers. In analyzing the social economics of such consultations, Blau and Homans regard the interaction as an exchange of values: both participants benefited, but both had to pay a price. The agent requesting help usually was rewarded by being enabled to do a better job; however, he paid the cost of implicitly admitting his inferiority to a colleague who by title was supposedly his equal. The consultant, on the other hand, gained prestige; however, he incurred the cost of time taken from his own work.

We have used this relatively simple interaction sequence between two hypothetical agents, whom we have named Ted and George, to begin actualizing in a computer program the dynamic implications of Homans' explanatory propositions as they relate to the decision processes of individuals involved in social exchange. The program, entitled HOMUNCULUS, now is running for interactions between two persons; but we are still in the stage of writing additional routines to introduce refinements into the basic model. The simulation appears to have verisimilitude, but its verity has not yet been tested against actual social interaction.

The Program

In planning the program of our model we first had to make explicit our conception of a person as an information processing organism. That is, in order to behave according to the principles set forth in Homans' explanatory propositions a person must be "programmed" to do at least the following: He must be able to receive stimuli, recognize stimuli, store stimuli in memory, and compare and contrast stimuli; he must be able to emit activities, differentiate reward and punishment, associate a stimulus situation with a response, and associate a response with a reinforcement; and, on the basis of past experience, he must be able to predict the probability of reward resulting from each response he contemplates. In social situations he must be able to differentiate among other members of a group, evaluate a social stimulus in terms of the specific person emitting it, and select his response accordingly so as to elicit a positive reaction in turn.

Once we had outlined some of the basic qualities necessary for a programmed social being, we then faced the practical problem of how to get

such a creature into the computer. Fortunately, IPL-V is ideally suited to the solution of this task. Since it is a list processing language, both data and routines are written in the form of lists. Very complex information can be handled efficiently through the use of list structures, or hierarchies of lists containing as many sublists as desirable, and of description lists which associate with any symbol a list of its attributes and their values. A person thus is represented in our program as a list structure containing a large number of description lists. Among the data included in the list structure of a person are such items as his identity, his abilities, his relative and absolute positions in various social groups, his image lists of his reference groups, and his image lists of other group members.

The flow chart depicted in Figs. 1 through 3 represents our interpretation of the processes involved in operationalizing Homans' propositions for elementary social behavior. The interaction sequence we have programmed begins with an agent, Ted, emitting to his colleague, George, a symbol which represents a request for help in completing a job assignment. Let us postpone discussion of Proposition 5 (P5, Box I in the flow diagram) and begin with Proposition 1 (P1, Box IV) and consider only the positive branches in the diagram so that each proposition can be described briefly in sequence. Our programmed statement of Homans' propositions specifies the symbol manipulating processes which enable George to decide what action he will emit in response to Ted's request.

Proposition 1

Homans' first explanatory proposition concerns the influence of stimulus and response generalization:

> *If in the recent past the occurrence of a particular stimulus-situation has been the occasion on which a man's activity has been rewarded, then the more similar the present stimulus-situation is to the past one, the more likely he is to emit the activity, or some similar activity, now* (1961, p. 53).

In translating the proposition into computer routines enacting decision processes, we found it necessary to consider in sequence two aspects of the "stimulus-situation." To begin with we hypothesized that our agent, George, would react in a relatively global manner to the general situation itself. Thus our interpretation of the initial information processing implied by Proposition 1 (see P1, Box IV in Fig. 1) involves George's considering whether AR (the activity received—in this example, a request for help) is a general stimulus situation in which his responses (AE's, or activities emitted) have been rewarded. In executing this process one routine representing a retrieval function of our programmed agent (George) searches

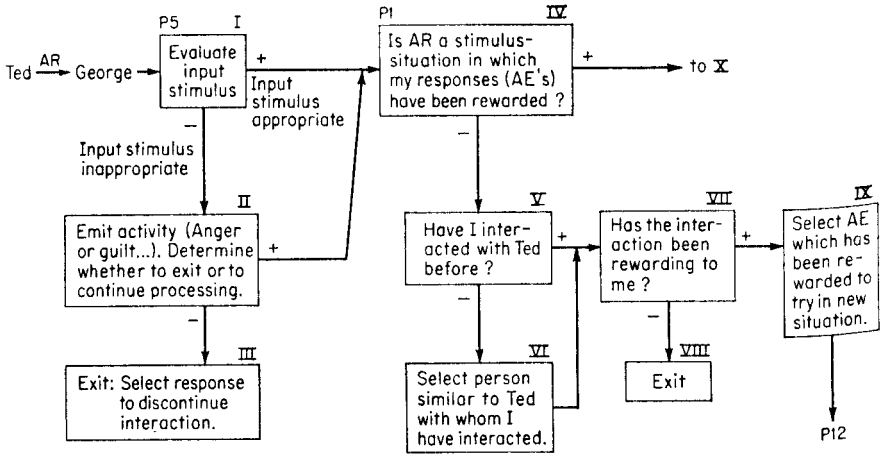


Figure 1.

a memory list of reinforced stimulus situations to determine whether the present input is among them.

