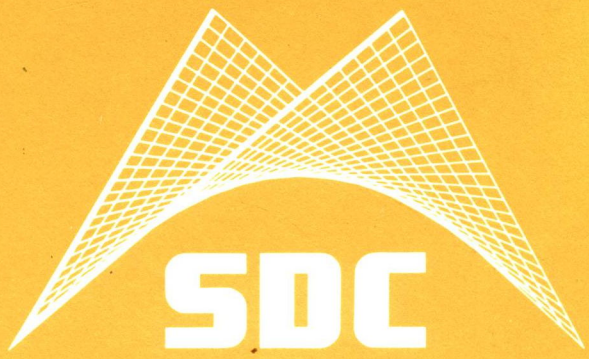


SP-1858/000/01

Three Branches of Artificial Intelligence Research

Aiko Hormann

12 November 1964

SP-1858/000/01

SP *a professional paper*

Three Branches of Artificial Intelligence Research

by

Aiko Hormann

12 November 1964

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

(page 2 blank)

ABSTRACT

This paper outlines three current approaches to the problems of artificial intelligence research--simulation of human behavior in "human-like" ways; achievement of efficient problem-solving behavior using techniques and procedures not necessarily like those used by humans; and improvement in the relationship between man and machine by an alteration in the division of labor between the two and by advances in man/machine communication. Examples are given of artificial-intelligence systems embodying the underlying principles of each branch. Some of the problems that remain to be solved in artificial intelligence research are specified.

THREE BRANCHES OF ARTIFICIAL INTELLIGENCE RESEARCH*

Aiko Hormann
System Development Corporation
Santa Monica, California

Have you ever tried to explain or describe to someone (or even to yourself) the process by which you solved a complex problem? You might have been able to make some global observations, noting one or more insights which were the crucial points, but you were probably not able to explain why you chose this method or that particular path or exploration from among the many other alternatives available.

People constantly "solve" problems that are insoluble by present-day computational standards--insoluble because of time limitations, insufficient data, or overwhelming combinatorial complexity; such occurrences are commonplace in everyday life. Human decision-making is characterized by imprecise terms such as "experience," "intuition," and "good judgment."

The desire to understand what happens and how it happens when we think and behave intelligently is the primary motivating force behind many research activities variously referred to as artificial intelligence, heuristic programming, or modeling of human behavior. But out of this basic motivation have sprung separate branches arising from the different attitudes and beliefs held by researchers, and from differences in what they wish to accomplish.

Some researchers are trying to simulate various aspects of human behavior, hoping that this may lead to better understanding. At this point, it is appropriate to point out the three important reasons this group gives for simulating human behavior on a computer. First, the computer program is a model that represents the researcher's hypotheses about the information

*A talk presented before the IEEE Professional Technical Group on Electronic Computers, Orange County Chapter, Anaheim, California, November 5, 1964.

processes underlying the behavior. Secondly, the information-processing approach forces them to specify their psychological assumptions explicitly and completely and in a way quite different from theories framed in words or in abstract mathematical expressions. Thirdly, the program allows them to generate behavioral consequences from a computer, and thus to study the strict implications of their theories in a way not possible with direct experimentation. The fact that one system of computer programs can serve all three purposes simultaneously is certainly a very strong point in favor of simulation by computer programs.

Another group of researchers wants to find new and better uses of computers by expanding the intellectual capacity of machines, using whatever means are applicable. This second group would say, "Since a person does not have a very good chance of simulating behavior which he does not really understand, we should ask what kinds of programs are needed in order for the machine to manifest the kinds of behavior we want. We might adopt some human-like processes and techniques where they are applicable or desirable, but we will not be restricted to using human-like features alone; we will use whatever techniques or methods we think would be useful and efficient." In short, this group is more interested in producing efficient problem-solving behavior, whereas the first group would deliberately introduce some groping, error-making, and irrational behavior, as well as some clever and efficient behavior.

There is still a third group whose interest is in man-machine partnership. They believe that research in artificial intelligence can lead to better uses of machines by men than are being made today. Conventional ways of using machines seem to accept the so-called "natural division" of intellectual labor between the human and machine; most routine work is done by the machine and all generalization and higher-level thinking by the man. The third group's attitude might be characterized by the belief that, in the context of artificial intelligence, this dividing line must be shifted so that the man can demand more cooperation and adaptive participation from machines in attacking complex and difficult problems. This group has an attitude about utilization of nonhuman features similar to that of the second group, except that they make stronger emphasis on close interactions between man and machine.

