SESSION 1

PAPER 1

# SOME METHODS OF ARTIFICIAL INTELLIGENCE AND HEURISTIC PROGRAMMING

by

Dr. MARVIN L. MINSKY

## BIOGRAPHICAL NOTE

Marvin Lee Minsky was born in New York on 9th August, 1927. He received his B.A from Harvard in 1950 and Ph.D in Mathematics from Princeton in 1954. For the next three years he was a member of the Harvard University Society of Fellows, and in 1957-58 was staff member of the M.I.T. Lincoln Laboratories. At present he is Assistant Professor of Mathematics at M.I.T. where he is giving a course in Automata and Artificial Intelligence and is also staff member of the Research Laboratory of Electronics.

# SOME METHODS OF ARTIFICIAL INTELLIGENCE AND HEURISTIC PROGRAMMING

by

Dr. MARVIN L. MINSKY*

SUMMARY

THIS paper is an attempt to discuss and partially organize a number of ideas concerning the design or programming of machines to work on problems for which the designer does not have, in advance, practical methods of solution. Particular attention is given to processes involving pattern recognition, learning, planning ahead, and the use of analogies or "models". Also considered is the question of designing "administrative" procedures to manage the use of these other devices. The paper begins with a discussion of what is meant by "Intelligence" and concludes with a section concerned with some techniques through which a machine might further improve itself by adding to its collection of problem-solving methods.

## 1. INTELLIGENCE

I feel that it would not be useful to lay down any absolute definition of "intelligence" or of "intelligent behaviour". For our goals in trying to design "thinking machines" are constantly changing in relation to our ever-increasing resources in this area.

Certainly there are many kinds of performances which if exhibited by a man we would all agree, today, require or manifest intelligence. But would we agree tomorrow? For some purposes we might agree with Turing *(ref.24)*, to regard the same performances in a machine as intelligent. In so doing we would be tying the definition of intelligence to some particular concept of human behaviour.

While such a convention might be useful in some kinds of discourse, its use in serious analysis is precluded by two serious faults. First, it

* The work leading to this paper was supported, in too many ways to cite individually, by the joint services of the U.S.A.

directly evades any concise specification of the kinds of behaviour we are looking for. Second, we can often find simple machines which in certain situations do exhibit performances which would be called intelligent if done by a man.

We are, understandably, very reluctant to confer this dignity on an evidently simple machine. Hence the conflict one would suffer in using this definition would threaten any descriptive value it might otherwise have.

In what situations are we less reluctant to attribute intelligence to machines? Occasionally, a machine will seem to be more resourceful and effective than one might expect from casual inspection of its structure. We may be surprised and impressed and we tend to remain so until through analysis or "explanation" the sense of wonder is removed. Whenever a system behaves as though it had more resources than were evident at "first glance" we react in this way, and this reaction is closely related to that involved when we judge a performance to be an exhibition of intelligence. But clearly this reaction depends also on the resources of the individual who is making the observation. The behaviour of any machine (as we use the term) is always explicable in terms of its past states, external contingencies, and the causal or probabilistic relations between them. Hence the significance of the observer's surprise in this; it can be inferred that the observer is not so good a mathematician that his first glance constitutes an adequate analysis of the situation. In the same way, our judgements of intelligence on the part of other humans are often related to our own analytic inadequacies, and these judgements do shift with changes in understanding.

We frequently find that a skill which seemed highly intelligent in others becomes much less impressive when we have learned the trick of doing it for ourselves. Indeed, we refer to many very complicated procedures as matters of "skill" rather than of intelligence apparently just because there happens to be a known method of instruction through which the ability can usually be acquired.

In attempting to design intelligent machines we are, in effect, concerned with the problems of "creativity". Many people are hostile to such an investigation, maintaining that creativity (or intelligence) is some kind of "gift" which simply cannot be understood or mechanized. This view can be maintained only through a constant shifting of definition. As soon as any process or performance has been mechanized or otherwise explained, it must be removed, with qualifications and apologies, from the list of creative performances. This part is perfectly reasonable; once a process has been mechanized one no longer needs terms like "creative" for its description, and we, too, remove it from the list of things to be accomplished. The weakness of the advocate of inexplicable creativity lies in the unsupported conviction that after *all* machines have been examined some items will still remain on the list.

Let us put it clearly then, that in exploring what we call "the artificial intelligence problem" we are not looking for any kind of closed solution to any question like "what is intelligence and how can it be mechanized?". The judgement of intelligence is more a reflection on what we understand than on what we, or machines, can do. Instead, we are searching for new and better ways of achieving performances that command, at the moment, our respect. We are prepared for the experience of understanding and the consequent reshaping of our goals.

## 2. PROBLEM-SOLVING

How do humans solve problems? To begin with we have to replace our intuitive requirements by reasonably well-defined technical questions. This may require the largest part of the intellectual effort, but we cannot dwell on the subject. A minimal requirement is that one have a method of discerning a satisfactory solution should one appear. If we cannot do this then the problem must be replaced by one which is well-defined in that sense, and we must hope that solution of the substitute problem will turn out to be useful.

In the best case we come equipped with an efficient *algorithm:* a systematic procedure which, given the problem as input, is guaranteed to produce a solution as output; efficient in that the solution will arrive within reasonable bounds on time and effort. But for new and interesting problems we don't usually have algorithms, or at least not efficient ones. At the other extreme we may know nothing about how to get a solution (except for the ability to recognize one). In such a case we have no alternative save to launch into an exhaustive search through the ensemble of potential solutions, e.g., the set of all proper expressions in our language. Random search is no better in general than systematic exhaustion, and may introduce the possibility of failure. It is tempting but irrational to look for a panacea in chaos. But in any case it is well known that for interesting problems exhaustive search is usually out of the question, even with the aid of the most powerful conceivable machines.

Normally, we are not motivated to attempt such problems. "Interesting" problems always have roots in areas which are at least partially understood. We usually have a good deal of partial information about how to get a solution. But this information may occur in fragmentary form: we may have some information about the "form" of a solution, recollections of similar problems solved in the past, general suggestions, hints, and the like.

We need to find ways of writing programs which will be able to use these fragments, or general advice, to reduce the amount of search to reasonable proportions.

"Hints", "suggestions", or "rules of thumb", which only usually work are called *heuristics.* A program which works on such a basis is called a *heuristic program.* It is difficult to give a more precise definition of heuristic program - this is to be expected in the light of Turing's demonstration that there is no systematic procedure which can distinguish between algorithms (programs that always work) and programs that do not always work.

## 3. METHODS USED IN HEURISTIC PROGRAMS

Below are mentioned some of the ingredients that might go into a complex problem-solving program. Too little is known about heuristics to justify attempting a systematic or exhaustive catalogue.
Some of the methods are later discussed in further detail.

### 3.1. *Methods which set up searches*

It has often been convenient to imagine problem-solving as taking place over a tree-like structure of trial possibilities. The tree structure is particularly natural for game problems and for theorem-proving; while it may be less appropriate in other areas, it does seem to have general suggestive value. Usually the problem is not so much to find the basic structure (or the domain of things to try) as to find ways of reducing this structure to reasonable size. Thus, in chess, one may use an evaluation function technique to terminate lines of search, and some kind of "plausible move" selection technique to reduce the amount of branching at each node of the game tree. The problem of designing chess-playing programs has provided fertile ground for study of such methods; see, e.g., Shannon *(ref.22)*, Newell *(ref.12)*, Kister *(ref.7)* and Bernstein *(ref.1)*.

### 3.2. *Methods which set up new problems as subgoals.*

If direct attack fails, it may be possible to replace the problem by one or more (presumably) less difficult problems, in such a way that solution of the subproblems will lead directly to a solution of the original. In complex problems this may extend to many levels. The Logic Theory machine of Newell and Simon *(refs.13,14)* uses a number of such methods.

