PROGRAMMING A MODEL OF HUMAN CONCEPT FORMULATION

Earl B. Hunt and Carl I. Hovland
Yale University

Summary

A model of human information processing during concept formation has been constructed, using a list processing, digital computer program. The program's input consists of descriptions of objects in terms of dimensions and values. The universe of objects is divided into two or more sets. The program attempts to form a decision rule, based upon the descriptions of the objects, which can be used to assign any previously presented or new object to its correct set.

The program is a model for human information processing, rather than an artificial intelligence system. It contains features which limit the number of objects in internal memory and the number of dimensions which may be involved in an answer. Using this program, simulations have been performed of a number of psychological experiments in concept learning. Comparison of these simulations with the data obtained from human subjects will be discussed.

What is a concept? Ordinary usage is not precise. The English "the concept of force", "the concept of massive retaliation," and the "concept of dogs" are all permissible. Church[3] 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 stimulus 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 paper 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, Simon, and Shaw (cf. Green[5]). Partly for local administrative reasons, we are in the process of converting our programs to LISP, a list processing language developed by McCarthy[14]. 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[6]). Dimensions are also organized into "dimension sets", or groupings. These groupings represent subsets of the sets 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 Figure 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 they can always re-examine 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 instance 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[8]). 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 Figure 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 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 <u>execution list</u> containing the names of three routines which will be executed in the order specified by the <u>execution list</u>. The contents of internal memory is 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 <u>ordering routines</u>. 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 copys 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, re-categorized and re-described. 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 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 succeeds, 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 constitutes 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 a nth 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 Figure 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 re-constructed. When this happens a new ordering routine is selected. 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 re-arranged) 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 re-arrangement 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 unravelling 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 execution 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 Evaluation

The model was not conceived *in vacuo*. Previous, unprogrammed models[7] 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[9] 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 overall "postdiction" of responses 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 pre-specified 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[21]. 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 discrepance 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 connective between two elements provided the answer. A "truth table" was constructed and presented to subjects in geometric form. For example, the connective "p <u>and</u> 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 <u>some</u> 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 phenomenon 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 have been considered. One system retains and extends the limited occupancy model. Instead of storing one "codeword" (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 codeword would be independent on each occupancy list. Codewords 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 codewords independently on each occupancy list, some of the codewords 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 "associations" 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[17,18]. There is also some similarity between this approach and the classification mechanisms based upon Selfridge's "Pandemonium" scheme[19]. 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[20,21] 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 re-transformed 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 $\underline{t}$ 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[12,13] 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 ordinal re-arrangements is that they do 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 relevant 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[15] uses intermediate calculations heavily. Amarel[1] 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[2] has presented an analogue, 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[10] 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 pre-code 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[6] 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[19], and by Rosenblatt[17,18], 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. Pattern recognizers are inspired by the capability of biological systems to amplify upon their sensory inputs. 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[11] and Amarel[1]. 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[16] postulates for plausible reasoning. One of Kochen's 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[1] 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[14] and his associates at M.I.T. In particular, the work of Feigenbaum[4], at Carnegie, 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.

References

1. Amarel, S. An approach to automatic theory formation. Paper presented at the Illinois Symposium on the Principles of Self Organization, 1960.

2. Bartlett, F. C. Thinking. New York: Basic Books, 1958.

3. Church, A. Introduction to mathematical logic, vol. I. Princeton, N.J.: Princeton U. Press, 1956.

4. Feigenbaum, E. An information processing theory of verbal learning. RAND Corp. publication, p. 1817, 1959.

5. Green, B. F. IPL-V, the Newell-Shaw-Simon programming language. Behavioral Science, 1960, 5, #1.

6. Hovland, C. I. A "communication analysis" of concept learning. Psychol. Rev., 1952, 59, 461-472.

7. Hovland, C. I. and Hunt, E. B. The computer simulation of concept attainment. Behavioral Science, 1960, 5, 265-267.

8. Hunt, E. B. An experimental analysis and computer simulation of the role of memory in concept learning. Unpubl. Ph.D. dissertation, Yale U., 1960.

9. Hunt, E. B. and Hovland, C. I. Order of consideration of different types of concepts. J. exp. Psychol., 1960, 59, 220-225.

10. Keller, H. Finite automata, pattern recognition, and perceptrons. AEC Computing Center and Applied Mathematics Center, Report NYO-2884, 1960.

11. Kochen, M. Experimental study of hypothesis formulation by computer. IBM Research report, RC-294. IBM Research Center, Yorktown Heights, New York, 1960.

12. Lawrence, D. H. and Coles, G. R. Accuracy of recognition with alternatives before and after the stimulus. J. exp. Psychol., 1954, 47, 208-214.

13. Lawrence, D. H. and LaBerge, D. L. The relationship between recognition accuracy and order of reporting stimulus dimensions. J. exp. Psychol., 1956, 51, 12-18.

