XV. ARTIFICIAL INTELLIGENCE[*]

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

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. This work has two aspects: a system for instructing the computer in declarative, as well as imperative, sentences, called the advice taker (2), and a programming language (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, a paper presented at the Symposium on the Mechanization of Thought Processes, National Physical Laboratory, Teddington, England, Nov. 24-27, 1958.

2. J. McCarthy, Programs with common sense, a paper presented at the Symposium on the Mechanization of Thought Processes, National Physical Laboratory, Teddington, England, Nov. 24-27, 1958.

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.

A. THE LISP PROGRAMMING SYSTEM

The LISP programming system is being improved in a number of ways.

(a) Brayton and Park are finishing the LISP compiler and are attaching it to the system. Already, a compiled program has completed in 0.6 minute a calculation that was not completed in 30 minutes in the interpretive mode.

[*]This work is supported jointly by Research Laboratory of Electronics and the Computation Center, M.I.T.

(b) Luckham and Edwards are finishing the necessary programs so that LISP programs can be operated from the flexo-writer, and share time with the M.I.T. operator program. Experiments have indicated that progress in checking out programs will be much faster, once we can work on-line.

(c) Fox is writing a programmer's manual for the LISP system.

(d) Experience with the present LISP system on a number of small problems has shown the importance of being able to compute with other quantities than symbolic expressions, and a revised version of LISP is being considered which will allow computation of recursive functions of a wide variety of kinds of information.

B. THE ADVICE TAKER

Minsky and Russell have begun programming a version of the Advice Taker, starting with routines for achieving goals that may involve establishing structures of subgoals.

C. THEORY OF COMPUTATION

The recursive function formalism as developed for LISP, described in Quarterly Progress Report No. 53, pages 124-151, has led to considerable simplification and clarification in the descriptions of algorithms. It now seems feasible to develop a logical system in which proofs of the properties of computations can be put into a machine-checkable form with little more work than is involved in an informal argument, and indeed, sometimes, with less work. We hope this will eventually make it possible to eliminate debugging, that is, the trial of programs on repeated test cases, by making it possible to prove that a program has desired properties and checking the proof on a machine.

The insight into the nature of computational processes that is being gained will help with the artificial intelligence problem because in present attempts to make machines improve their own programs most of the effort is spent fighting the language in which the programs are described.

Some of the results of this theory are the following:

(a) The conditional expression and recursive function formalism described in Quarterly Progress Report No. 53 allows us to define the class $C\{\mathscr{F}\}$ of functions <u>computable in terms of</u> the functions of a base class $\mathscr{F}$. Adding quantifiers and a description operator allows the definition of $A\{\mathscr{F}\}$, the class of <u>functions arithmetic over</u> $\mathscr{F}$.

(b) A formal method of proof called "recursion induction" makes proofs of the simpler properties of recursive functions entirely mechanical. For example, consider the recursive function

$$f(n, m, p) = (m=n \rightarrow p, T \rightarrow f(n, m+1, (m+1)p))$$

It is desired to prove that $f(n, m, p) = \frac{n!}{m!}p$. The method of recursion induction allows us to assume the result for the occurrences of the function on the right. Thus we have

$$\begin{aligned} f(n, m, p) &= \left(m=n \rightarrow \frac{n!}{m!}p, T \rightarrow \frac{n!}{(m+1)!}(m+1)p\right) \\ &= \left(m=n \rightarrow \frac{n!}{m!}p, T \rightarrow \frac{n!}{m!}p\right) \\ &= \frac{n!}{m!}p \end{aligned}$$

which allows us to conclude that the formula holds for all n, m, p for which the computation terminates. A general form of recursion induction has been proved to be valid.

(c) A formalism for the recursive definition of sets has been developed. Consider a set S defined recursively by

$$S = A \cup (S \times S)$$

where $\cup$ stands for union, and x for Cartesian product. If A is the set of atomic symbols, S is the set of S-expressions. As another example, the set I defined by

$$I = \{o\} \cup (\{o\} \times I)$$

where $\{o\}$ is the set whose sole member is o, may be regarded as the set of non-negative integers, and the operations on integers are definable in terms of the canonical projection and injection mappings associated with the definition.

(d) Considerable progress has been made in devising a formalism whereby proofs that are checkable by computer can be easily written. When proofs are written to be read by a person the style chosen is a compromise between what is easy for the writer to write and for the reader to follow. If proofs are to be checked by machine, we can make things easier for the writer by taking advantage of the machine's ability to verify the assertion that a given calculation has a given result, regardless of how complicated the calculation is. We accomplish this by formalizing metasyntax so that a procedure for generating a proof is acceptable as a metaproof. Whatever the base system, the metasystem contains recursive functions of symbolic expressions so that any proof-generation procedure is expressible.

(e) Park has written LISP programs to simplify conditional expressions, and Abrahams is programming part of the metamathematical procedure.

(f) A full report of the work on the theory of computation is being prepared.

D. CHESS

The chess program has been taken over by a syndicate of sophomores.

E. INTEGRATION

Slagle's integration program has solved some simple problems, and it is being extended.

J. McCarthy