### 3.3. *Methods which set up new problems as models.*

Some of the most powerful techniques used by humans replace problems by new ones whose relation to the original is less immediate. These problems are not equivalent, but only heuristically similar to the original, and their solution need not directly lead to solutions of the original. But to the extent that the problems are really related, we may expect that the *methods* which work on the substitutes will be worthwhile trying on the

original. These techniques include the use of models, analogies, and other ways of transforming a problem into one in an area in which we have more experience or more powerful methods.

3.4. *Methods of characterization or description*

The choice of which method to apply to a given problem depends on what kind of problem it is. In order to select appropriate methods, the machine must have facilities for recognizing problems as members of heuristic categories. If the machine is to be able to learn on the basis of past experience, it must have some way of discerning which records are relevant to a given situation, and again the machine must be able to sort things into heuristically useful classes, i.e., to recognize certain "patterns".

3.5. *Administrative methods.*

As the problem-solving process goes on, we accumulate an elaborate structure of related subgoals and analogue problems. New problems are adjoined, and old ones change their status as evidence accumulates. The management of such a structure requires methods which allocate time and effort, deciding where to concentrate attention. It would seem that these methods should include ways of making the following three kinds of estimates:

*(a) Difficulty estimates.* For each subproblem, one has to be able to make some estimate of its difficulty - an estimated relation between effort expended and the chance of getting a solution. Such estimates would be based on past experience (using methods of type 4) as well as on whatever direct simplicity estimates could be discovered.

*(b) Sub-problem utilities.* The second estimates the utility of solving each subproblem with respect to solving the main problem. This will provide a basis for deciding which problem next to attack. This method would use the estimates of type (a) as data as well as data describing the interrelations between the problems. While solutions to some problems may contribute directly to solutions of others, frequently solution of one problem will merely provide "evidence" which has a good chance of being helpful on another. The utility estimate has to weigh the different kinds of evidence and take into account the whole sub-problem structure.

*(c) Methods for selecting methods.* The estimates above help to select the next problem for attack; one must then select what methods to try. This choice may be based on records of past successes and failures, special experiments on models, and perhaps on the basis of "advice" given from the outside. In each case it is clear that the advice obtained

will be most useful if it really does have general applicability, and again the machine must be able to use adequate methods of categorization.

### 3.6. *Methods of self-improvement*

In the most advanced systems we must include in the framework resources which can improve the operation of the system through adjustment of existing methods and addition of new ones. Discussion of these methods is deferred to section 9.2.

## 4. PATTERN RECOGNITION

The problem of sorting events and situations into useful categories arises in so many ways that it is tempting to regard it as the central problem of artificial intelligence. The enormity of the usual underlying search process requires that each trial result be used to remove (on the average) a relatively large class of trial possibilities. Each method will be fruitful only when applied to some particular class of problems, and efficient operation requires that these be recognized. The sorting operation involved may be called "pattern-recognition" or "characterization". Each category can be assigned a conventional or a computed name. It is only through such names that we can hope to introduce "general" or "informal" advice.

In a machine designed to recognize, or learn to recognize, visual objects or the like, the characterization problem is itself the centre of attention. The simplest techniques are those in which objects are "matched" more or less directly against standards or "templates". One usually has to introduce some notion of similarity; this will generally involve (1) an appropriate measure of goodness of fit with the template and (2) searching for such a match over a set of transformations of either the object or the template. This type of recognition seems limited, in practice, to patterns defined by equivalence with respect to modest collections of easily performed transformations - otherwise extensive search and too many templates would be involved.

A more flexible scheme of categorization is that of listing "properties" or "characteristics". A characteristic of an object is the value of some function from the space of objects into some smaller, more convenient, range. (The values might be, e.g., numerical, verbal, or even neural firing patterns. Objects having a common characteristic should tend, of course, to be heuristically related, e.g., through equivalence under transformations frequently encountered in the problem area involved. *(refs. 16,21).*

The ordered set of values assigned to a given object by a chosen sequence of characteristic functions may be called the *character* of that object and may be used as a computed name of the whole set of objects which have that "character", i.e., have that sequence of properties.

Recognition by (conventional) name of an object is done by matching its character against a set of standards which now may be very compact, easily stored, expressions. Here the matching criterion and the templates play a very small part in the process, and, unless conventional names are required, they may not be needed at all.

While property lists are quite adequate for many purposes, they do reflect the use of a number of computations performed independently of one another. This is not the most efficient way to use computations, and it may not be adequate for general pattern recognition. Often, the decision concerning which properties are to be examined should depend on the outcome of previous computations. Consider the case in which we want to recognize the (topological) pattern of those figures which contain a *closed loop*. This particular recognition clearly requires some kind of *recursive* process (since topological connectedness is not a local property). The pattern involved is defined by invariance with respect to a class of transformations much larger and more complex than is involved in those for the patterns, e.g., "square" and "circle".

Again, even if we did have a set of properties which could usefully characterize "single" visual objects, how could we handle scenes containing a number of such objects? Clearly we want to be able to characterize them individually, and also characterize their mutual relationships. To do this in terms of independent properties would clearly be very awkward. But it would not be very difficult to do it in terms of a *program* containing appropriate conditional transfers.

We can expect then that the descriptive expressions, or names of categories, will require more than simple lists, at least in advanced systems. They will have to take the form of complex phrases in expressive languages, e.g., of fragments of program. Only in this way can we expect to recognize and express the nature of those most important patterns whose definition contains a *recursive* ingredient.

In any case, the success of a recognition program will depend largely on the adequacy of the elementary characterization operations. In section 9 we discuss some of the problems in generating and evaluating such operations.

## 5. CHARACTER-METHOD MEMORIES

Each time a subproblem is considered as a candidate for action, it is important to have access to records based on information concerning

similar problems, i.e., problems of the same type (character). These records should be useful in predicting which methods are most likely to succeed, how much effort will probably be involved, and if possible, what kind of new sub-problems are likely to be generated. Such records can be obtained in a number of ways:

5.1. *Learning by experience*

In the very simplest domains the machine's experience may be summarized by processes of the "stimulus-response reinforcement" type. In such a process there is stored, for each *(character, method)* pair, a record which reflects the utility, over the past, of applying that method to problems having that character. After each instance of such an event, the program must evaluate the relative success, and accordingly modify the corresponding utility estimate. (The resulting numbers could be interpreted in various ways, e.g., as giving a priority ordering for the methods, or perhaps determining their probabilities of future application.)

The precise details of the way in which this record is up-dated will constitute a committment, on the part of the programmer, about the machine's basis for *inductive inference.* For they influence the way in which the machine's behaviour depends on the evidence acquired earlier in similar situations.

For that reason we cannot say, once and for all, which "learning" or "reinforcement" operators are the most desirable. Different tasks requires different standards of evidence. These depend on, among other things, (a) the cost of acquiring new evidence, (b) the penalty for being wrong, and (c) the rate at which the environment may be changing - i.e., the way in which old evidence loses value. The choice of the general form of the operators has to be made on such bases. Fortunately, the strongly self-correcting nature of most learning systems makes them insensitive to the finest details of the operators involved.

In particular one has to decide on how much of the history is to be recorded and preserved. One might store for each Character-Method pair a single number representing relative success, or one might store detailed information about what happened in each such instance. Should one allow for a possible retreat to an earlier position? Should one record failure and success information separately? These options always face the programmer. See Gorn *(ref. 25).*

The futility of aspiring to an absolute answer to these heuristic questions can be seen (I think) by examining the evidently equivalent question - "How can we decide when a question has been properly answered?". Clearly this question is itself not well-defined until we have *chosen* (not discovered!) its answer.

In any case simple reinforcement will be nearly useless in those problems, like chess-playing, in which a very large amount of effort is

expended in each trial, e.g., game. One has to obtain much more information than is implied by success or failure in reaching the main goal. This can be done by the independent reinforcement of the mechanisms involved in each of the many subgoals. As Newell *(ref. 12)* notes, we can reinforce for each subgoal achieved, the methods directly responsible, as well as the methods which generated its sub-sub-goals. If such information is to be useful the machine must, of course, be able to evaluate the utility of the subgoals with respect to the main goals.