14. McCarthy, J. Recursive functions of symbolic expressions and their computation by machine. Communications of the Association for Computing Machinery, April, 1960.

15. Newell, A. and Shaw, J. C. Programming the logic theory machine. RAND Corp. publication, p. 954, 1957.

16. Polya, G. Mathematics and plausible reasoning. Princeton: Princeton U. Press, 1954.

17. Rosenblatt, F. The perceptron, a probabilistic model for information organization and storage in the brain. Psychol. Rev., 1958, 65, 368-408.

18. Rosenblatt, F. Perceptual generalization over transformation groups. In Self organizing systems, London, Pergamon Press, 1959.

19. Selfridge, O. and Neisser, U. Pattern recognition. Scientific American, 1960, 203, 60-79.

20. Shepard, R. N. and Chang, J. J. Stimulus generalization in the learning of classifications. Bell Telephone Lab. mimeographed report, 1961.

21. Shepard, R. N., Hovland, C. I. and Jenkins, H. M. Learning and memorization of classifications. Psychol. Monogr., 1961, in press.

Acknowledgment

The work reported in this paper was supported by a grant for the study of concept learning, from the Ford Foundation to Carl I. Hovland. The computational work involved was supported, for the most part, by the Computation Center, Massachusetts Institute of Technology. Dr. Bert F. Green, Jr., and Alice Wolf, of Lincoln Laboratory, M.I.T. made available the 709 version of the IPL-V interpreter and instructed the first author in its use. The aid received is greatfully acknowledged.

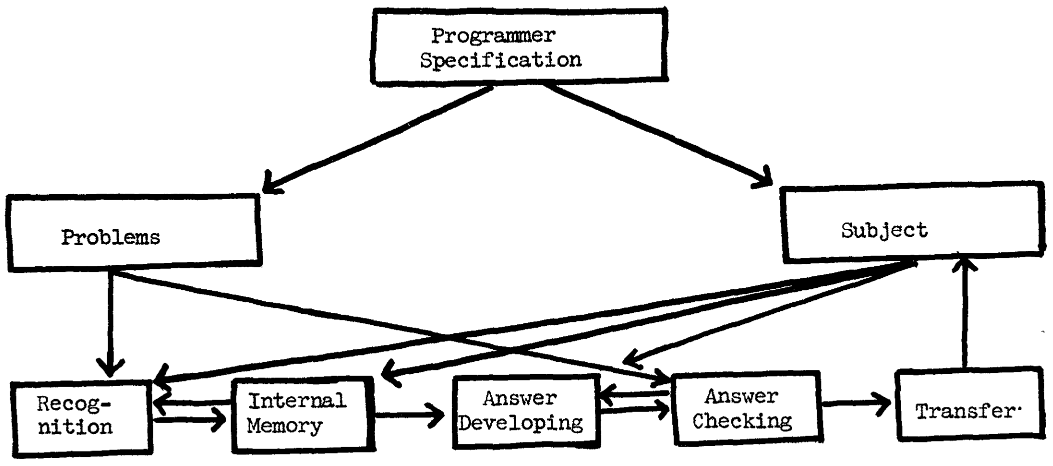

Figure 1. Program Control Chart.

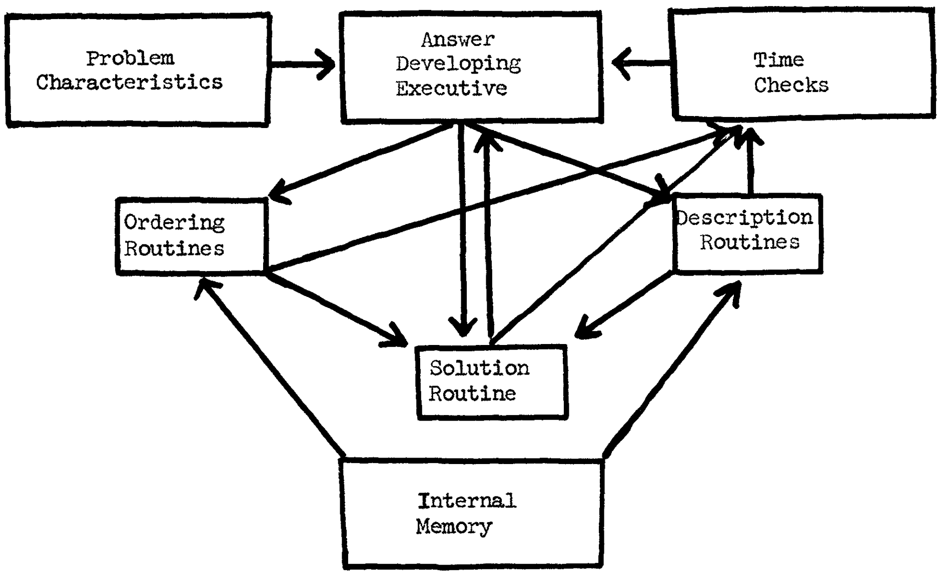

Figure 2. Answer Developing Procedure.