These three groups are not clearly separated disciplines, as I may have made them appear to be. Some researchers have several projects covering all three types under their supervision or within their advisory control. As for myself, I can put myself in the second group, but I admit that my learning system--called Gaku--embodies many features that have been inspired or suggested by some of the findings reported by the psychologists and researchers in the first group. Furthermore, I am getting more and more interested in the approach of the third group, because in man-machine partnership, the machine should take advantage of man's special capabilities for its own "intellectual growth," in the same way that the man exploits the machine. The machine can be made to interact with the human so that it can receive suggestions in a particular problem-solving context or can learn to generalize by imitation

after receiving demonstrations of generalization processes in a particular situation. This may be analogous to a classroom situation, in which students follow, step-by-step, the reenactment of the discovery of general principles or generalization processes.

To begin this survey of heuristic programming, let me go back to my first group--the simulators of human behavior. A computer program called the Logic Theorist (developed in 1956 by Newell, Shaw, and Simon on the RAND JOHNNIAC computer) was probably the first serious attempt to simulate human theorem-proving behavior on a computer. Although the program could not model a brilliant logician, it made a significant landmark in the sense that it helped psychologists and computer scientists to gain deeper insight into problems and techniques related to understanding human behavior, and it started a whole new trend of symbol manipulation and so-called heuristic programming. To characterize the symbol-manipulation or information-processing approach to modeling, its techniques and methods are based on the conception that the brain is an information processor with sense organs as input channels, effector organs as output devices, and internal "programs" for testing, comparing, analyzing, transforming, and storing information. It treats the environment as a highly abstracted, symbolically represented mass of information. It is interesting to note that the Logic Theorist was accompanied by, or gave rise to, another development of great importance to artificial intelligence research--the first list-processing computer language, called Information Processing Language, or IPL.

Feigenbaum's EPAM program, or the Elementary Perceiver and Memorizer, models verbal learning behavior, specifically the learning of nonsense syllables; it simulates the information-processing activity underlying human discrimination and association learning. Feigenbaum's model accounts for many of the phenomena observed in these experiments. He has an interesting explanation of the "forgetting" phenomenon he observed in his model. He says, "Forgetting occurs in EPAM not because of information destruction but because learned material gets lost and inaccessible in a large and growing association network.... [Therefore,] forgetting occurs as a direct consequence of later learning by the learning processes. Furthermore, forgetting is only temporary: lost associations can be reconstructed [and therefore can be recalled] by storing more cue information [in the association network.]"

Samuel's checker-playing program, written for the IBM 7090, is an excellent example of the fact that programs can be written to exhibit "learning" behavior, though internally they contain no human-like problem-solving and learning processes. The program is able to improve its performance as it gains play-experience. The program defeated Robert Nealey, a former Connecticut checkers champion, in 1962. In this limited discussion, I cannot describe the program in detail, but a short characterization of the program is that it uses a linear polynomial expression with ingenious ways of selecting and changing its terms and of determining and modifying corresponding coefficients. The terms represent characteristics of the game situations

such as mobility, advancement, and center control. I do not know of any recent activities by Samuel, though I have heard from indirect sources that he has some new ideas but is currently too busy with administrative responsibilities.

The work of Gelernter and his associates at IBM extends heuristic programming ideas to the proof of theorems in Euclidean geometry. His program makes use of human-like heuristic methods where they are most effective, but it also applies more powerful and more direct symbol manipulation processes where these are useful. Therefore, he can also be placed in the second group I talked about earlier.

In the third group, some of the advocates of "man-machine symbiosis or synergism" can be placed, depending on their attitudes. Some of the activities under "Project MAC" at MIT seem to belong to this third group. On the other hand, those researchers who believe in the so-called "natural division" I spoke of earlier, and emphasize only a close interaction between man and machine, do not belong to the third group, in my opinion.

By this time, your reaction might be, "Checker-playing and theorem-proving may be good and fun, but what good is artificial intelligence research? Can the devices and techniques developed be used in the solution of real problems?" There have been some attempts in this line, a good representative of which is the assembly-line balancing program developed by Fred Tonge in 1960 on the RAND JOHNNIAC. This program is an application of heuristic programming to an important management science problem. Balancing an assembly line involves finding an efficient arrangement of workers, tasks, and work stations so as to maximize the rate of assembly or minimize the number of workers needed for a given rate of assembly. More-or-less straightforward procedures for doing this have been devised, but they are generally not practical, since they involve the enumeration of a very large number of tries--combinations of the work elements. Tonge's program differs from these in that it employs a variety of line-balancing heuristics--"tricks of the trade"--to simplify the constraints, to structure the assignment part of the problem, and to carry out the actual computations.