### 5.2. *Advice from the outside*

One could also introduce information into the character-method tables directly from the outside. Each entry in these tables entails an injunction to the machine of this form: - "In situations of such and such a type, try such and such a method...". Of course, the ease with which this can be done will depend on the extent to which the characters in use represent abstractions familiar to the operator, or to humans in general. McCarthy *(ref. 9)* discusses this general topic.

Devotees of completely self-organizing systems may feel that this is an irrelevant, if not objectionable, technique. But perhaps we ought not be so ambitious as to be able to design a machine which, from the very start, will have great problem-solving ability. We do not yet know enough about heuristics. Perhaps we have to first learn how to design machines with which we can establish effective communication. At this stage we need, at the least, a channel through which we can introduce suggestions which will help the machine evolve.

### 5.3. *Guidance through gradation of problem difficulty*

In evolutionary systems, the ability to learn to solve difficult problems usually depends on the opportunities for exposure to sequences of related problems of graded difficulty, e.g., to a slowly changing environment. We do not expect systems like those of Solomonoff *(ref. 23)*, Newell et. al. *(ref. 14)*, or the like, to efficiently solve problems far in advance of the kind they have already worked. Presentation of a graded series of problems provides the opportunity to work up sets of simple heuristics which may later be combined to form more powerful ones. Complex actions are usually formed of previously available elements; blind search through the space of complex methods would be as hopeless as exhaustive search in any other complex problem domain. We note that the presentation of carefully prepared sequences of problems may be regarded as a form of (indirect) *instruction*. Evolution moves rapidly in the presence of a helpful intelligence. Thus, in teaching arithmetic, one presents children (and Solomonoff's machine) first with addition without carries. When that is

mastered, one can introduce carries; to do this at the start would strain the available resources for method-generation or concept-formation.

In the very near future we shall be working with heuristic programs of such complexity that the designers will be unable to keep in mind a clear picture of their manner of operation. This is already the case in regard to programs resulting from collaboration, and in the case of some existing large switching systems. Even so, if we have available a method or channel through which we can tell how a problem is being attacked, we may still be able to guide the system with helpful suggestions. But without such communication, we may quickly lose the ability to diagnose, correct, or improve the operation of the system.

## 6. PLANNING AHEAD

Before we attempt to solve a problem or sub-problem we want to select promising methods, perhaps arranged in a strategy or plan. People often do this by indirect methods involving simplifications or models of the problem. If the task is to prove a general theorem, we may first try to find a string of methods which will work in some special case, and later extend the proof scheme. In plane geometry we will draw a relatively crude figure and use visual pattern recognition techniques. In a game one might play out continuations confined to a limited part of the board; only if something promising turns up will one examine the rest of the problem in detail.

Heuristic models need not be consistent in order to be useful. A mathematician can often test the plausibility of a high-dimensional geometric assertion through the use of 2- and 3- dimensional figures whose mathematical structure is quite obscure and personal. Much abstract thinking seems to involve the use of such structures, for which some calculations may be especially simple, provided that the operator knows how to steer around the contradictions that may arise. It may be very difficult to find any single model which reflects all the important features of a class of problems. But fortunately this isn't always necessary. For a partial model may serve well on those sub-problems which themselves don't contain all those features. More detailed examples of this will be seen later.

Just what is a model? One cannot give an absolute definition of the relation between an object and a model of it, simply because the adequacy of the model depends on the questions that one is going to ask it. Perhaps one should say that A* is a model of A to the extent that A* and A give similar answers to those questions considered relevant to the current problem (given suitable codings for the questions). This dependency on a third party cannot be avoided by phrases like "preserving the essential

features of the situation"; still, a structure will not be very useful unless it will serve as a source of models for a range of situations.

### 6.1. *"Algebraic prediction models".*

Suppose that the character-method tables have accumulated a body of reliable predictions concerning (1), which methods are likely to be useful in solving problems of a given character and (2) the expected distribution of the resultant subproblems. It may be possible to use this information in planning ahead, even when the entries do not point directly toward a solution method. This is done by finding, in the algebraic structure of the predictions, chains of methods whose expectation of success is reasonably high. No particular formalism is proposed here, but one can imagine that for each character there is a table indexed by (1) the plausible methods and (2) the plausible resulting subproblem characters. The entries contain data concerning (1) the plausibility, and (2) the estimated difficulty of carrying the method through.

In the most favourable cases the predictions will be relatively definite, asserting that it is good to apply method Mi to problems of type Cj and that the resulting situation will have character Ck. This will tend to occur in machines which have really well-matched characters and methods.

To the extent that these predictions contain reliable information about how problems are transformed, it will be practical to look ahead a number of steps. Naturally, the reliability falls off rapidly with the number of steps. The important point is that this unreliability is out-weighed, for a certain prediction span, by the fact that the look-ahead itself is computationally trivial; at each stage it involves only a few table look-ups. The corresponding application of an actual method to a real problem would involve a substantial effort. Thus we search through the algebraic structure looking for predicted solution chains of reasonable reliability, and we choose those for which the estimated total difficulty is not excessive.

If such a chain of both reasonable estimated reliability and difficulty is discovered, it will be worth-while attempting to perform the corresponding methods. Naturally, there is a good chance that the plan will not work out in detail. But the value of the plan does not then evaporate. It still retains heuristic value. The chain of methods may fail in such a way that it can still be repaired locally; one might eventually invoke, on the basis of experience, a measure of relatedness between methods, and so explore the chains neighbouring those of the plan.

### 6.2. *Applicability*

In applying the character-method algebra, one proceeds to look far ahead, without paying attention to small details, and taking a rather

optimistic attitude. Perhaps some such technique is required in any system which can hope to solve complex problems. One has to first lay down a plan or plausible sequence of major steps, and only then try to fill in the details. Otherwise one gets involved in hopelessly large search trees. Whenever possible, we do this by using an analogy or model which is either very simple or very familiar. But if no such structure is available, we can try a formal device such as the above algebraic summary of the available information.

Naturally, if the predictions are unreliable, the model will not be very helpful. But if this is the case then the machine would have been in difficulty already, since either (1) the characteristics in use don't describe the situations in heuristically effective ways (so that methods cannot be efficiently selected), or (2) the machine's learning methods are not adequate to discover the relations between problem types and successful attacks.

The plans or chains or predictions could be interpreted as assertions or generalizations of the form: 'Problems of such and such a type can be solved by such and such a strategy'. We might choose for the main goal of a machine, the discovery of reliable statements of this sort. If this were done in a machine working on mathematical problems, such statements might take on the form of assertions of metatheorems or deduction theorems. Clearly the machine would only be asserting such statements, while not necessarily capable of *proving* them. But if they had a good record of validity, this alone might seem impressive. In mathematics, one frequently gives as much or more credit to the man who first conjectures a theorem as one gives to the man who later finds a proof; the former may be considered the more creative achievement.

## 7. "SEMANTIC" MODELS AND HEURISTIC DIAGRAMS

In order that a model be useful, it is not necessary that it be especially simple; it may even be of value to use a substitute problem which is really more complex than the original, provided that we happen to have more experience and facility in that domain. Consider the behaviour of a student attempting to prove a theorem in plane geometry. He has been given a set of basic propositions and is presumed to be able to apply some (usually unformalized) rules of inference so as to obtain proofs.

Now both the teacher and the student may, and usually do, pretend that they are operating within the formal system. While they make liberal use of "diagrams" they like to believe that the diagrams are "mere heuristic devices" that they could, if necessary, do without. Of course the fact is that most, if not all, geometers do make use of these diagrams, and if paper were not available they would construct them "in their heads".