Taking the positive branch of the flow diagram and thus assuming that George has found that responding to a request for help has led to reward in the past, let us proceed to his next consideration, depicted in Box X of the flow chart in Fig. 2. George now must determine whether his responses to a request for help have been rewarded by Ted, the person currently introducing the situation.

In order to check on past interactions with Ted, George must search deeper into his memory structure. We noted that the list structure of an individual includes a list in which he stores his image of every person within the group. One routine locates the image list, finds the sublist on it which describes George's previous interactions with Ted, determines whether he has received the present stimulus from Ted before, and if so whether his repsonses to it have generally been rewarded by Ted. In the case we are discussing, George discovers that in the past his responses to requests for assistance have been reinforced by Ted.

Having determined that the stimulus situation has been a rewarding occasion and that Ted has been an agent of reinforcement, George now must consider response alternatives. If he has interacted in similar situations with Ted and has emitted several different activities which Ted has rewarded (*e.g.*, solved Ted's problem, referred Ted to a helpful source or to an expert on the problem in question), then George must choose among these possible reactions to Ted's request. In our program George selects up to three activities from a memory list of responses Ted has rewarded

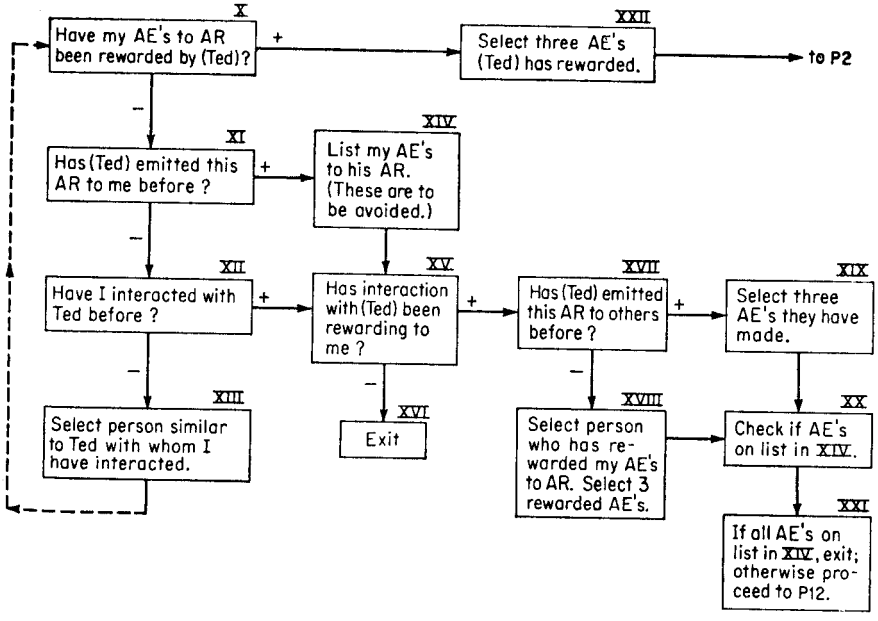


Figure 2.

(Box XXII, Fig. 2) and then proceeds to process further information regarding these contemplated activities.

Proposition 2

Homans' second proposition deals with the positive influence of the frequency and recency of reinforcement:

> *The more often within a given period of time a man's activity rewards the activity of another, the more often the other will emit the activity* (1961, p. 54).

Reformulating this proposition for computer simulation posed a number of problems. It would have been relatively simple merely to set a counter for each reinforced response and then retrieve the desired information regarding reward frequency. However, we felt this procedure would not adequately simulate human information processing systems. Of course, people do avail themselves of precise measurement scales and use various cultural artifacts—such as computers—to increase their accuracy. But in making estimates concerning frequencies and values of rewards ensuing from everyday social interaction, people seem to use a less refined means of measurement. In programming this proposition, therefore, we devised a rather crude five-point ordinal scale for reward frequency, ranging from an estimate that a response was "nearly always rewarded," through a judg-

ment that it was "rewarded about half the time," to an assessment that it was "almost never rewarded."