Again, I am anticipating your reaction which, this time, might be, "But all these programs are specialized in one problem or one specific area. Intelligence can not be evaluated by cleverness in just one specific problem situation, but requires versatility and adaptability to many situations." That is true. Critics from both outside and inside artificial intelligence research agreee that the main difficulty with artificial intelligence is that it cannot generalize beyond the very specific tasks for which programs are written and systems designed. The checker-playing program plays excellent checker games but cannot even understand the languages of much simpler games such as tic-tac-toe, much less play them (unless a large portion of the program is rewritten).

There are usually two explanations or defenses for the specificity of artificial intelligence systems. One is that by restricting the task environment to one problem or aspect of intellectual behavior, the task of investigation and evaluation is made much easier. After enough information about different aspects of behavior patterns is accumulated, some common features may be abstracted or theories may be formed that will serve to explain and predict in more general problem-solving situations. Criticism of the first explanation is that intelligence is not the mere aggregation of each capability separately studied and utilized. The mere accumulation of separately studied aspects of behavior will not give us a total picture of an intellect. Parallels are drawn with the old poem by John Godfrey Saxe about the six blind men feeling an elephant. The second explanation is that if artificial intelligence research is to be useful in solving some immediate problems, we must concentrate on single problems or even on portions of a given problem in order to exploit its special characteristics toward efficient solution procedures. In other words, the aim is practicality and efficiency of the program in terms of its outcome. Tonge's line-balancing program has been developed under this philosophy.

In spite of these two explanations, criticisms continue, and there have been attempts toward construction of more general intelligent systems. The General Problem Solver or GPS developed by Newell, Shaw, and Simon is the first large-scale attempt of this kind. Newell, Shaw, and Simon have used their previous experience with the Logic Theorist in the design of GPS; a deliberate effort has been made to abstract and separate some common features observed to be generally useful heuristics. These are kept separate from the problem-specific part of the program--generally referred to as the "task environment." With the general part of GPS and appropriate task environments, computers can be programmed to solve trigonometric identities, "missionary and cannibal" puzzles, to compile computer programs, and to prove theorems in the propositional calculus. One of the most extensively used general heuristics in GPS is the "means-ends analysis." An initial problem state is transformed into a target state by selecting and applying operations which, step-by-step, reduce the difference between the two states.

My computer program called Gaku is another attempt toward a more general system. GPS and Gaku use similar heuristics and a similar hierarchical structure of decision-making units. In Gaku, a stronger emphasis is placed on learning mechanisms than in GPS, and the system is highly structured, having four mechanisms that are clearly separable conceptually but are closely interacting. In order to test Gaku's performance, a sequence of problems taken from the Tower of Hanoi puzzle has been used. The experiment was successful in the sense that Gaku was able to find solutions to progressively more difficult Tower-of-Hanoi problems, each time utilizing its previous experience and finally finding a general solution

pattern through its induction mechanism. However, in terms of its ability to handle a wide variety of tasks, the desired generality and versatility of the system have not been achieved. A fairly detailed discussion of Gaku's design, its performance, and some of the difficulties to be overcome, can be found in an article entitled, "How a computer system can learn," which appeared in the July 1964 issue of the IEEE Spectrum.

An approach to increased generality is abstraction, as observed in mathematics; but for practical purposes, we must consider a reasonable balance or trade-off between efficiency and generality. When concepts and processes incorporated in the system are made abstract, the class of objects and behavioral actions they represent becomes larger, but at the expense of a less precise representation and less efficient uses of processes in a particular situation. A similar situation is observed in the different ways of finding a functional value, say "log x" for a given argument x. Should we store the functional values in table form, or should we store a log routine to generate log x for each given value x? To make the program even more abstract and general, should we store a derivation rule for a class of formulas to generate more than one function?

We employ such techniques of indirect specification when we construct relatively more general problem-solving systems. Since it is not feasible to prestore all the explicit responses required by all problem situations, some of which are not even conceived at the time the system is designed, we store a set of general rules for making decisions along with general information-handling programs. These prestored rules and programs determine implicitly what kinds of behavior patterns can be expected in particular situations. Of course, when a specific problem is given, a specific sequence of responses is eventually determined. This technique of indirect specification can be extended to learning mechanisms. Programs can be written to instruct the system to modify its own rules, or to modify the way in which it modifies rules, and so on through many levels of indirectness. Of course, an appropriate balance between specificity and generality must depend on the purpose of the researcher in artificial intelligence.

I shall try to conclude this sketchy survey talk by pointing out some of the difficulties that are significantly slowing the pace of artificial intelligence research. One is the difficulty of understanding the concept of similarity and hence the use of similarity judgments in solving problems. If a learning system is to make clever use of its past experience, it is essential for it to be able to discern the degree and aspects of similarity between two objects, two situations, and two processes. This is not a simple, clear-cut concept to define; two objects may be called similar in one instance and totally dissimilar in another.