Most people find that searching within the logic alone seems meaningless - they find that the diagrams are really necessary to give any "meaning" or "interpretation" to the logical expressions. Without such meaning one cannot guide or motivate the search. But with the "meaning" we can use our "geometric intuition"; we can usually tell quite quickly, for example, whether a proposition is true (and thus save a vast amount of work in trying to find proofs for false subgoals). This ability represents, of course, our powerful (informal) resources derived from our spatial experience.

Thus once the mathematical problem is given a geometric interpretation, we can bring to bear heuristic methods acquired in a more familiar domain. A glance at a diagram tells us at once that it is plausible that two lines are parallel - if there is any doubt it is settled by drawing the figure again with some variation of the unconstrained relations. The good geometer may be able to view the figure as a constrained linkage during such a variation, and thus see whether an apparent parallelism is or is not an accident in a particular sketch.

Now such a heuristic diagram certainly represents some kind of (semantic) interpretation of the problem. But is is important to observe that these interpretations need not always be strictly consistent with the logical system as a whole. We may be satisfied with a much more modest kind of "consistency". For our figure needs to supply interpretations only for those expressions in the formal work which are under scrutiny or are likely to arise shortly. If a new expression fails to find an interpretation in the figure, then we draw a new one with, perhaps, a few additional constructed elements. It is far easier to modify figures occasionally than to try to find one that is completely satisfactory from the start.

Thus in working on a problem, our model need not always be fixed from the start. We may modify it whenever an important expression fails to find a meaning. In a geometry theorem-proving program the figure-drawing process could be under the control of an interpretation unit which would make the alterations when necessary. (This is reminiscent of the system proposed by MacKay *(ref. 11)*. A heuristic model may serve well up to the point where trouble is detected, and even then it may turn out that the trouble can be ignored with relative safety. Such is usually the case in practical mathematics; - one seems to stay out of trouble unless one is seeking it, e.g., in connection with the operational calculus or the logical paradoxes.

## 8. A PROPOSED SEMANTIC MACHINE FOR PLANE GEOMETRY

The experimental programming of machines to prove theorems in Euclidean geometry is of particular interest in connection with the

possibility of substantial use of the interpretations provided by diagrams. (The problem is also particularly attractive because of our highly developed understanding of this interpretation, and because the rather "natural" language used by geometers is not particularly difficult to put into a form suitable for computers.) A program of this sort might consist basically of (1) a logic program, perhaps along the lines of Newell et. al. *(ref. 14)*, (2) a figure-construction program, and (3) a collection of methods which interpret logic expressions, find properties of the figures, and apply the results to control the logical exploration. A program that could do all this would be quite complicated, and our plans are not yet complete. Below are discussed a few of the ingredients that promise to be useful.

8.1. *Construction methods*

Much of the complexity, as well as the richness, of geometry is due to the large domains of constructions available at each stage; one rarely finds a proof that can be easily expressed solely in terms of the geometric elements mentioned explicitly in the statement of the theorem. The new elements are supplied, of course, by "construction", and in the formal system, the construction methods may be regarded as additional rules of inference. If we compare this logic with the one treated by Newell, we see that the available search trees are comparatively overwhelming, and clearly we will need powerful heuristics to minimize the consideration of irrelevant constructions. One might do this by screening proposed constructions for "plausibility", just as one might in a chess-playing program. The program would compute some carefully chosen function of the set of simple relationships between elements of the figures (with special weighting on the kinds of relations important in the current subgoals. Such a program might include some visual pattern recognition techniques, also discussed below.

Introspection seems to reveal that one does not consider a great many constructions in the course of finding a proof, and that many of these are quickly rejected. In addition to the above screening technique, the program would be limited to a small number of construction techniques, e.g., those which produce only a small number of new elements or which have simple symmetry properties.

8.2. *Validity tests*.

We have already noted (section 7) one important way of coupling the model to the formal system, in connexion with deciding whether or not a proposition is true. This could be done by more or less direct measurement with little computer effort. A simple procedure has been found which takes the statement of a theorem as input and constructs an appropriate figure in "general position"; this procedure works for the

great majority of theorems in Euclid. It fails on a few, for which an approximation method can be used. The accuracy required is not large, for if two angles or segments differ by very little there would be no great risk in assuming their equality. It is important to observe that an error in the interpretation programs will *not* induce the machine to present false proofs; an error will only tend to mislead the main program and make it more difficult to find a correct proof. It should also be noted that confining the logical search to the study of propositions which have been shown (by the semantical methods of this section) to be true statements does not in itself make trivial the problem of finding proofs for difficult theorems. It is, however a crucial heuristic step in reducing the otherwise hopelessly large search through the expressions of this big logical system.

8.3. *Pattern recognition methods.*

Another use of the figures suggests itself in connexion with the problem of characterizing situations and choosing methods. For once a subgoal has been chosen, a glance at the diagram will often tell the student how to proceed. He recognizes within the diagram certain patterns which suggest the application of particular methods. Thus, if the subgoal is to prove that a certain pair of angles are equal, the student will become alert to certain configurations of parallels and transversals, congruent or similar triangles, or appropriate patterns involving arcs. (He will recognize these configurations long before he can logically establish their character.) It will not be particularly difficult to program these recognitions. The programmer has the choice of representing the diagrams within the machine either in an abstract algebraic form, or as a real punctate visual pattern. The former is, of course, more convenient for formal manipulations, and the latter would be a valuable aid for the programmer for communicating with, and understanding what is happening within, the machine.

8.4. *Characters and methods.*

The author has hand-programmed the machine to the point that it can handle some of the early theorems of Euclid with reasonably efficient search processes. In the early stages it seems sufficient to characterize problems in terms of their goal relations. Thus theorems are classified simply according to whether they assert that two angles are equal, two lines equal, one line twice another, etc. The methods, too, in the early stages can be simply described. Thus to show that two angles are equal, the character-method tables will list, in order of descending priority, methods whose goals are to (1) show that they are in congruent triangles, (2) in similar triangles, (3) are alternate interior angles, etc. (The proper order is not yet known; they must certainly follow the Euclidean order of

generation, and an important part of the program will contain the methods which decide when the assertions of earlier proven theorems should be incorporated into new methods. This machine will probably be supplied, from the start, with deduction metatheorems which can so exploit earlier results. Later, one can try learning programs for each of these things.)

Each method may itself generate a sequence of subgoals. Thus the first method we mentioned was one which tries to show two triangles congruent. Subgoals for this would be to (1) show equality of two corresponding sides and included angle, (2) three corresponding sides, etc. The very first test of this proposed initial priority structure actually obtained a proof unknown to the author and nearby associates.

The first theorem attempted was the proposition: *The base angles of an isoceles triangle are equal.* The character-method tables generated the following steps; In *ABC*, with *AB* equal to *AC*, we are to show that two angles (*B* and *C*) are equal. Goal: Are they in congruent triangles? (What triangles are they in? *ABC* and *ACB*.) Can we show these congruent by "side-angle-side"? Yes. (*BA,A,AC* matches *CA,A,AB*!) Q.E.D. Most students prove this theorem by constructing the median and proving (easily enough) that the two inner triangles are congruent.

While this doesn't really tell us much about the power of the basic schemes used, it does point out an interesting property of such machines. They need not become so confused as do humans in "degenerate" situations; the student feels uneasy about using the same triangle in two ways at the same time. This does not point to any basic difference between people and machines, however; when more powerful heuristics are programmed, we will not be surprised if the machines too become confused in the special cases in which the internal models fail to give unambiguous representations of things.*

*NOTE. Most of the remarks in this section were based on multilithed notes distributed by the author at the Dartmouth Summer 1956 Artificial Intelligence Project which was supported by the Rockefeller Foundation (with the author attending as a member of the M.I.T. Lincoln Laboratory and also of the Harvard University Society of Fellows). Since then work along these lines has been under way on a program for the IBM 704 by H. Gelernter and others in N. Rochester's group at the IBM Research Laboratories. The work of the latter group will presumably be reported elsewhere. It will be some time before the author's program will actually run on a computer. The actual coding is awaiting construction of a suitable programming language and compiler.

