XVI. ARTIFICIAL INTELLIGENCE*

Prof. J. McCarthy	P. W. Abrahams	D. C. Luckham
Prof. M. L. Minsky	D. G. Bobrow	D. M. R. Park
Prof. C. E. Shannon	R. K. Brayton	S. R. Russell
Dr. Phyllis Fox	D. J. Edwards	J. R. Slagle
	L. Hodes	

RESEARCH OBJECTIVES

The purpose of our work is to investigate ways of making machines solve problems that are usually considered to require intelligence. Our procedure is to attack the problems by programming a computer to deal directly with the necessary abstractions, rather than by simulating hypothetical physiological structures. When a method for solving a problem is not known, searches over spaces of potential solutions of the problem, or of parts of the problem, are necessary. The space of potential solutions of interesting problems is ordinarily so enormous that it is necessary to devise heuristic methods (1) to replace the searching of this space by a hierarchy of searches over simpler spaces. The major difficulty, at present, is the excessive length of time required for building machinery or even for writing programs to test heuristic procedures. For this reason, the major part of our effort is going into the development of ways of communicating with a computer more effectively than we can now communicate. This work has two aspects: development of a system for instructing the computer in declarative, as well as imperative, sentences, called the advice taker (2), and development of a programming language called LISP (3) for manipulating symbolic expressions that will be used for programming the advice-taker system and will also be of more general use.

J. McCarthy, M. L. Minsky

References

1. M. L. Minsky, Some methods of artificial intelligence and heuristic programming, Proc. Symposium on the Mechanisation of Thought Processes, National Physical Laboratory, Teddington, England, Nov. 24-27, 1958 (H. M. Stationery Off., London, 1959).

2. J. McCarthy, Programs with common sense, Proc. Symposium on the Mechanisation of Thought Processes, National Physical Laboratory, Teddington, England, Nov. 24-27, 1958 (H. M. Stationery Off., London, 1959).

3. J. McCarthy, Recursive functions of symbolic expressions, Quarterly Progress Report No. 53, Research Laboratory of Electronics, M. I. T., April 15, 1959, pp. 124-152, and Communs. ACM 3, No. 4 (April 1960).

A. INTEGRATION

J. R. Slagle's integration in LISP has been completed. This computes indefinite integrals symbolically in the manner of a freshman calculus student. The program has integrated approximately 70 forms, including 38 forms taken from MIT freshman calculus final examinations. Two examples of integrated forms are:

$$\int \frac{xe^x}{(1+x)^2}\,dx \quad \text{and} \quad \int \frac{\sec^2 t}{\sec^2 t - 3\tan t + 1}\,dt$$

*This work is supported in part by the Computation Center, M. I. T.

The solutions of each of these examples took approximately 13 minutes on the IBM 7090 computer. The central problem dealt with in this program is the same as that faced by the freshmen – namely, the choice of which standard transformations of calculus are to be applied to the integral that is being evaluated. The result of such a transformation may be either a solution of the problem, a substitution of another integral to be evaluated, or a split into two or more new integration problems. After a tree of subproblems has been accumulated, the program has to decide which subproblem in this tree to work on. For the choice of method, a variety of pattern recognition heuristics is used: estimates of difficulty and estimates of the progress made thus far on the problem are used to decide which branch to take. The program will be described in a forthcoming publication.

This program as written in the LISP system taxed the 32, 768 word memories of the IBM 709 and IBM 7090 computers to the utmost. A good part of the last year's effort was devoted simply to squeezing the program into the available memory. Even so, lack of space prevented use of the LISP compiler, so that the much slower LISP interpreter had to be used. Furthermore, a number of heuristic devices that might have sped up the operation of the program could not be included because of lack of space.

The conclusion can be drawn that heuristic programming for projects more complex than formal integration will require substantially larger magnetic core memories if the research on heuristic methods is not to be diluted excessively by coding difficulties.

J. R. Slagle

B. LISP

The LISP Programming Language has been extended by the addition of facilities for the manipulation of integer-indexed arrays, by a new READ program that accepts a more flexible input language, and by various improvements in the LISP compiler. The LISP Programmers Manual is now available (1).

Work is being started on the LISP 2 System, which will be completely recoded in the Linking Loader System for easier maintenance. The main advance of LISP 2 over the present LISP will be the incorporation of facilities that permit the programmer to define new kinds of quantity and the operations on them in terms of the basic quantities or in terms of quantities previously defined by him. LISP 2 will include facilities for both numerical and symbolic computation, including all the facilities of ALGOL. No completion date for LISP 2 has been set.