Another difficulty is to find out how computers (and people) "learn" new heuristic methods and rules. Are there basic sets of a priori capabilities from which everything else can be evolved? If so, how are the higher-level capabilities developed from simpler ones? Any significant advancement in this area would offer the promise of enabling us to "bootstrap" our way into very much more powerful intelligent learning systems.

Inductive inference is another very powerful but very evasive human capability which we do not know how to mechanize. The products of generalization--that is, general rules or principles of unification--can be easily used and evaluated once they are discovered. The processes of generalization are another matter; only dimly do we understand the ways in which intelligence functions to discover regularities in nature or to form hypotheses from limited data.

Finally, perhaps the most urgently felt problem to overcome is the communication barrier between man and machine. As Simon has indicated, our dilemma is that we could design more intelligent machines if we could communicate with them better but we could communicate with them better if they were more intelligent.

As you can see from the examples in this talk, artificial intelligence is so far dealing only with extreme simplifications of complex processes and behavior. Simplification often leads to problems--unjustified assumptions may be made, or unsupported conclusion reached, on the basis of oversimplified models. But simplification is a necessity at the start, because we must start somewhere, somehow, and we are just beginning to get our feet wet in the sea of intellect. Artificial intelligence research poses tremendously difficult problems, but I believe that we can anticipate steady progress toward their solution.

REFERENCES

Amarel, S. On the automatic formation of a computer program which represents a theory. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein (Eds.), Self-organizing systems, Washington, D. C.: Spartan Books, 1962. Pp. 107-175.

Feigenbaum, E. A. and Feldman, J. (Eds.) Computers and thought. New York, N. Y.: McGraw-Hill Book Company, Inc., 1963. (Most examples of artificial-intelligence systems in the foregoing text were selected from this book.)

Hormann, A. M. Programs for machine learning, Part I. Information and Control, 1962, 5 (4), 347-367.

Hormann, A. M., et al. Gaku: An artificial student of problem solving. System Development Corporation document TM-1524/000/00, Santa Monica, California, September 1963.

Hormann, A. M. Programs for machine learning, Part II. Information and Control, 1964, 7 (1), 55-77.

Hormann, A. M. How a computer system can learn. IEEE Spectrum, 1964, 1 (7), 110-119.

Kelly, J. L., Jr. and Selfridge, O. G. Sophistication in computers: a disagreement. IRE Trans. Inform. Theory, 1962, IT-8, 78-80.

McCarthy, J. Programs with common sense. In Mechanisation of thought processes, Vol. 1, National Physical Laboratory Symposium No. 10, London: Her Majesty's Stationery Office, 1959. Pp. 75-84.

Miller, G. A., Galenter, E., and Pribram, K. H. Plans and the structure of behavior. New York: Henry Holt and Co., 1960.

Minsky, M. Learning systems and artificial intelligence. In Applications of logic to advanced digital computer programming, Ann Arbor: University of Michigan, College of Engineering, 1957.

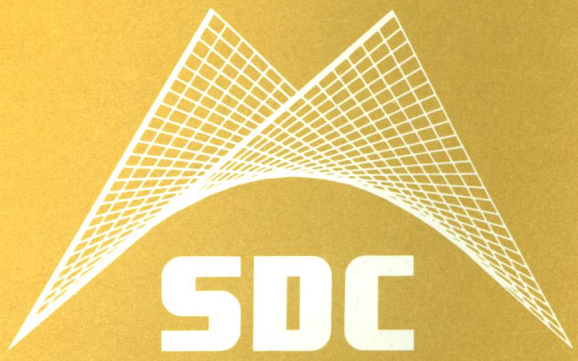
Newell, A. Some problems of basic organization in problem-solving programs. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein (Eds.), Self-organizing systems, Washington, D. C.: Spartan Books, 1962. Pp. 393-423.

Newell, A., et al. Information Processing Language-V manual (2nd ed.), Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1964

Newell, A., Shaw, J. C., and Simon, H. A. A variety of intelligent learning in a general problem solver. In M. C. Yovits and S. Cameron (Eds.), Self-organizing systems, London: Pergamon Press, 1960. Pp. 153-189.

Simon, H. A. Experiments with a heuristic compiler. ACM Journal, 1963, 10 (4), 493-506.

Simon, H. A. How computers can learn from experience. In Walter F. Freiberger and William Prager (Eds.), Applications of digital computers, Boston: Ginn and Co., 1963. Pp. 11-27.



SYSTEM DEVELOPMENT CORPORATION / 2500 COLORADO AVENUE / SANTA MONICA, CALIFORNIA 90406