## 9. THE GENERATION OF CHARACTERS AND METHODS

An intelligent system should have the ability to learn to solve more and more difficult problems as it gains in experience. Such a system ought to be able to increase its resources when necessary, e.g., through the creation of new and/or better characteristics and methods. How can such resources be invented?

One could legitimately regard this as just another kind of search problem - a method, or a characteristic, is just a fragment of program. If we had a way of deciding when such an object was useful, then we could, in principle, search through the space of programs. As usual, to attempt such a search without having powerful heuristics, would be folly. How can such a search be made reasonable?

### 9.1. *Theories or hierarchies of complexity*

An ancient and still attractive idea is that of trying the simplest methods first, and progressing through successively more complex techniques. It would be easy to make this more precise if there were available an acceptable theory or effective method of enumerating methods (or programs, or machines, or logical expressions) in order of simplicity. A number of such rankings have been proposed; these are too numerous to mention here and none of them may be said to give much practical guidance.

In this connection the results of recursive function theory, in establishing hierarchies of relative solvability classes, are certainly suggestive. The trouble is that all of the computable functions get lumped into one base category. Perhaps these "transfinite" techniques contain ideas which could be used to make finer and more useful distinctions within the basic class of computable functions. Time will tell.

### 9.2. *Invention based on a measure of similarity.*

For want of an absolute ranking of methods, one must, as always, fall back on the basic heuristic principal: Try methods that are similar "in an appropriate sense" to those that have been successful in the past. Each problem domain has, of course, its own structure, and its own relevant kinds of similarity. In each domain it turns out that, *once the available methods are designated by expressions in some chosen language*, then, for better or for worse, particular notions of similarity suggest themselves.

Suppose, for example, that the methods are to be described by programs in the order code of a particular computer. Then we might take two programs to be "similar" to the extent that they have, or share, common instructions. This idea was explored in an interesting program due to R. Friedberg *(ref. 5)*. The method did not seem to work out as a powerful heuristic generator in itself. Presumably, a powerful method generator of

this kind would have to be able to detect those larger fragments of program ("subroutines") which act as more or less separate entities (performing isolable functions). The detection of such entities constitutes in itself a major problem in inductive inference, of course. The author knows of at least one effort under way to work out a program along these lines *(ref. 18)*.

9.3. *Assembly from smaller elements*

It is clear that ordinary elementary machine instructions are too small to serve well as basic terms in languages which can conveniently describe methods. Perhaps such schemes will be much more effective when operating on the expressions of the more powerful programming languages under development. The most effective kinds of schemes are those in which the methods already known to be useful can be "factored" into relatively large common parts (raising again the kind of problem involved in detection of subroutines). Two examples of generators of apparently high heuristic power are described in the works of Selfridge and Dineen *(refs. 4, 15, 16, 17)*, and Solomonoff *(ref. 23)*.

9.4. *The model of Selfridge and Dineen*

The target of this program is to obtain, for the purpose of recognition of visual patterns, a property list description of the type discussed in section 4 above. What is interesting here is the manner in which it is proposed that new characteristics be generated.

The program starts equipped with a basic set of "elementary" transformations $A_1$, each of which is an operation or transformation which operates on one picture to produce another. These elementary operations might include such potentially useful operations as extraction of boundaries, detection of vertices or line segments, intensification of contrast, filling of hollow figures, and the like. The machine may apply to a given figure any sequence of such elementary operations, each working on the figure resulting from the previous application.

Now each such sequence could be regarded as a particular transformation, so that (starting with a finite base) there is available an infinite variety of picture-transforming operations. Each of these can be converted into a "characteristic" or "property" by following it, e.g., with any function from the space of pictures to the integers. The "blobbing" operation, which replaces a picture by the *number* of separate objects in the picture, was chosen for this purpose.

It would be impractical to generate trial sequences either by enumeration or at random. As Selfridge remarks, we want to try new sequences which are like those that have already proven their worth. Determining the value of a characteristic does not pose any particular problem here; one might compute the correlations between the names of pictures and the values of a given characteristic.

The proposed generation scheme is this: the machine will collect data on the transition properties of the sequences of established utility, and build new sequences for trial on the basis of these transition data. The occurrence of a relatively large coefficient would tend to indicate the presence of something like a subroutine. It would be remarkable if such a generator did not perform much more efficiently than one based on random selection. It may be noted that if one of the elementary operations is itself useless or destructive it will automatically be retired. A worthless but benign operation will be retired, but more slowly. Thus one could occasionally add new elementary operations, if desired.

This promising approach to pattern recognition has never been followed through in full detail, to the author's knowledge, The author has been able to explore some of the properties of these sequences, with the help of a much faster computer than was available to Selfridge and Dineen at the time their work was done, and my results bear out their contention that this way of generating new characteristics is very effective. Further interesting work along this line is reported by Kirsch et. al. *(ref. 6)*.

### 9.5. *Solomonoff's proposed Inductive Inference Machine*

Each of the method-creation schemes discussed above may be regarded (although the reader is not compelled to do so) as operating on the expressions or strings of a method-describing language. The probably useful methods are obtained by reshuffling in various ways those terms and other grammatical structures that have already been outstanding. As experience accumulates, it might be desirable to revise the language, e.g., by assigning briefer names to the more useful linguistic objects.

The generating schemes of Solomonoff, working on a different class of problems, are much more difficult to describe in brief, and we must refer to his original paper *(ref. 23)*. The difficulty is twofold; his elementary structures are two-dimensional, and the published description is not complete in all details. A revised version is expected to appear in an early issue of the Journal *Information and Control*. The proposed mechanism is quite complex, and can be summarized only incompletely here. The machine is to be given a sample collection of correctly worked examples of (initially simple) mathematical problems. It is then given some uncompleted problems and required to supply the missing parts. To do this, it must be able to discover the meanings of the symbols involved and learn how to use them. This is a prototype problem of the sort in which the task is to make an acceptable generalization about a body of available empirical evidence.

Now as we know, there is no possibility of finding a method of constructing generalizations which will always remain valid. The machine may be able to learn the meaning of the symbol for addition in the context of

problems which do not require "carries". But this interpretation, or generalization, about what the symbol means will certainly turn out to be incorrect as soon as the machine is given addition problems which do require carries. When this happens, the "empirical utility" of the now incorrect generalization must drop to zero. It has no further value in the role of finding solutions. But it would be disastrous if the machine had then to start again from the beginning. The "patterns" involved in addition with carries are very complex (since the most significant digit of the answer is a function of all the digits of the summands). Fortunately, as Solomonoff observes, even when a generalization loses all predictive value, it may still have precious heuristic content, e.g., for the generation of new hypotheses. The next discovery may be accomplished through a relatively small variation of the otherwise discredited structures. To exploit this value, one must of course have some method of building new structures out of old ones, and the body of the paper under discussion is concerned with proposals in this direction.

In connection with the problem of constructing administrative methods to manage these techniques, it turns out that one must deal with at least two distinct notions of utility: (1) the relatively straightforward notion of an empirical or predictive utility, and, (2) some kind of *a priori* estimate of the utility that a structure will have in its role of generating new structures. The forthcoming version of this proposed program will attempt to formulate in mathematical form some of the properties required of these functions.