At present we are experimenting with different means of manipulating this scale. One routine we have written increases the ordinal scale value for the reward frequency after three reinforcements of the response. This procedure, however, is not completely satisfactory. Indeed, one may argue that estimates of reward frequency are not necessarily independent of the emotional salience of the reinforcement. When HOMUNCULUS has reached the stage of simulating small group behavior in controlled conditions, it should be possible to test various approximations of human judgments of reward frequencies from social interaction and to select the routines which simulate the actual behavior most accurately.

When the processing for this proposition is completed (P2, Box XXIII, Fig. 3), George has a rough estimate of the frequency with which Ted has rewarded each of the activities he is considering in response to Ted's current request for help. Homans' Proposition 2, taken alone, would lead to the expectation that George would then merely emit the most frequently rewarded response alternative. But other information must be processed before a decision is reached.

Perhaps here we should indicate how the program keeps all this material in immediate memory for George. Up to one hundred named private storage cells are assigned for this purpose, and instructions in each routine specify which cells it is to use for storing its findings. At present George is using about fifty of these cells. In addition, important information available to all group participants—for example, what could be seen and heard during the last five interactions—is kept in named public storage cells.

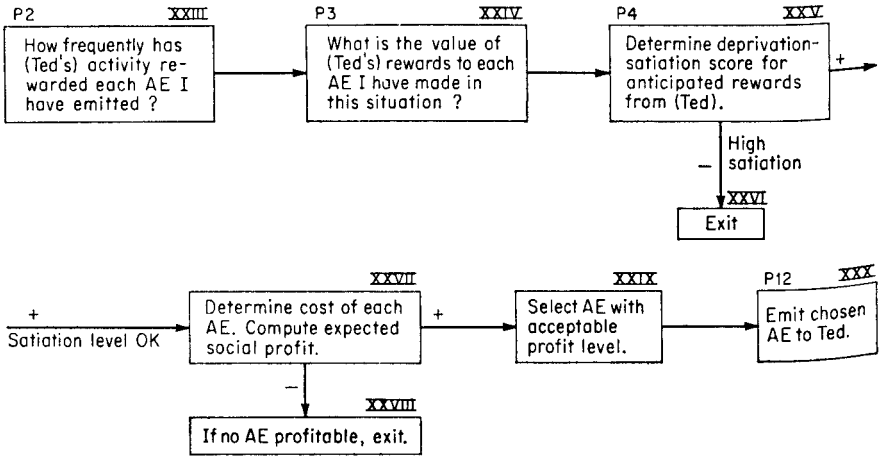


Figure 3.

Proposition 3

Among the other relevant factors that must be considered in selecting an activity to emit is the value of the anticipated reward. Homans' third general position states.

> *The more valuable to a man a unit of the activity another gives him, the more often he will emit activity rewarded by the activity of the other* (1961, p. 55).

Assessing the value of an activity is somewhat more complicated than estimating the frequency with which it occurs. Value has two components —one relatively constant and the other, which we shall discuss in Proposition 4, relatively variable for the periods of time involved in the simple interactions comprising elementary social behavior. The value component referred to in Proposition 3 concerns an individual's rank ordering of the subjective reward attendant on receiving one activity rather than another. With reference to our example, we might predict that George would find warm social approval involving Ted's complimenting him in front of colleagues to be more "valuable" than a halfhearted response of "Hmm, thanks," or an annoyed retort, "Well, sorry I bothered you."

At this point in our program, therefore, we have what game programmers term a "look-ahead." In considering Ted's request, George has "in mind" three responses he recalls Ted's having rewarded in the past, and he has estimated the frequency with which Ted has reinforced each response. Now he must consider more carefully the particular reward he expects Ted to give to each response so that he may determine the inherent worth of each anticipated reward (P3, Box XXIV, Fig. 3). Taking in turn each activity George is contemplating, the routines executing this proposition retrieve the responses Ted previously has made to each, determine which one he is likely to emit now, and search description lists to find the subjective value of the reward for George.

Proposition 4

Homans' fourth proposition deals with the other component of value—the deprivation-satiation aspect, or the marginal utility of a given unit of activity.

> *The more often a man has in the recent past received a rewarding activity from another, the less valuable any further unit of that activity becomes to him* (1961, p. 55).