J. McCarthy

References

1. The LISP Programmers Manual is available at the Technology Book Store, 40 Massachusetts Ave., Cambridge 39, Mass.

C. CHESS

The Chess Project, initially undertaken by J. McCarthy, was turned over to a "sophomore syndicate" in the fall of 1959. The project for that year was a computer program for chess problems in which a checkmate in three moves was demanded. Previous work had given us the mechanism for making and retrieving legal moves. We needed a scheme for choosing good moves because an exhaustive method would be too time-consuming and not particularly subtle.

We focussed our attention on the critical area – the square occupied by the enemy king and the eight squares immediately surrounding it. It is apparent that a checkmate is possible if and only if the king is attacked and none of the surrounding eight squares are accessible to him.

Research into 150 three-move checkmates taken from actual games showed that in all but six of them, the winning first move was a check. Thus we decided to weight checks heavily. Further statistical study showed that critical area conditions should be examined and weighted in the following order:

(i) checks;

(ii) attacks – moves that further limit the enemy king's mobility;

(iii) mask attacks – moves that would attack, except for the presence of one interposing friendly piece;

(iv) elimination of enemy pieces defending the critical area.

Since it is possible to examine all legal moves for checks rapidly, (i) is treated as a special case and investigated first. We then assign values to all other moves, using the criterion stated above with appropriate weighting factors, and choose the best move as our opening move.

If the opponent has a large number of legal replies (e. g., our first move is not a check), we order his moves by the reverse order of the criterion and enter the best move as his reply. This procedure is then repeated for our second move and his second reply; for our last move, we merely examine all checks.

The three-move checkmate program was operable in May 1960. On the average it takes approximately 1.1 minutes of computer time to solve a problem. The longest running time for a problem with a solution was 3.6 minutes. This program was coded for the IBM 704 computer in the Computation Center, M. I. T., and was written mainly in the Fortran source language, with short sections coded in the SAP language.

In the fall of 1960, we began work on the general chess machine. Our first task was research in previous efforts in the field (1, 2). A description of the programs of Shannon, Turing, the Los Alamos Group, Bernstein at IBM, and Newell, Shaw, and Simon was examined (3). We felt that these programs seemed to have one disadvantage in common – aimlessness. This, we felt, could best be overcome by dominating our choice of moves with the broad goal that a chess player calls "a strategy." Much of the previous work has proved to be invaluable and will be incorporated into our system. Because the earlier work did not consider the precise formulation of strategy, we have had to perform our own basic research by investigating how a human being forms and employs strategy.

Our proposed system for the machine playing of chess is the following. Assuming that the opponent has made a move, we combine the effects of this move with our previous strategy and evaluate the new board position with respect to material, mobility, and time to form a new strategy. This strategy determines the relative importance of each of a set of move generators. Each generator is capable of proposing a specific type of move. The generators are then activated in order of importance, and the proposed move is made on a pseudo board. The opponent's replies are then generated by the same method. Continuing this process, we construct a tree of possible sequences of moves. Branches are terminated when a stable position is reached, and a numerical value is then assigned to the position. When all branches are terminated, we evaluate the tree according to the minimax process (4) and assign values to each of our first moves. The move of highest value is then chosen.

For various reasons, not all of which are discussed here, we feel that proper implementation of our general scheme with more efficient control and information flows will result in a machine capable of playing masterly chess. In our opinion, we would be incorporating all of the classical components of a chess game, with the possible exception of psychological influences.

A. Kotok, M. A. Lieberman, C. W. Niessen, B. F. Wells III
Members of the Class of 1962, M. I. T.

References

1. A. Bernstein, et al., A chess playing program for the IBM 704, Proc. Western Joint Computer Conference, May 1958 (AIEE, New York, 1958), pp. 157-159.

2. C. E. Shannon, Programming a computer for playing chess, Phil. Mag. 41, pp. 256-275 (1950).

3. A. Newell, J. C. Shaw, and H. A. Simon, Chess playing and the problem of complexity, IBM J. Research Develop. 4, 320-335 (1958).

4. Ibid, p. 322.

D. PATTERN RECOGNITION WITH LISP

An investigation into mechanical description and recognition of line drawings with LISP has been undertaken. We assume that a drawing is given as a picture on a 100×100 square array. The first part of our process converts the picture into a description so that the second part can work with descriptions.

For the purposes of this report, we ignore the methods of that part which converts line patterns to descriptions. We need say only that the vertices are found and some characteristics of the line segments between vertices are computed. The characteristics chosen were: beginning angle, end angle, length, segment curvature (beginning angle minus end angle), and total curvature (sum of the absolute values of the segment curvatures of portions of the line between inflection points).