### 9.6. *Structural similarity and network machines.*

Most of the systems discussed up to this point may be regarded as dealing with rather clear-cut syntactic processes involving the manipulation of symbolic expressions. But other kinds of adaptive machines have been studied for which there seem to be no such natural linguistic descriptions. These are the "network" machines which are composed of large assemblies of interconnected components, many of which may be operating at one moment. The variations of behaviour, the source of new modes of operations, are obtained from local variations in the properties of and the interconnections between the elements or "cells". In such machines, the cells may be simple and more or less similar to one another, as in the so-called "neural-net" models, or they may be specialized and individually relatively powerful, as in the "pandemonium" schemes of Selfridge (see this volume). In the neural-net models (to which this discussion is confined) the interconnections are often supposed to be organized in a rather loose way on the local level; more precise specifications may be made about how larger sub-assemblies are put together.

The motivation for the study of such machines is at least threefold. First, people feel that the brain must be such a machine, so that eventually we shall be bound to make such studies. Second, we may have here a sort of "natural" source of heuristically useful behavioural variations (useful if we can find ways to reinforce those that we prefer) and the problem is thus of interest from the purely artificial intelligence viewpoint. Finally it is clear that for reasons of speed and economical use of components we shall eventually be forced into the use and study of "parallel" machines.

The variations (and the associated notion of similarity) are derived from local variations in the properties and interconnexions of the individual cells, and it is hoped that these changes on the "microscopic" level will lead to heuristically useful variations on the gross behavioural level. It is well known that this would be a very poor concept to apply to machines like modern computers which are designed so that local variations produce very large behavioural changes. ("Large" has to be interpreted in an appropriate heuristic sense.) We are very nearly forced to the conclusion that this kind of variation can be useful only in *machines whose exact connexions don't much matter from the beginning.* This, I think, is the reason that proposals of this sort have so often been concerned with varieties of "locally random" machines. One should add, of course, that there is general agreement that the genetic control over brain structure is probably not precise enough to permit much more highly organized alternatives. Among the proposals of this sort are, for example, those of Hebb *(ref. 8)*, Farley and Clark *(refs. 2, 3)*, Minsky and Edmonds *(ref. 10)*, Rochester et. el. *(ref. 19)*, and Rosenblatt *(ref. 20 )*.

It has been shown, in all these ways, that it is possible to obtain elementary learning and other kinds of adaptive behaviour (including various reinforcement, association, and servomechanical modes) from these locally random or "self-organizing" nets. We cannot escape the feeling that, if really sophisticated behaviour is desired, there must be some way in which the system can acquire highly developed *hierarchies* of organization; that there must develop some kind of relatively substantial physical representation for higher order concepts. Most of the net theorists who were not satisfied with the simplest forms of learning have tended in this direction, usually by trying to show that fragments of the net will tend to become grouped into functionally unitary entities. This was first made plausible by Hebb, who formulated the first reasonably sophisticated, if rather incomplete, theory; he named these functional groups "cell-assemblies". These cell-assemblies are themselves supposed to represent heuristically useful concepts, e.g., individual characteristics of stimuli, and higher order concept-formation is to occur through the development of useful connexions between them. Although at first the development of satisfactory theories about such structures appears nearly hopeless, it would not be too surprising if, once we have mechanisms for

the formation of simple concepts as physical sub-structures, the remaining heuristic theory would not be very different from the kind concerned with the formal or linguistic models. One might soon need have little concern with the underlying nets. This is one reason why the author feels that, even for those whose central interest is in unravelling the mysteries of the brain, it might be well to devote a major share of the effort, at the present time, to the understanding and development of the kind of heuristic considerations that some of us call "artificial intelligence". There seems to be no real danger, at the moment, that we are going to be overwhelmed with so many plausible theories of how people *might* think, that we would not be able to choose from these a few which might help in the understanding of "natural intelligence"!

## BIBLIOGRAPHY


1. BERNSTEIN, A. Computer vs. Chess-Player. *Sci. Amer.*, 1958, **198**, No. 6.
2. FARLEY, B. G., & CLARK, W. A., Simulation of self-organizing systems by digital computer. *Trans. IRE Prof. Group on Info. Theory Symposium*, Sept. 1954.
3. CLARK, W. A., & FARLEY, B. G., Generalization of pattern recognition in a self-organizing system. *Proc. Western Joint Computer Conference*, IRE, March 1955.
4. DINEEN, G. P. Programming Pattern Recognition. *Proc. Western Joint Computer Conference*, IRE, March 1955.
5. FRIEDBERG, R. M. A learning machine, Part I. *IBM J. Res. Dev.*, 1958, **2**, No. 1.
6. KIRSCH, R. A., RAY, L. C., CAHN, L., & URBAN, G. H., Experiments in processing pictorial information with a digital computer. *Proc. Eastern Joint Computer Conference*, IRE, 1957.
7. KISTER, J., STEIN, P., ULAM, S., WALDEN, W., & WELLS, M. Experiments in chess. *J. Assoc. for Computing Machinery*, 1957, **2**, 4.
8. HEBB, D. O. The organization of behaviour. *Wiley*, (1949).
9. McCARTHY, J. See this volume.
10. MINSKY, M. L. Neural nets and the brain-model problem. *(Avail. University Microfilms, Ann Arbor, Michigan).*
11. MacKAY, D. M. The epistemological problem for automata. In *Automata Studies*, Princeton, 1956.
12. NEWELL, A. The chess machine. *Proc.* 1955 *Western Joint Computer Conference*, IRE, March 1955.
13. NEWELL, A., SHAW, J. C. Programming the logic theory machine. *Proc* 1957 *Western Joint Computer Conference*, IRE, Feb. 1957.

14. NEWELL, A., SHAW, J. C., & SIMON, H. A. Empirical explorations of the logic theory machine. *Proc.* 1957 *Western Joint Computer Conference*, IRE, Feb. 1957.
15. SELFRIDGE, O. G. Pattern recognition and modern computers. *Proc.* 1955 *Western Joint Computer Conference*, IRE, March 1955.
16. SELFRIDGE, O. G. Pattern recognition and learning. In Third London Symposium on Information Theory, *Academic Press*, (1956).
17. SELFRIDGE, O. G., See this volume.
18. SILVER, R. L. Private communication.
19. ROCHESTER, N., DUDA, HAIBT, L., & HOLLAND, J. Tests on a cell assembly theory of the action of the brain, using a large digital computer. *Transactions on Information Theory*, IRE, IT-2, No.3, Sept. 1956.
20. ROSENBLATT, F. *The Perceptron*. Cornell Aero. Lab. Report VG-1196-G-1.
21. PITTS, W., & McCULLOCH, W. S. How we know universals. *Bull. Math. Biophysics*, 1947, 9.
22. SHANNON, C. E. Programming a digital computer for playing chess. *The World of Mathematics*, 4, Simon and Schuster, 1956.
23. SOLOMONOFF, R. J. An inductive inference machine. 1957 *Convention Record*, IRE Part 2, March 1957.
24. TURING, A. M. Can a machine think? The World of Mathematics, 4, *Simon and Schuster* 1956.
25. GORN, S. Paper to appear in *Information & Control*.

# DISCUSSION ON THE PAPER BY DR. M. L. MINSKY

PROF. VAN SOEST: I am very much impressed by so many most interesting papers presented to this symposium, one of which being that of Dr. Minsky. It is of great value for me to come into contact with others working in the same sort of field as we are doing at the Technical University at Delft.

Our aim there is to look for useful and efficient combinations of pure logical machinery with machinery having some degree of intelligence. The pure logic machine, in some cases, can be too complicated, maybe too expensive. The other type, equipped with trial and error and with learning behaviour could be too wasteful in time. It is a search for a reasonable harmony in combining both together. As a very preliminary example indeed, we have built, among others, an apparatus determining the design characteristics of an electromagnetic transformer of which a priori, a few data have been given.

I completely agree with Dr. Minsky on his views on a definition of intelligence and creativity. Doing otherwise, demonaical thoughts of the outer physical world are too easily introduced, confusing the straightforward way in which research in this field seems to be necessary.