In contrast to the relatively constant intrinsic satisfaction aspect of value, the deprivation-satiation component varies over a range of possible rankings. Taking into account the amount of an activity a person has received,

we note that he "values" that activity more when he has been deprived of it than he does when he is in a state of relative gratification. Thus, while social approval may be highly rewarding to an individual, if in the recent past he has received a great deal of this generalized reinforcer then he is not likely to be so interested at the moment in receiving more.

In processing the information necessary for completion of this stage of the program, George must evaluate his relative deprivation with reference to the rewards he anticipates from Ted. George now has in immediately available memory a record of each activity he is contemplating, and stored with each activity is various information about it, including the response he expects Ted to make to it. The routines which execute Proposition 4 search the description lists of each of the anticipated rewards to determine the degree of George's current deprivation or satiation with respect to them. A deprivation-satiation score based on a simple ordinal scale is stored as the value of a special attribute on the description list of each activity. In executing Proposition 5, which we shall discuss later, routines update the deprivation-satiation score whenever an activity is received.

With the information retrieved thus far George has an estimate of the relative frequency with which Ted has rewarded each activity he is considering emitting. Furthermore, he has predicted Ted's reaction to each of the projected actions and has determined how rewarding each of these anticipated reactions is to him, personally, as well as how deprived or satiated he currently feels with respect to each of these expected rewards. At this point, therefore, George can rank his contemplated responses in terms of their expected payoff. But he is not yet ready to emit the highest ranked action.

Another important consideration is the cost of the proposed response. Homans defines the cost of an activity as the value of the reward obtainable through an alternative activity, forgone in emitting the given one. In our example, George must forgo working on his own assignment if he takes time to assist Ted; therefore George must determine the relative reward value of this alternative activity. To do this he follows a procedure analogous to that just described, processing information concerning the frequency of past reinforcement and the value of the anticipated reward ensuing from this activity as well as his relative satiation with the reward. Then he can compare the over-all expected reward from his contemplated response to Ted with the anticipated reward from continuing with his own work, and he can compute what Homans terms the psychic profit—the reward of an activity less its cost.

Let us suppose George tentatively is planning to give Ted direct assistance on his problem because in the past Ted has praised him for this activity, and social approval is a reinforcement George values highly and

one for which he feels relative deprivation at present. But let us also suppose that George has an important assignment to complete, and that taking time from it might detract from the quality of his work and thus lessen the approval he anticipates from his boss for a good job. In this case George would incur a loss rather than a profit in helping Ted directly; therefore he will continue processing to see whether one of the other activities he was contemplating might yield a profit. In this illustration, George probably will decide that referring Ted to another source will net him a profit, since he expects some approval for this activity (albeit less than he would get from directly assisting Ted), and he will incur a very small cost in terms of time taken from his own work.

Having selected what he expects to be a socially profitable activity, George emits that response to Ted. At this point our program cycles, and the activity George has emitted becomes the activity Ted has received. Now Ted must process information in order to select an appropriate and profitable response to George.

Proposition 5

Distributive justice, the subject of Homan's fifth proposition, is perhaps the most complex of the concepts involved in the explanation of elementary social behavior. At the very least it requires consideration of information at another level—that of social norms or accepted expectations for behavior within a group. Through repetition of interaction situations within a group, certain behavior patterns become stabilized so that expectations develop regarding what constitutes justice in the distribution of rewards and costs between persons. The greater a man's costs in a given interaction, the greater his rewards ought to be. But the implications of distributive justice go even further, taking into account a person's investments in an interaction—for example, his seniority, skill, experience, age, and sex. The greater the investment a person makes in an interaction, the greater the net profit he has a right to expect. Thus according to the principle of distributive justice it is consensually expected that certain antecedent costs and investments should have as consequents certain types and degrees of reinforcement.

Homans states the related proposition as follows:

> *The more to a man's disadvantage the rule of distributive justice fails of realization, the more likely he is to display the emotional behavior we call anger* (1961, p. 75).

More is included, however, for if a man receives rewards beyond those to which he considers himself entitled, he is likely to experience guilt feelings.

Translating this proposition into computer routines posed some of the

most interesting problems we have yet encountered in working with HOMUNCULUS. In effect, the list structures of our agents had to be programmed to have consciences, and they had to include a repertoire of appropriate anger responses.