The description of a connected-line drawing takes the form:

(1, ((END VERTEX OF FIRST LINE STARTING AT VERTEX 1, BEGINNING ANGLE OF THIS LINE, END ANGLE, LENGTH, SEGMENT CURVATURE, TOTAL CURVATURE), (END VERTEX OF SECOND LINE STARTING AT VERTEX 1, BEGINNING ANGLE, . . .), (. . .), . . .), 2, ((END VERTEX OF FIRST LINE STARTING AT VERTEX 2, BEGINNING ANGLE, . . .), (. . .), . . .), 3, ((. . .), . . .), 4, ((. . .), . . .), . . .)

In this description, line segments are represented twice: once as going from vertex i to vertex j, and again as going from j to i. There is also a list of vertex coordinates, ((1, X1, Y1), (2, X2, Y2), . . .).

The rest of this report illustrates the use of LISP for processing descriptions of line drawings.

Format creates functions for the various parts of a line description:

```
format [LINE; (ENDVERTEX, BEGANGLE, ENDANGLE, LENGTH, CURVE,
TCURVE); (ENDVERTEX, BEGANGLE, ENDANGLE, LENGTH, CURVE, TCURVE)]
```

"Sumlines" is used to join two lines while "aplines" joins a list of lines:

```
sumlines = λ[[a;b][null[b] → a; T → line[endvertex[b]; begangle[a]; endangle[b]; sum
[length[a]; length[b]]; sum[curve[a]; curve[b]]; sum[tcurve[a]; tcurve[b]]]]]]
aplines = λ[[a]; sumlines[car[a]; aplines[cdr[a]]]]
```

An arbitrary test for straightness of a segment was introduced:

```
straight = λ[[a]; not[or[greater[curve[a]; 20°]; greater[tcurve[a]; 40°]]]]
```

Slightly more complex LISP expressions are used to get pairs of lines which continue through vertices, and another sequence of expressions finds triplets of vertices which form triangles. Further work, which includes a program for finding triangles in overlapping figures, is described in Memorandum 18 of the Artificial Intelligence Group, Research Laboratory of Electronics, M. I. T.

L. Hodes

E. DIAGRAMMING SENTENCES

We are writing a program in LISP that will enable a computer to diagram sentences; that is, it will enable us to put all sentences into one general form: subject, verb, direct object, and indirect object. Comparison of diagrammed questions with similarly diagrammed sentences of text will then enable a computer to answer these questions. Thus far, we have written functions to find the complete subject, verb, direct object, and indirect object of a simple sentence; that is, the subject, etc., complete with modifiers, are placed in a predetermined order. The step for diagramming compound sentences is nearly complete. We then plan to consider complex sentences, and finally, the question-answering routine.

The work is related to previously reported work (1).

S. Dunten, J. Martinson
Members of the Class of 1963, M. I. T.

References

1. A. V. Phillips, A Question-Answering Routine, M. S. Thesis, Department of Mathematics, M. I. T., 1960; and J. McCarthy, Memorandum 14, Artificial Intelligence Group, Research Laboratory of Electronics, M. I. T., Feb. 1960.

F. ADVICE TAKER

A major problem in heuristic programming is that of managing situations in which two or more goals are to be achieved at the same time. It is often a simple matter to discover strategies or plans for the goals which will be successful separately but which are incompatible.

When this happens, it is necessary to modify one or more of the separate plans, or to adjust the manner of their execution, or both. A number of heuristic problems for plan modification and adjustment are being studied in a version of the Advice Taker, whose problem domain is the writing of programs for the manipulation of algebraic quantities in a model digital computer.

M. Minsky

G. REVIEW PAPER AND BIBLIOGRAPHY

An extensive discussion of the Artificial Intelligence problem will appear in the Proceedings of the IRE, January 1961, entitled "Steps Toward Artificial Intelligence." This paper is, in part, a review of work on heuristic programming and artificial intelligence and, in part, a summary of the author's work during several years. A subject index bibliography of this field, with some 600 entries and 100 descriptive categories, will appear in the IRE Transactions on Human Factors in Electronics, Vol. 2, No. 1, 1961.

M. Minsky

H. AUTOMATA AND RECURSIVE FUNCTION THEORY

The "problem of Tag," proposed by Post (1) has been laid to rest by a proof that shows that this problem is recursively unsolvable and, in fact, that the "Tag" systems are equivalent, in a rather complicated sense, to general Post canonical systems or Universal Turing machines. The proof will appear in a paper in the Annals of Mathematics. As a consequence, we have constructed a Universal Turing machine that has only six symbols and seven states; this state-symbol product of 42 is now the smallest known product of this kind.

M. Minsky

References

1. E. L. Post, Formal reductions of the general combinatorial decision problem, Am. J. Math. 65, 197-215(1943).