A problem related to the topics of Dr. Minsky's paper is of particular interest to me. It is a problem of systematics and taxonomics, say in botany: the identification and recognition of species. Biologists use a complexity of means for this purpose, the most powerful, and the most extensive at the same time, being the use of identification keys, generally of a tree-like structure, as mentioned by Dr. Minsky in para. 3.1. Those keys only can represent in a useful way if the definitions used are sharp, and if the plant material in study is complete enough. If not, the botanist has to take other considerations into account.

An example for this is the use of pictures or figures in addition to the language written or printed keys. They deliver a pattern recognition in a better way than is possible with a rather short text. See Dr. Minsky's paper, para. 3.4.

Very often the identification key is not composed in the most scientific way. Say, for instance, that fruit characteristics are predominant, but that in practice fruits very often are not available; if a first split-up in the tree structure of the identification key depends on it, this will not be very useful at all. So a transformation in key-characteristics,

often to vaguer ones, is necessary; see Dr. Minsky's paper para. 3.3. I could continue to show the resemblance with Dr. Minsky's problem.

There exist three ways to improve, by some mechanisation process, the identifications in question. Firstly; to change the existing key into a, maybe, totally automatic machine that observes colours, patterns and so on. The technology to reach this may be difficult, but it is not impossible. It does not so much belong to the items of this symposium, perhaps. I would go over to the second point, to develop a machine that has the property of pattern recognition in order to compare identification-key results with pictures and other figures. Thirdly, to develop methods with which a machine, exactly in the way as an author-taxonomist has to do, can compose such keys. A priori, the whole knowledge of the species and their variability concerned has to be put at its disposal.

How can all this be done? I am studying very carefully the general philosophy of how the botanist is doing this work himself, and how this analytical process takes place in him. The ideas explained by Dr. Minsky are relevant to all this, and important and stimulating for my study. I wish him all success with his geometrical model.

DR. H. B. BARLOW: I am impressed by the fact that Dr. Minsky does not seem to put forward any ideas as to the general nature of intelligence, or about the end towards which it is normally directed. He is concerned with a particular manifestation of it — that which enables a person to prove simple geometrical thereoms for instance -- and as I understand him he is trying to get this capacity incorporated into a machine. But this is an advanced manifestation of intelligence which is beyond the powers of many human brains. In other words, isn't Dr. Minsky trying to put the thick end of the wedge into his programmes?

People seem agreed that, in their present form, high speed digital computors are fantastically "stupid"; it would therefore seem logical to try inserting the thin end of the wedge first. A task which seems to me to require such elementary "intelligence" would be the processing of data so that they can be displayed or stored more economically. At its simplest this is an easy task, but it links up directly with Ernst Mach (*ref. 1*) and Karl Pearson's (*ref. 2*) contention that "Economy of Thought", or the economic description of observations, is the essential activity of science.

REFERENCES

1. MACH, E. The Analysis of Sensations. Trans. from the fifth German Ed. by S. Waterlow. *Open Court Publishing Co., London & Chicago* (1914).

2. PEARSON, K. The Grammar of Science. *Scott, London* (1892)

The programming of such tasks might throw more light on intelligence than the programming of problem solving routines.

DR. L. C. PAYNE: I should first like to congratulate the author on what I thought was a most stimulating paper. I have about five points to make and five minutes to do it in, and each of them really requires a paper on its own to do it justice. Although there are extremists on both sides I don't think anyone is under any illusions about whether a machine can think in the same sense that a human being can think. We are here this week to discuss the role of data processing and control systems. Nearly all important control systems depend on the flow of formation in closed loop systems - what might be described as data processing circuits - and such loops generally embody the human brain as the processing mechanism of the loop. What we are trying to do is to find how effectively we can replace this by computing techniques. Simple and familiar examples of first attempts in this direction are the thermostat using one bit of information, and analogue circuits with linear feedback. The problem is, in what way, and to what extent, can we complicate the data processing part of the servo-loop so as to achieve more sophisticated forms of control along the lines suggested by Dr. Sutherland; that is doing less than completely emulating a human being, but perhaps doing something faster or for less cost, which is certainly a sufficiently limited first objective.

I believe that starting in this modest way it is realistic to hope that further developments will lead to equipments that simulate thinking more and more. The grounds for this belief will be clearer if I start with a fairly large generalization and show how it can be narrowed and focussed to manageable proportions.

The essence of all scientific activity, what we call the application of scientific method, is talking consistently about what we have observed. This activity distils itself into the laws of science. With these rules - they might be called programmes - the human brain operates. With the rules of, say, Newton's Laws of Motion, a brain can evaluate the motion of a projectile without going through the original labour of Newton. This is analogous to a computer, which once loaded with a set of rules - a programme - can evaluate results without going through the original labour of drawing up the rules - or programme. The human being can therefore be regarded as a digital computer *par excellence;* his inputs are the very diverse sensory inputs, not nearly so simple as punched paper tape and such like, and many orders more subtle. His data processing element - his brain - can draw from the whole "library" of systematized knowledge that we call science. He can draw from Newton's "subroutines", or Darwin's or anybody else's, by which to process the sophisticated input data. Furthermore, he can then output it in a very complex way through elaborate muscular actions, which make punched and magnetic tapes look very primitive indeed.

That is certainly a large generalization, and I do not want to pursue it; but I think it gives hope that we can go a long way in making control loops with sophisticated control elements simulating human control. As A.N. Whitehead said in "Science and the Modern World","it is the large generalization limited by the happy particularity which is the really fruitful conception". Having sketched a rather large generalization I would now like to particularize it, to bring it down to earth and link it with what is being, and could be, achieved.

We may define intelligence, in a way which is perhaps less than adequate to a philosopher but certainly good enough to recognise intelligence, as the ability to discern relationships both analytically and synthetically. The analytical, of course, we associate with the deductive mode of human reasoning, and the synthetic is what we call the inductive mode of human reasoning. The mechanical nature of deductive reasoning has been recognized for a long time; a person trained exclusively in this process, such as many mathematicians, do not make very successful research workers, at least to begin with. The power of deductive logic in tackling new problems is very limited, and indeed mathematics tend to follow in the wake of new developments rather than in the man. As my Cambridge professor aptly put it, "mathematics is no substitute for clear thinking". However, once rules are established, and I have maintained that scientific activity is simply consistent rule making about observed phenomena, deductive logic can elicit many latent ramifications which are by no means immediately obvious. Further, when science has found rules, for example the physics of colour, we do not think it has failed in its object because subjective experiences, such as redness, are no part of the rules. Similarly, I think eventually, rules will be found for human intelligence so that machines built on those lines can simulate intelligence. I suggest that these will be no less complete because they do not treat the subjective experience of intelligence; and that the arguments that a machine can not be intelligent because it can not *feel* or experience intelligence as we feel it are irrelevant. Once such laws are found, doubtless they will become more and more refined as the science of intelligence becomes established. Deductive logic applied to such laws may elicit many ramifications which engineered into programmes will doubtless give the appearance of subtlety to many machine decisions.

However, one feels much less optimistic that a machine will simulate the inductive mode of reasoning, especially of the kind that we call intuitive insight. This constitutes the essence of originality and creativeness which we tend, perhaps rightly, to regard as more essentially characteristic of human intelligence, than the mechanical-like processes of deduction. Inductive inference is controversial, and some people would contend that there are no systems for induction. However there is one

system which Sir Ronald Fisher calls the fiducial argument (*ref. 1*). It only applies to a restricted class phenomena but it is a mode of induction which is logically sound and watertight. It could therefore be programmed into a machine so that one class of inductive inferences could be made by machine. This is a start. I think any intensification of the study of induction which could result in further systemization would greatly extend the ability of computers to emulate inductive mode of human reasoning. There is one thing more I would like to say on this subject of induction, or the synthetic aspect of human reasoning, that is that a good part of research, of solving problems, is formulating them in such a way that a solution is possible. This should make us very sceptical about a machine solving problems in the more general sense.