In essence, our programmed interpretation of this proposition asks whether a stimulus activity is appropriate in the given circumstances (P5, Box I, Fig. 1). If so, then the person receiving it can process it as George did Ted's request, which he considered appropriate. If, however, the stimulus activity is judged inappropriate, then more complex behavior results. To illustrate this let us shift to a description of the interactions between George and Tom, another worker in the same agency.

It is an accepted office norm that a worker who asks for help should do so openly in a manner acknowledging the superiority of his consultant with respect to the given problem. Tom, however, has been seeking aid from George in a rather devious manner, coming to George with "an interesting problem" and saying he would like to see whether George arrives at the same solution as he. This has occurred three times in the recent past, and on each occasion, Tom has greeted George's suggested solution with the comment, "Yes, you reached the same conclusions I did." George decides Tom is violating the norms of fair exchange by evading the cost of thanking him for his assistance and conceding his superiority. The fourth time Tom presents him with an interesting problem George angrily responds, "Look, why don't you do your own work!"

This description, of course, does not answer the question of how the computer is programmed to behave in such an all-too-human way. George is programmed to treat time spent solving a problem presented by another worker as being help to that person for which recognition and social approval are due. When his colleague responds to his efforts with an unrewarding confirmation that he arrived at the same conclusion, George finds this input inappropriate in terms of his expectations regarding distributive justice. Therefore, routines processing Proposition 5 change George's image list of Tom so that next time he expects greater recognition and thanks than normal to atone for the present evasion. After three repetitions of this interaction sequence the discrepancy between Tom's behavior and George's expectations will be so great that when George evaluates Tom's response he will plant a signal in his image list of Tom indicating that interacting with him is not rewarding because Tom violates group norms.

The next time Tom asks for an opinion after this warning signal has been set, George will respond by displaying anger or by storing up aggression to be expressed against someone else. In the computer program an anger response involves emitting behavior punitive to another person. But before actively punishing Tom, George will first assess the conse-

quences to himself of such behavior. In one possible interaction sequence, if George finds that Tom is in favor with George's own boss, he may suppress his aggression at the moment and then release it the next time he interacts with a subordinate.

The routines processing the negative branch of Proposition 5 (Box II, Fig. 1), thus not only modify image lists but also use some of the routines from the other propositions to evaluate the probable consequences of direct anger responses. Depending on the outcome of this processing, the program either proceeds to Proposition 1 or the interaction is terminated.

Conclusions

Like other behavioral scientists who are expressing their theories in IPL-V in order to learn about human processes by simulating them on a digital computer, we are reducing complex social behavior to symbol-manipulating processes. Even in this brief outline of our program it should be obvious that we, too, have found IPL-V particularly appropriate for operationalizing our model. We have already noted the flexibility afforded by organizing information in lists and list structures and the elegant simplicity yet powerful efficiency provided by utilizing description lists for storing information concerning certain symbols. In addition, the relative ease of organizing routines in hierarchial structures greatly facilitates the sequential processing of information involving numerous conditional subroutines. There has not been time to explore the negative branches of our flow diagram except for Proposition 5; however, the general processes should be apparent.

In contrast to the more purely cognitive models of behavior, *e.g.*, Feigenbaum (1959) and Feldman (1959), our model focuses on individual decision-making in social interaction where normative considerations must be processed in the interplay of reciprocal rewards and punishments. HOMUNCULUS is neither a completely general model, like Feigenbaum's Elementary Perceiver and Memorizer, nor is it specific to particular subjects, like Feldman's simulation of binary choice behavior. Rather, the program blends the two approaches. The information processing involved in the routines is common for all simulated subjects; however, each list structure describing a group participant is highly specific; consequently, individual idiosyncracies and recent past histories determine whether certain subroutines will be executed and what their outputs will be in specific interaction sequences.

With reference to underlying assumptions, however, our model shares several characteristics with those programmed by Feigenbaum and Feldman. Like their models, HOMUNCULUS emphasizes nonnumerical proc-

esses and is essentially deterministic rather than probabilistic. In addition, the decision-making processing is assumed to be serial—that is, we consider a person capable of doing only a limited number of things at one time. Furthermore, our model conceives of a person as an hypothesis testing, information processing organism capable of receiving, analyzing, reconstructing, and storing information. HOMUNCULUS is an attempt to explicate in a way not possible with verbal theory the ability of a person engaged in normal social interaction to evaluate the context of behavior, retrieve information necessary to project alternative plans of action, and—before actually committing himself overtly—to select the conditions under which he will emit one activity rather than another.