The history of science, for example the hereditary process, is punctuated with wrong questions, metaphysical propositions which only yielded ground when a scientific approach was adopted. I think a lot of obscure metaphysical thinking surrounds discussions of intelligence today. With a truly scientific approach, a single new concept can clarify the air considerably, making further progress possible, and other questions meaningful. I think it is possible to advance in a similar way with human intelligence and that with the rules or laws established, that is, intelligent behaviour logically systematized, we can look forward to engineering these rules and having machines acting as intelligently as we understand the mechanism of intelligence at any given epoch.

DR. M. L. MINSKY (in reply):

1. (to Prof. van Soest) I wish to thank Prof. van Soest for his kind remarks. The problem of taxonomic retrieval is a challenging one and mechanization of the composition of keys would require a powerful combination of resources. Prof. van Soest's second proposal -- to mechanize the comparison between a real visual object and a set of diagrammatic representations or pictures --is an interesting one and I wonder how this task would compare in complexity with the more usual problem of characterizing objects in terms of symbolic descriptions or names.

2. (to Dr. Barlow) I am surprised that Dr. Barlow feels that it is somehow wrong to attack the more impressive manifestations of intelligence. The apparent "stupidity" of digital computers is, of course, only a reflection of the things that their programmers make them do (or know how to program). Accordingly, most computers spend most of their time either on conventional

REFERENCE

1. FISHER, R. A. Statistical Methods and Scientific Inference. *Oliver & Boyd, London,* (1956).

numerical calculations or indeed on the processing of data so that it can be stored or displayed more efficiently. To be sure, an intelligent system will need a powerful data retrieval facility, *if* it has to work in an area whose problems require large amounts of storage. But a retrieval system alone, i.e., one not concerned with *the meanings* of the data, will not be intelligent: one cannot forever avoid the intellectual structures involved in the strategies and techniques of problem-solving.

I think there is some misunderstanding about the reasons for the choice of problem domains in the earlier "artificial intelligence" programs. It is true that these have tended to be concerned with problems of formal or mathematical character. But this was not done as a deliberate attempt to simulate the most "advanced manifestations" of intelligence. On the contrary, such problems have been chosen for their relative clarity and simplicity! And they bring out the problems of strategy with a minimum of concern for ordinary data-processing and retrieval routines. Indeed, we have found it more difficult to deal with the "ordinary" manifestations of intelligence that the "extraordinary" ones of games and mathematical problems. In the work on the "Advice-Taker", Dr. McCarthy and I hope to make some attempts on this retrieval problem that Dr. Barlow has raised. The system will have to file advice away in a manner which reflects what is known about its relevancies.

3. (to Dr. Payne). I feel that Dr. Payne is over-pessimistic in two respects. The first matter is connected with the desire for a logically sound, watertight, basis for inductive inference. Now everyone should agree that we cannot expect to find an absolute basis for inductive inference. (See my paper, 5.1). For the appropriate inductive inference basis depends on the universe in which we are embedded, and this in turn can be hypothesized only through an already assumed basis of inference. Clearly *for either men or machines* the basis of induction must be empirical and heuristic; suitable for some universes and not others; and only partially suitable for most (including, presumably the one in which the machine will actually find itself). Now I think that everyone will also agree that the problem of mechanization of inductive inference is indeed basic to the artificial intelligence problem. If we are realistic about this we see that we need not and must not concern ourselves with the question of absolutes but can attend instead to the practical problems of 1) formulating heuristically useful (pragmatic) bases and 2) effectively mechanizing them. The same is true of the problem of pattern-recognition or categorization.

I disagree with Dr. Payne in a second matter (we have no real quarrel on the first) concerning our guesses about how to proceed and what we can expect to achieve. It may seem like common-sense to say that scientific

procedure requires that each phenomenon be well-understood before more complex systems can be effectively studied. But one must be cautious about such assertions when one is in an essentially *engineering* domain. Obviously Mechanical Engineering goes far beyond what can be inferred on the basis of present-day *chemical* theories of structural materials. And the latter, as we all know, goes far beyond what can be inferred from the state of atomic physics, etc. Similarly, I believe, the overall flow diagrams or "principles" of some complex intellectual processes can profitably be studied without full attention directed at the fine structure of, e.g., some of the "learning" sub-processes. For one thing the fine details of the sub-processes may not have very much influence on the overall performance of the system. For another, we will be in a position to evaluate the significance of various proposed sub-processes only when they are embedded in larger systems involved in really respectable performances.

We have to be careful not to attack problems that are really too simple. Otherwise our models will perhaps not tell us anything new or, more serious, not tell us whether the methods that we have tried are really extendable to more challenging problems. This extensional inadequacy is a property of most present-day psychological models of elementary learning and association. They can be as elaborate and mathematical as you like, but they simply don't suggest what to do when, e.g., the rewards are contingent in an interesting way on what the creature might do. This is because they summarize, in a "scientific" way the data obtained when experimental animals are challenged with perfectly trivial problems. As logicians, perhaps, we ought to "walk before we can run" but, as engineers, we must run just as fast as we possibly can: Science will stop if we wait to allow the "foundations" catch up with the pragmatic theories!

Now what I am maintaining is that Dr. Payne is over-pessimistic in his statement that "we can look forward to...having machines acting as intelligently as we understand the mechanism of intelligence at any given epoch".

Let me explain my position with a few remarks on the question of *systems design* and the problem of how one is to understand how very complex processes work. These remarks tie in very closely with those of my colleagues John McCarthy and Oliver Selfridge. (See Session 1, paper 3 and Session 3, paper 6). If you will examine the systems proposed in these papers you will see that underlying each of them is a deliberate attempt to show how one might be able to create large systems *without actually understanding them* in full "chemical" or "atomic" detail. (As a visitor I must bow to your great humorist who has so relentlessly shown us how to do so many things "without actually" doing anything much at all. In America there is rising, among those connected with computers, a science called "Research Simulation" which seems distinctly related to this.)

The idea is that we hope to be able to build up some very complex structures with these systems by use of empirical, heuristic, evolutionary

techniques. These techniques depend critically on being able to observe the system in operation and *make small changes in the desired direction.* Both the *Pandemonium* and *Advice-Taker* systems are expected to be capable of tolerating such interference: in the first by alteration of the "sub-demons" and in the second by addition of general advice to handle new classes of situations. The hope is that in the usual case this interference can be tolerated without extensive revision of the system. We are prepared to find that as these modifications accumulate we will lose our detailed understanding of all but the more general principles of the system's operation. But we should learn, so to speak, the art of steering such evolutionary processes into levels of performance well beyond what we could reasonably hope to obtain through any kind of analytic solutions of systems equations (if any exist).

CHAIRMAN: I wonder whether, in your last few minutes you could say a little bit more about the word 'heuristics', as that is something which is not quite clear. It appears that in different papers the word is used in rather different senses. We would be very grateful if you would say a little about its use.

DR. M. L. MINSKY: By "heuristic" we mean to refer to things related to problem solving. In particular we tend to use the term in describing rules or principles which have not been shown to be universally correct but which often seem to be of help, even if they may also often fail. The term "heuristic program" is thus used in reference to the distinction between programs which are guaranteed to work (and are called "algorithms") and programs which are associated with what the programmer feels are good reasons to expect some success. The term "heuristic" has come into use as a noun as well. We ask someone "what heuristics does your program use?" and he answers with a listing of particular tricks, short-cuts, inductive bases, and the like, regarding each heuristic feature as an individual "heuristic".

(The term has been much used by Polya, whose pioneering work on problem-solving and induction is, it seems, being widely neglected at the present time. *(ref. 1)* The reason for this neglect is, I think, the Polya's writing is actually so rich in unsystematic heuristic suggestions that one's own attempt at systematic development is overwhelmed. As the field becomes more organized we will be more able to exploit this great quarry of insight.)

REFERENCE

1. See the bibliography in POLYA, G., Patterns of Plausible Inference, *Princeton,* (1954).